
MarkLogic Server

Release Notes

Release 4.1
June, 2009

Last Revised: 4.1-4, December, 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.

This product includes 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 © 1998-2001 The OpenSSL Project. All rights reserved. Contains software derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm. Copyright © 1991-1992, RSA Data Security, Inc. Created 1991. All rights reserved.

Table of Contents

Release Notes

Copyright	2
1.0 Introduction	5
2.0 Installation and Upgrade	6
2.0.1 Supported Platforms	6
2.0.2 Upgrade Support	7
3.0 New Features in Release 4.1	8
3.1 SSL Support	8
3.1.1 SSL Encryption Over App Servers	8
3.1.2 Incoming SSL Connections Over xdm: http-get Functions	8
3.1.3 SSL Encryption for Communication Between Hosts in a Cluster	8
3.2 Expanded Support for RESTful Applications	9
3.2.1 URL Rewriting	9
3.2.2 Serializing JSON Objects	9
3.2.3 Enhancements to xdm:set-response-code and Other App Server Built-In Functions	9
3.3 Japanese Language Support	9
3.4 Portuguese Language Support Beginning in 4.1-2	9
3.5 Administrative Enhancements	10
3.5.1 Read-Only Forests	10
3.5.2 General Purpose Scheduler	10
3.6 Application Services	10
3.6.1 Search API With Support for Facets and Snippets	10
3.6.2 Application Builder	11
3.6.3 Library Services API	11
3.7 XQuery Enhancements	11
3.7.1 Full W3C XML Schema Validation	11
3.7.2 Function Values	11
3.7.3 More Robust Regular Expression Processing	12
3.7.4 xdm:pretty-print	12
3.7.5 xdm:elapsed-time	12
3.8 Search Enhancements	12
3.8.1 Control Over Term Frequency Normalization	12
3.8.2 More Geospatial Built-In Functions	13
3.8.3 cts:properties-query Constructor	13

3.8.4	Enhancements to the Spelling Suggestion Functionality	13
3.8.5	cts:walk and Enhancements to cts:highlight	13
3.9	Documentation Enhancements	13
4.0	Known Incompatibilities with Previous Releases	15
4.1	Forest Updates-Allowed Admin API Signature Changes	15
4.2	xdmp:eval/xdmp:invoke Changes for xdmp:set-response-code and Other App Server Built-In Functions	16
4.3	Regular Expression Changes	16
4.4	Spelling Dictionaries are Now Required to be in the Spell Namespace	17
4.5	spell:suggest Now Returns Suggestions Even If Word Is In the Dictionary	17
4.6	nobody User Has app-user Role	17
4.7	xdmp:directory-create Throws Exception if Directory Exists	17
4.8	XCC Now Requires Java 1.5 or Greater	17
4.9	Upgrade and Reindex Required	18
5.0	Other Notes	19
5.1	Memory and Disk Space Requirements	19
5.2	Compatibility with XQuery Specifications	20
5.3	XQuery Extensions	20
5.4	Documentation	20
5.5	Browser Requirements	22
5.6	Support	22

1.0 Introduction

MarkLogic Server 4.1 is a major release that includes many new features. The new features are described in “New Features in Release 4.1” on page 8. The following lists the major features with links to where they are described:

- [SSL Support](#)
- [Expanded Support for RESTful Applications](#)
- [Japanese Language Support](#)
- [Administrative Enhancements](#)
- [Application Services](#)
- [XQuery Enhancements](#)
- [Search Enhancements](#)

If you are upgrading from 4.0, some applications will require minor changes to run correctly on 4.1. For details, see “Known Incompatibilities with Previous Releases” on page 15.

For a list of bugs fixed in the latest maintenance release and a list of known bugs, see the Mark Logic Technical Support website at <http://support.marklogic.com> (supported customers only).

2.0 Installation and Upgrade

This chapter describes the supported platforms and upgrade paths for MarkLogic Server, and has the following sections:

- [Supported Platforms](#)
- [Upgrade Support](#)

2.0.1 Supported Platforms

This release of MarkLogic Server is supported on the following platforms:

- Microsoft Windows Server 2008 (x86), Microsoft Windows 2003 Server (x86), Microsoft Windows XP SP2, Microsoft Windows Vista 32-bit Edition (x86)*
- Microsoft Windows Server 2008 (x64), Windows 2003 Server 64-bit Edition (x64), Windows Vista Server 64-bit Edition (x64)*
- Sun Solaris 10 (64-bit SPARC)
- Sun Solaris 10 (x64)
- Red Hat Enterprise Linux 4.0 and 5.0 (x86)** ***
- Red Hat Enterprise Linux 4.0 and 5.0 (x64)** *** ****
- CentOS 5 (x64)** ***

* Microsoft Windows Vista is supported for development only.

If MarkLogic Server fails to start up on Windows with the error “the application failed to initialize properly (0xc0150002)”, then a dependency is missing from your environment and you need to download and install one of the following DLLs:

32-bit versions of Windows require the DLL at the following link:

<http://www.microsoft.com/downloads/details.aspx?familyid=200B2FD9-AE1A-4A14-984D-389C36F85647&displaylang=en>.

64-bit versions of Windows require the following DLL:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=eb4ebe2d-33c0-4a47-9dd4-b9a6d7bd44da&DisplayLang=en>.

** The deadline I/O scheduler is required on Red Hat Linux platforms. The deadline scheduler is optimized to ensure efficient disk I/O for multi-threaded processes, and MarkLogic Server can have many simultaneous threads. For information on the deadline scheduler, see the Red Hat documentation (for example,

<http://www.redhat.com/docs/wp/performance/iotuning/iosubsystem-scheduler-deadline.html>,
<http://www.redhat.com/docs/wp/performance/iotuning/iosubsystem-scheduler-selection.html>, and
<http://www.redhat.com/magazine/008jun05/features/schedulers/>).

***The `redhat-lsb` and the `glibc` packages are required on Red Hat Linux. Additionally, on 64-bit Red Hat Linux, both the 32-bit and the 64-bit `glibc` packages are required.

****Red Hat Linux 5.0 (x64) is also supported in a VMWare ESX 3.0.2 (installed on bare metal) environment.

2.0.2 Upgrade Support

This section describes upgrade support to 4.1. For details on installing MarkLogic Server and for the upgrade procedure, see the *Installation Guide*.

Upgrading from an Early Access Release of 4.1 is not supported; if you loaded data in an Early Access Release, you must reload it in the production release of 4.1.

Upgrading is supported from 4.0-1 or later to 4.1-1 or later. If you are running a release prior to 4.0, you must first upgrade to 4.0 before upgrading to 4.1. If you are upgrading an Enterprise Edition cluster, you must first upgrade the node in which the Security database forest is located before you upgrade other nodes in the cluster.

An upgrade from 4.0 to 4.1 will reindex any databases that have `reindex enable` set to `true`. If you choose not to reindex your databases, they will run in either 3.0, 3.1, 3.2, or 4.0 compatibility mode, depending on the version of MarkLogic Server in which they were last loaded or reindexed. Running in compatibility mode will disable certain 4.1 features and may treat all content in the database as English language content. For details on database compatibility, see the *Installation Guide*.

There are some known incompatibilities between 4.0 and 4.1. You might need to make some minor code changes to your 4.0 applications before they can run correctly in 4.1. For details on the incompatibilities, see “Known Incompatibilities with Previous Releases” on page 15. For instructions on upgrading to 4.1, including information about database compatibility between 4.1 and 4.0, see the *Installation Guide*.

3.0 New Features in Release 4.1

This chapter describes the new features in Release 4.1 of MarkLogic Server. The feature descriptions are divided into the following categories:

- [SSL Support](#)
- [Expanded Support for RESTful Applications](#)
- [Japanese Language Support](#)
- [Portuguese Language Support Beginning in 4.1-2](#)
- [Administrative Enhancements](#)
- [Application Services](#)
- [XQuery Enhancements](#)
- [Search Enhancements](#)
- [Documentation Enhancements](#)

3.1 SSL Support

MarkLogic Server 4.1 includes support for the HTTPS protocol in the following areas:

- [SSL Encryption Over App Servers](#)
- [Incoming SSL Connections Over xdm:http-get Functions](#)
- [SSL Encryption for Communication Between Hosts in a Cluster](#)

3.1.1 SSL Encryption Over App Servers

In 4.1, you can set up an App Server (HTTP, XDBC, or WebDAV) to use Secure Socket Layer (SSL) encryption. SSL encryption provides secure transmission between browsers (or other clients) and App Servers. For details on configuring SSL for an App Server, see [Configuring SSL on App Servers](#) in the *Administrator's Guide*.

3.1.2 Incoming SSL Connections Over xdm:http-get Functions

You can get data via a secure HTTPS connection using the `xdm:http-get` and other `xdm:http-*` functions. For details on the `xdm:http-*` functions, see the *Mark Logic Built-In and Module Functions Reference*.

3.1.3 SSL Encryption for Communication Between Hosts in a Cluster

You can set up a cluster to use SSL over the XDQP protocol to encrypt communication between hosts. You can use the Admin Interface to set up SSL over XDQP on the configuration page for each group. For details on setting up SSL over XDQP, see [Enabling SSL communication over XDQP](#) in the *Administrator's Guide*.

3.2 Expanded Support for RESTful Applications

4.1 has several features to enhance support to easily create RESTful applications.

- [URL Rewriting](#)
- [Serializing JSON Objects](#)
- [Enhancements to `xdmp:set-response-code` and Other App Server Built-In Functions](#)

3.2.1 URL Rewriting

4.1 includes URL rewriting, which allows you to intercept a request to an HTTP App Server and change the URL it accesses, without the browser seeing the rewritten URL. URL rewriting makes it easy to create applications with clean URLs, allowing the developer to decide what URL is needed to access a resource regardless of the implementation of that resource. For details, see [Setting Up URL Rewriting for an HTTP App Server](#) in the [Controlling App Server Access, Output, and Errors](#) chapter of the *Application Developer's Guide*.

3.2.2 Serializing JSON Objects

4.1 includes the `xdmp:to-json` and `xdmp:from-json` functions, which allow you to serialize XQuery datatypes as a JSON string and vice-versa. JSON (JavaScript Object Notation) is a data-interchange format which is designed to pass data to and from JavaScript. For details, see the [JSON: Serializing To and Parsing From](#) chapter of the *Application Developer's Guide* and, for the signatures and description of each function, see the *Mark Logic Built-In and Module Functions Reference*.

3.2.3 Enhancements to `xdmp:set-response-code` and Other App Server Built-In Functions

The App Server built-in functions can now set App Server functions within an `xdmp:eval` or `xdmp:invoke` operation. Previously, many of these functions did not change the App Server request when called in an `xdmp:eval` or `xdmp:invoke` operation.

3.3 Japanese Language Support

There is now advanced language support for Japanese. See the [Encodings and Collations](#) chapter of the *Search Developer's Guide* for a list of sample collations and character sets in the Japanese language.

3.4 Portuguese Language Support Beginning in 4.1-2

In release 4.1-2 and later, there is advanced language support for Portuguese. See the [Encodings and Collations](#) chapter of the *Search Developer's Guide* for a list of sample collations and character sets in the Portuguese language.

3.5 Administrative Enhancements

4.1 includes the following administrative enhancements.

- [Read-Only Forests](#)
- [General Purpose Scheduler](#)

3.5.1 Read-Only Forests

You can now mark a forest `read-only` and `flash-backup` for its `updates` allowed states, in addition to `all` and `delete-only`. When a forest is in the `read-only` or `flash-backup` state, any content in that forest cannot be changed in any way, which also means that merges cannot occur on the forest. The `flash-backup` state is designed to temporarily stop any writing to the forest so a flash-backup can take place, and any updates that are attempted while a forest is in the `flash-backup` state will retry until the retry limit is reached; when a forest in the `read-only` state, updates are not retried, and any updates attempted will immediately throw an exception. For details, see the [Forests](#) chapter of the *Administrator's Guide*.

3.5.2 General Purpose Scheduler

4.1 provides a general purpose scheduler, which allows you to set up a time to run an arbitrary XQuery job. Like the `cron` scheduler on UNIX, you can set up jobs to run at periodic intervals. For details, see the [Scheduling Tasks](#) chapter of the *Administrator's Guide*.

3.6 Application Services

MarkLogic Server 4.1 includes Application Services, which is a set of APIs and tools to make it easier to build search applications. The following are some Application Services features:

- [Search API With Support for Facets and Snippets](#)
- [Application Builder](#)
- [Library Services API](#)

3.6.1 Search API With Support for Facets and Snippets

The Search API makes it easy to create complex search application. The Search API includes support for faceted navigation, automatic query parsing, built-in snippeting, and many search features. For details on the Search API, see [Search API: Understanding and Using](#) in the *Search Developer's Guide* and for the Search API function signatures see *Mark Logic Built-In and Module Functions Reference*.

3.6.2 Application Builder

Application Builder is a browser-based application that allows you to create a fully functional search and analytics application very quickly. With a database of your own content, you can create a sophisticated search application very rapidly using Application Builder. To get started with Application Builder, see [Application Builder Quick Start](#) in the *Application Builder Developer's Guide*.

As part of Application Builder, you can build the Oscars sample application. If you select the Oscars template and take all of the defaults, you will generate the Oscars sample application, which is a search application based on Wikipedia data about the Oscar awards. For details, see [Building the Oscars Sample Application](#) in the *Application Builder Developer's Guide*.

3.6.3 Library Services API

The Library Services API is designed to build check out/check in applications that allow versioning of documents. The Library Services API is an XQuery module, and it makes it easy to build applications that manage different versions of content. For details, see [Library Services Applications](#) in the *Application Developer's Guide*.

3.7 XQuery Enhancements

Various XQuery enhancements.

- [Full W3C XML Schema Validation](#)
- [Function Values](#)
- [More Robust Regular Expression Processing](#)
- [xdmp:pretty-print](#)
- [xdmp:elapsed-time](#)

3.7.1 Full W3C XML Schema Validation

In 4.1, you can now validate complex W3C XML schema types as well as simple types using the XQuery `validate` expression. Additionally, as an extension to the XQuery 1.0 `validate` expression, there is a `validate as` expression in the 1.0-m1 XQuery dialect and a corresponding pragma in both the 1.0 and 1.0-m1 dialects. For details on the `validate` expression, see the [Validate Expression](#) section in the *XQuery Reference Guide* and the W3C XQuery Recommendation (<http://www.w3.org/TR/xquery/#id-validate>).

3.7.2 Function Values

Function Values are implemented as an XQuery datatype, allowing you to reference functions as variables, pass a function value as a parameter to other functions, and return function values from function calls. For details, see `xdmp:apply` and `xdmp:function` in the *Mark Logic Built-In and Module Functions Reference*, as well as the [Function Values](#) chapter of the *Application Developer's Guide*.

3.7.3 More Robust Regular Expression Processing

The regular expression processing in 4.1 has been enhanced, so functions like `fn:matches` and `fn:replace` are both more conformant to the XQuery specification and have better performance characteristics. The conformance changes can also cause some minor incompatibilities with 4.0 applications, as described in “Regular Expression Changes” on page 16.

3.7.4 `xdmp:pretty-print`

The `xdmp:pretty-print` function formats an XQuery module for printing. This function is useful for applications that generate XQuery code (for example, Application Builder uses it for the code it generates).

3.7.5 `xdmp:elapsed-time`

The `xdmp:elapsed-time` function returns the elapsed time since the start of processing of a query. It provides the same information as the `elapsed-time` element of the `xdmp:query-meters` output, but has less overhead than calling `xdmp:query-meters`.

3.8 Search Enhancements

The following enhancements to the search functionality in MarkLogic Server are included in 4.1:

- [Control Over Term Frequency Normalization](#)
- [More Geospatial Built-In Functions](#)
- [cts:properties-query Constructor](#)
- [Enhancements to the Spelling Suggestion Functionality](#)
- [cts:walk and Enhancements to cts:highlight](#)

3.8.1 Control Over Term Frequency Normalization

MarkLogic Server calculates relevance based on the frequency with which the terms in the search appear in a document. By default, the term frequency is normalized based on the size of the document, so larger documents would need relatively more terms to end up with a score equivalent to a smaller document. The idea is that larger documents do not overshadow smaller documents. For most applications, this algorithm produces excellent relevance rankings.

For some content, however, you might care only about absolute term frequency, and not care if larger documents overshadow smaller documents. In such cases, you might want the term frequency to be a direct function of the number of terms in the document, not a normalized version. 4.1 provides the capability to change the term frequency normalization with the `tf normalization` setting in the database configuration. The default is `scaled-log`, which normalizes the term frequency based on the document size, and you can also set it to `unscaled-log`, which does not normalize the term frequency. Both methods use a log function of the term frequency, and that goes into the relevance calculation. If you change the setting, the database must complete reindexing before all searches will reflect the newly calculated relevance scores.

3.8.2 More Geospatial Built-In Functions

4.1 adds the following geospatial calculation functions:

- `cts:shortest-distance`
- `cts:bearing`
- `cts:arc-intersection`
- `cts:destination`

These functions complement the `cts:distance` function, and return calculated information from geospatial input.

3.8.3 `cts:properties-query` Constructor

4.1 includes the new `cts:query` constructor `cts:properties-query`. A `cts:properties-query` matches when the `cts:query` passed into it matches in the properties of the documents passed into the `cts:search`, providing a way to compose queries over properties and documents in a search. Such searches effectively provide an efficient way to join properties with their corresponding documents.

3.8.4 Enhancements to the Spelling Suggestion Functionality

There are several enhancements to the spelling suggestion functionality in 4.1. You can now control the number of suggestions returned from `spell:suggest`, and you can use the `spell:suggest-detailed` function to get a detailed report about the spelling suggestions which include the Levenstein distance, information about how the terms match the double-metaphone algorithm, and other information. For more details about the spelling correction functions, see [Using the Spelling Correction Functions](#) in the *Search Developer's Guide*.

Also, the spelling correction functions in 4.1 are stricter about some rules for the namespace of the dictionary documents, which can cause an incompatibility with 4.0, as described in “Spelling Dictionaries are Now Required to be in the Spell Namespace” on page 17.

3.8.5 `cts:walk` and Enhancements to `cts:highlight`

A new built-in, `cts:walk`, and some enhancements to `cts:highlight` allow more robust support for multi-pass highlighting. Multi-pass highlighting is useful for many purposes, including creating a condensed summary of a document (known as a snippet). To support this functionality, two new variables, `$cts:start` and `$cts:break`, have been added to `cts:highlight` (and they are also part of `cts:walk` and `cts:entity-highlight`). For details on `cts:highlight` and `cts:walk`, see the *Mark Logic Built-In and Module Functions Reference* and the [Highlighting Search Term Matches](#) chapter of the *Search Developer's Guide*.

3.9 Documentation Enhancements

MarkLogic Server 4.1 includes a two new guides, the *Application Builder Developer's Guide* and the *Search Developer's Guide*.

The *Application Builder Developer's Guide* ([Application Builder Developer's Guide](#)) describes how to use the new Application Builder, including information on generating the Oscars sample application and on customizing generated application.

The *Search Developer's Guide* ([Search Developer's Guide](#)) consolidates all of the search functionality from the 4.0 *Developer's Guide* into a single guide, and also includes detailed information on the new Search API. The rest of the material from the 4.0 *Developer's Guide*, in addition to new 4.1 material, has been reorganized into the 4.1 *Application Developer's Guide*.

4.0 Known Incompatibilities with Previous Releases

The vast majority of applications implemented on MarkLogic Server 4.0-* will run either without modifications or with very minor modifications in Release 4.1. There are, however, a number of changes that will cause compatibility issues with 4.0 applications. This section describes those incompatibilities and includes the following topics:

- [Forest Updates-Allowed Admin API Signature Changes](#)
- [xdmp:eval/xdmp:invoke Changes for xdmp:set-response-code and Other App Server Built-In Functions](#)
- [Regular Expression Changes](#)
- [Spelling Dictionaries are Now Required to be in the Spell Namespace](#)
- [spell:suggest Now Returns Suggestions Even If Word Is In the Dictionary](#)
- [nobody User Has app-user Role](#)
- [xdmp:directory-create Throws Exception if Directory Exists](#)
- [XCC Now Requires Java 1.5 or Greater](#)
- [Upgrade and Reindex Required](#)

4.1 Forest Updates-Allowed Admin API Signature Changes

The following functions for the Admin API have different signatures from 4.0, and are therefore incompatible:

- `admin:forest-get-updates-allowed`
- `admin:forest-set-updates-allowed`

In 4.1, there are four values for `updates-allowed`; in 4.0, it was a boolean value. If you have code that uses these APIs, you must modify the code to use the new values. The `$value` parameter of `admin:forest-set-updates-allowed` can now have the following values:

- `all` (was `true` in 4.0)
- `delete-only` (was `false` in 4.0)
- `read-only`
- `flash-backup`

The new values reflect the ability to mark a forest read-only in 4.1, as described in “Read-Only Forests” on page 10.

4.2 `xdmp:eval/xdmp:invoke` Changes for `xdmp:set-response-code` and Other App Server Built-In Functions

Previously, `xdmp:eval` and `xdmp:invoke` ignored any calls to the various App Server Built-Ins that change or get state from the response or the request body (for example, `xdmp:set-response-code`, `xdmp:redirect-response`, and so on). In 4.1, these calls work in an `xdmp:eval` or an `xdmp:invoke` the same way they do when you run a query in any other context. If you have applications that relied on the old behavior (that is, applications that used App Server Built-In functions in an `xdmp:eval/xdmp:invoke` but expected them to be no-ops), then you might need to rewrite those `xdmp:eval` or `xdmp:invoke` statements.

4.3 Regular Expression Changes

The regular expression evaluation in 4.1 has been improved and is more efficient than in 4.0. It is also more conformant to the XQuery specification than 4.0. Some of these conformance changes will cause some regular expressions to behave differently in 4.1 than they did in 4.0. Regular expressions are used in the `fn:matches`, `fn:tokenize`, and `fn:replace` functions. Some of the changes are as follows:

- You can no longer match the empty string in a regular expression with `fn:replace` or `fn:tokenize` (you can with `fn:matches`, however). Previously, the empty string in `fn:replace` and `fn:tokenize` matched everything, but in 4.1 it throws an `XDMP-MATCHZERO` exception.
- If you place an invalid escape sequence in a regular expression, an exception is raised. In 4.0, some invalid escape sequences (for example, `\/`) were allowed. In 4.1, any invalid escape sequence throws an exception.
- Certain invalid character classes are no longer allowed. For example, the regular expression `[z-a]` is acceptable in 4.0 (although it does nothing), but throws an exception in 4.1. All invalid character classes now throw an exception.

For example, each of the following expressions contains an invalid regular expression, and throws an exception in 4.1 but completes in 4.0:

```
xquery version "1.0-ml";

fn:replace("", "\s*", "x"),
fn:replace("http://marklogic.com", "\/", "X"),
fn:replace("abc", "[z-a]", "z")
(:
  Throws exception in 4.1, returns the following in 4.0:
  x
  http:XXmarklogic.com
  abc
:)
```

If you have any code that uses these regular expressions, you should review the regular expressions and rewrite it as needed.

4.4 Spelling Dictionaries are Now Required to be in the Spell Namespace

When using the spelling correction functions in 4.1, the dictionary documents must be in the `http://marklogic.com/xdmp/spell` namespace. Previously, the dictionary documents were supposed to be in this namespace, but in some cases it still worked if it was in no namespace. If you have dictionaries that are not in the `spell` namespace, then you must transform them to be in that namespace for them to work correctly in 4.1.

4.5 `spell:suggest` Now Returns Suggestions Even If Word Is In the Dictionary

The `spell:suggest` function now returns spelling suggestions when a word appears in the dictionary (that is, when it is spelled correctly). In 4.0, `spell:suggest` returns the empty sequence if the word is in the dictionary. To find out if a word is in the dictionary (that is, to find if a word is spelled correctly), use the `spell:is-correct` function. If you have code that relies on the old behavior of `spell:suggest` returning an empty sequence for a correctly spelled word, you should rewrite that code to use `spell:is-correct` in 4.1.

4.6 `nobody` User Has `app-user` Role

Upgrading to 4.1 adds the `app-user` role to the `nobody` user. The `app-user` role has privileges to run the `xdmp:value` and `xdmp:with-namespaces` functions. If you have any applications that use the `nobody` user, those applications will now run with more privileges than they had in 4.0 (unless you had manually changed the roles assigned to the `nobody` user). The `nobody` user is, by default, the default user for App Servers before authentication takes place. If you do not want the `nobody` user to have these privileges, then you can modify the `nobody` user after the upgrade.

Application Builder relies on the `nobody` user having these privileges, however, so if you modify the `nobody` user (or the `app-user` role), then Application Builder might not work as expected. Additionally, Application Builder uses the `app-user` role to put execute permissions on the error handler, and if the `nobody` user does not have the `app-user` role then no users will be able to log in to applications built with Application Builder.

4.7 `xdmp:directory-create` Throws Exception if Directory Exists

In 4.1, if you try to create a directory (using `xdmp:directory-create`, for example) and the directory already exists, then an exception is thrown (`XDMP-DIREXISTS`). In 4.0, the directory was re-created, and no exception was thrown. If you have code that relies on the old behavior, you must modify that code to make it handle the exception when the directory exists.

4.8 XCC Now Requires Java 1.5 or Greater

The 4.1 Java XCC libraries now require Java 1.5 or greater. Previously they required Java 1.4 or greater. If you have XCC applications that run with Java 1.4, in order to use them with a 4.1 XCC package, you will need to upgrade those application environments to use Java 1.5 or greater. If you are still using a 4.0 XCC package, then the applications should continue to work, although you will not be able to use any of the 4.1-specific XCC feature (SSL, for example).

4.9 Upgrade and Reindex Required

When you log into the Admin Interface after installing 4.1, you will be prompted to upgrade the security database and the configuration files. The 4.1 security database and configuration files are not backwards compatible with 4.0. Make sure to do a full backup of your databases and data directory before upgrading. If you do not want to reindex after upgrading, turn off reindexing for each database in 4.0 before installing 4.1. For details and index compatibility, see [Upgrades and Database Compatibility](#) in the *Installation Guide*.

Warning Upgrading from a 4.1 Early Access release to 4.1 is not supported.

5.0 Other Notes

This section provides the following information about MarkLogic Server:

- [Memory and Disk Space Requirements](#)
- [Compatibility with XQuery Specifications](#)
- [XQuery Extensions](#)
- [Documentation](#)
- [Browser Requirements](#)
- [Support](#)

5.1 Memory and Disk Space Requirements

MarkLogic Server requires at least 512 MB of system memory.

The first time it runs, MarkLogic Server automatically configures itself to the amount of memory on the system, reserving as much as it can for its own use. If you need to change the default configuration, you can manually override these defaults at a later time using the Admin Interface.

Mark Logic recommends the following two guidelines for server sizing:

- Configure your server with 1 GB of physical memory for every 16 GB of source content you expect to manage.
- Configure your server with at least one CPU (or core) per 100 GB of source content.

Pragmatically, we recommend running most configurations with a minimum of two CPUs (or two cores).

MarkLogic Server requires 3 times the disk space of the total forest size. Specifically, each forest on a filesystem requires its filesystem to have at least 3 times the forest size in disk space. This translates to three times the disk space of the source content after it is loaded. For example, if you plan on loading content that will result in a 10 GB database, reserve at least 30 GB of disk space. The disk space reserve is required for merges.

On UNIX systems, MarkLogic Server requires swap space at least equal to the amount of physical memory on the machine. Swap space equal to twice the amount of physical memory is highly recommended. For example, if you have a UNIX machine with 32 GB of memory, you should ideally configure the swap space to be 64 GB (and at least 32 GB). This is true on Windows systems also, but the system is normally set up to grow the swap (page) file as needed.

For more details about memory, disk, and swap requirements, see [Memory, Disk Space, and Swap Space Requirements](#) in the *Installation Guide*.

5.2 Compatibility with XQuery Specifications

This release implements the XQuery language, functions and operators specified in the W3C XQuery 1.0 Recommendations:

- <http://www.w3.org/TR/xquery/>
- <http://http://www.w3.org/TR/xquery-operators/>

Additionally, there is backwards compatibility with the May 2003 version of the XQuery 1.0 Draft specification used in MarkLogic Server 3.2 and previous versions. For details on the XQuery implementation in MarkLogic Server 4.1, including the three different dialects supported, see the *XQuery Reference Guide*.

5.3 XQuery Extensions

Working within the W3C XQuery 1.0 Recommendation, Mark Logic has created a number of language extensions enabling key functionality not supported in the current release of the language specification. These extensions provide transactional update capabilities, assorted search and retrieval features, various data manipulation functions, and administrative tools.

The extensions, as well as the XQuery standard functions, are documented at <http://developer.marklogic.com>.

5.4 Documentation

MarkLogic Server includes the following documentation, available through the support web site and through <http://developer.marklogic.com/>:

Documentation	Description
<i>Installation Guide</i>	Provides procedures for installing MarkLogic Server.
<i>Administrator's Guide</i>	Provides procedures for administrative tasks such as creating servers, creating databases, backing up databases, creating users, setting up your security policy, and so on.
<i>Application Developer's Guide</i>	Provides procedures, methodologies, and conceptual information about general MarkLogic Server application development tasks.
<i>Search Developer's Guide</i>	Provides procedures, methodologies, and conceptual information about search-based application development tasks.
<i>Application Builder Developer's Guide</i>	Provides step-by step information on using Application Builder to build a search application, and also provides information on customizing applications built with Application Builder.

Documentation	Description
<i>Content Processing Framework</i>	Provides an introduction to the Content Processing Framework and procedures for installing the default content processing framework.
<i>Understanding and Using Security</i>	Provides information on the role-based security model in MarkLogic Server.
<i>Query Performance and Tuning</i>	Provides performance-related information that is useful to application developers and administrators.
<i>Scalability, Availability, and Forest-Level Failover</i>	Provides information on large-scale system architecture, clustering, availability, and details on setting up forest-level failover.
<i>Mark Logic Built-In and Module Functions Reference</i>	API documentation for the Mark Logic built-in and module extensions to the XQuery standard functions, as well as API documentation for the W3C functions implemented in MarkLogic Server.
<i>4.1 Release Notes</i>	Contains a summary of new features, upgrade compatible information, and known issues.
XCC Javadoc and .Net C# API Documentation	API documentation for the Mark Logic XML Contentbase Connector for Java API (XCC/J) and .Net XCC C# API documentation.
<i>XCC Developer's Guide</i>	An overview of the what you can do with the XCC libraries, examples of how to use XCC, and an overview of the sample applications included with XCC.
<i>XQuery Reference Guide</i>	A condensed overview of the XQuery language, including information on the three XQuery dialects in MarkLogic Server. This book does include some syntax information, although it is primarily intended as in introduction and quick-reference to the language, not as a comprehensive reference.
<i>Getting Started with MarkLogic Server</i>	A quick, step-by-step overview of how to get up and running with MarkLogic Server.

XQuery language documentation is provided through the W3C working group drafts specified in “Compatibility with XQuery Specifications” on page 20. Sample code is provided through the demo server at <http://localhost:8000/>, which is automatically installed as part of the MarkLogic Server installation process. Additionally, there are many samples available on the Mark Logic developer site (<http://developer.marklogic.com>).

XQuery language extensions specific to MarkLogic Server are documented online in the *Mark Logic Built-In and Module Functions Reference*. Example code snippets are provided as part of that documentation. The Admin Interface provides a large-scale example of complex XQuery programming, using many of the MarkLogic Server XQuery language extensions.

The Admin Interface includes built-in help screens that explain the purpose of the various controls and parameters in the Admin Interface.

Known bugs are documented online as we find them or as they are reported to us. See <http://support.marklogic.com> (supported customers only) for more details.

5.5 Browser Requirements

The MarkLogic Server Admin Interface is supported on Internet Explorer 6 or newer and Firefox browsers.

Application Builder and the applications generated with Application Builder are supported on Internet Explorer 6 or 7 on Windows, Firefox 3.0 or newer on Windows, and Safari 3 or newer on Mac OS 10.5. Other browser/platform combinations may work but are not as thoroughly tested.

Authentication functions in MarkLogic Server are supported for Internet Explorer 6 or newer and Firefox.

5.6 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.

For registered customers, we invite you to visit our support website at <http://support.marklogic.com> to access our full suite of documentation and help materials. For all customers, including community licensed customers, visit the Mark Logic Developer’s site at <http://developer.marklogic.com>, which includes full product documentation, downloads, and developer community open-source projects.

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.