
MarkLogic Server

Release Notes

Release 4.0
September, 2008

Last Revised: 4.0-5, April, 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 Icen 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

Release Notes

Copyright	2
1.0 Introduction	6
2.0 Installation and Upgrade	7
2.0.1 Supported Platforms	7
2.0.2 Upgrade Support	8
3.0 New Features in Release 4.0	9
3.1 XQuery 1.0 Support	9
3.2 Geospatial Search and Analytics	10
3.3 Alerting Infrastructure	10
3.4 Advanced Analytics	11
3.4.1 Value Lexicons For Elements and Attributes of All Datatypes	11
3.4.2 Sample=N Option for Lexicon APIs	11
3.4.3 Element and Attribute Co-Occurrence Lexicons	11
3.4.4 Range Lexicons	12
3.4.5 Geospatial Lexicons	12
3.5 Entity Enrichment	13
3.6 Positions on Attribute Values	13
3.7 Application Development Features	13
3.7.1 Maps in XQuery	14
3.7.2 Evaluate XQuery Expressions With xdm:value	14
3.7.3 SGML Entity Output Capability	14
3.7.4 Choose Output Encoding	14
3.7.5 Custom Error Pages For HTTP Servers	14
3.7.6 Support for Multi-Part Mime Types (for File Upload Applications)	15
3.7.7 cts:element-query Scope Now Includes Child Attributes	15
3.7.8 xdm:get-request-body Now Allows Binary, XML, or Text Content	15
3.7.9 More Built-In XQuery Functions	15
3.8 Modular Documents	16
3.9 Administrative Enhancements	16
3.9.1 Scriptable Admin API	16
3.9.2 Non-Insertable (Read and Delete Only) Forests	17
3.9.3 Forest-Level Failover For High Availability	17
3.9.4 Disable/Enable Databases, Forests, and App Servers	17
3.9.5 Improvements to the Admin Interface	17

3.9.6	Scheduled Backups	17
3.9.7	XQuery Built-Ins for Admin Operations (Including Backup and Restore) .. 18	18
3.10	Enhanced Security Features	18
3.10.1	Auditing	18
3.10.2	Request Blackouts	18
3.10.3	Show and Change Permissions in the Admin Interface	18
3.10.4	New Privilege for xdm:add-response-header	18
3.11	Function Mapping	19
4.0	Known Incompatibilities with Previous Releases	20
4.1	XQuery Compatibility	20
4.1.1	XQuery Version Declaration	21
4.1.2	0.9-ml Syntax Remains the Same as 3.2	21
4.1.3	New Prolog Syntax for 1.0-ml and 1.0	22
4.1.4	Triggers Now Default to 1.0-ml, So Trigger Modules Must Be Modified ... 22	22
4.1.5	Effective Boolean Value Changes	22
4.1.6	Function Mapping	23
4.1.7	New Privilege on xdm:add-response-header	23
4.1.8	element() Test in 0.9-ml Equivalent to schema-element() Test in 1.0-ml	24
4.1.9	Namespace Declaration Attributes (xmlns) in Element Constructors Must Now be Literal Values	24
4.1.10	MarkLogic Server err Prefix Bound to a Different Namespace in 1.0-ml	24
4.1.11	XML Serializaton Change for Default Element Namespace With Prefix	26
4.1.12	Can No Longer Bind No Namespace to a Prefix	26
4.1.13	Ampersand Character (&) Must Be Escaped in String Literals in 1.0 and 1.0-ml	27
4.1.14	p:state-transition and p:status-transition Now Have Priority Parameter	27
4.1.15	Less-Than Character (<) No Longer Allowed in Attribute Values	27
4.2	Functional Behavior Changes	28
4.2.1	Ordered cts:near-query With Distance of 0 Now Matches Itself	28
4.2.2	cts:element-query Now Includes Attributes of Element in its Scope	29
4.2.3	Lexicon Functions Now Default to Document Values, Was Any Value	29
4.2.4	Functions that Now Require Execute Privileges	30
4.2.5	Maximum Length of xs:float and xs:double Casts	30
4.2.6	xdmp:describe Differences Due to the xs Namespace Prefix Replacing the xdt Prefix	31
4.2.7	Function Calls Now Check Return Types	31
4.2.8	Can No Longer Load XML Documents With Multiple Root Nodes	32
4.2.9	xdmp:get-request-body Now Returns an item()*	32
4.2.10	repair Option of xdm:document-load, xdm:unquote, and Others Now Default to none in 1.0 and 1.0-ml	33
4.2.11	Attribute Whitespace No Longer Normalized	33
4.2.12	HTTP and WebDAV Servers Now Default to Digest Authentication	33
4.2.13	CPF State Changes for Default Conversion Option	33

4.2.14	XCC for Java SPI ConnectionProvider obtainConnection Method Has Different Signature	34
4.2.15	XCC Duration Type References that Used XDT Now Use XS	34
4.3	Upgrade and Reindex Required	34
5.0	Other Notes	35
5.1	Memory and Disk Space Requirements	35
5.2	Compatibility with XQuery Specifications	36
5.3	XQuery Extensions	36
5.4	Documentation	36
5.5	Browser Requirements	38
5.6	Support	38

1.0 Introduction

MarkLogic Server 4.0 is a major release that includes many new features. The new features are described in “New Features in Release 4.0” on page 9. The following lists the major features with links to where they are described:

- [XQuery 1.0 Support](#)
- [Geospatial Search and Analytics](#)
- [Alerting Infrastructure](#)
- [Advanced Analytics](#)
- [Application Development Features](#)
- [Administrative Enhancements](#)

If you are upgrading from 3.2, some applications will require minor changes to run correctly on 4.0. For details, see “Known Incompatibilities with Previous Releases” on page 20.

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 9 and 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)** *** ****

* 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 4.0. For details on installing MarkLogic Server and for the upgrade procedure, see the *Installation Guide*.

Upgrading from an Early Access Release of 4.0 is not supported; if you loaded data in an Early Access Release, you must reload it in the production release of 4.0.

Upgrading is supported from 3.2-6 or later to 4.0-1 or later. If you are running a release prior to 3.2-6, you must first upgrade to 3.2-6 or later before upgrading to 4.0; the 3.2 release must be 3.2-6 or later. 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 3.2 to 4.0 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, or 3.2 compatibility mode, depending on the version of MarkLogic Server in which they were last loaded or reindexed. Running in compatibility mode will disable certain 4.0 features (for example, attribute value positions) 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 3.2. You might need to make some minor code changes to your 3.2 applications before they can run correctly in 4.0. For details on the incompatibilities, see “Known Incompatibilities with Previous Releases” on page 20. For instructions on upgrading to 4.0, including information about database compatibility between 4.0 and 3.2, see the *Installation Guide*.

3.0 New Features in Release 4.0

This chapter describes the new features in Release 4.0 of MarkLogic Server. The feature descriptions are divided into the following categories:

- [XQuery 1.0 Support](#)
- [Geospatial Search and Analytics](#)
- [Alerting Infrastructure](#)
- [Advanced Analytics](#)
- [Entity Enrichment](#)
- [Positions on Attribute Values](#)
- [Application Development Features](#)
- [Modular Documents](#)
- [Administrative Enhancements](#)
- [Enhanced Security Features](#)
- [Function Mapping](#)

3.1 XQuery 1.0 Support

MarkLogic Server has always included a robust implementation of the W3C XQuery standard, and MarkLogic Server 4.0 enhances that by implementing the W3C XQuery 1.0 Recommendation. Because there are some syntax and semantic differences between the XQuery 1.0 recommendation and the XQuery Draft implemented in previous versions of MarkLogic Server, and to ensure robust backwards and forwards compatibility for existing applications, MarkLogic Server 4.0 includes three dialects of XQuery:

- The XQuery 1.0 Recommendation with the MarkLogic enhancements (for example, `try/catch`). This dialect is known as `1.0-ml`.
- The strict version of the XQuery 1.0 Recommendation. This dialect is known as `1.0`.
- The May 2003 XQuery 1.0 Draft version used in MarkLogic Server 3.2. This dialect is known as `0.9-ml`.

MarkLogic Server 4.0 has a robust way to use library modules from one dialect with library or main modules of another dialect. This maximizes the backwards and forwards compatibility, and makes it possible to migrate your applications to the latest XQuery specification one module at a time. It also allows you to use any standard-compliant XQuery 1.0 code in MarkLogic Server.

For details on these three XQuery dialects and on the XQuery 1.0 support, see the *XQuery Reference Guide*.

3.2 Geospatial Search and Analytics

MarkLogic Server 4.0 provides support for geospatial search, which allows you to search based on location-based data. You can perform these geospatial searches based on a point, a box specified by four points, or a polygon. The geospatial search and analytics capabilities have the following components:

- New `cts:query` constructors for geospatial queries (for example, `cts:element-geospatial-query`).
- A set of geospatial APIs to support geospatial queries (`cts:box`, `cts:point`, and so on).
- The ability to define regions using geospatial polygons, which can have any number of vertices.
- A set of XQuery convenience libraries to support various geospatial data markup formats (GML, KML, GeoRSS/Simple, or Metacarta).
- New geospatial indexes, which are similar to range indexes, but operate on geospatial data (for example, points).
- New geospatial lexicons which operate on the geospatial indexes (for more details, see “Geospatial Lexicons” on page 12).

These components allow you to build robust and scalable applications that search and analyze content marked up with geospatial data.

Note: Using the geospatial query constructors requires a valid geospatial license key; without a valid license key, searches that include geospatial queries will throw an exception.

3.3 Alerting Infrastructure

MarkLogic Server 4.0 provides the APIs needed to build complex, high-performance alerting applications. An alerting application has functionality which is the reverse of a typical search application. Search applications find nodes that match a query; alerting applications store queries and are efficient at finding the stored queries that a node would match. To create these applications, there are several new components in MarkLogic Server:

- An Alerting API, which is a high-level API designed to create all aspects of an alerting application, including ways to handle security, ways to run alerting requests on new and changed documents, and actually performing the alerting searches.
- The ability to create XML representations of `cts:query` constructors (and therefore the ability to store the serialized `cts:query` constructors in a MarkLogic Server database).
- The `cts:reverse-query` API which finds queries that a set of nodes would match.
- A new index option for `cts:reverse-query` operations. The indexes are not required to use the `cts:reverse-query` API, but make it perform and scale better.

Note: Alerting applications require a valid alerting license key. The license key is required to use the reverse index and to use the Alerting API.

3.4 Advanced Analytics

MarkLogic Server 4.0 adds several analytic capabilities to complement the other analytic features in the product (for example, lexicon frequency analysis):

- [Value Lexicons For Elements and Attributes of All Datatypes](#)
- [Sample=N Option for Lexicon APIs](#)
- [Element and Attribute Co-Occurrence Lexicons](#)
- [Range Lexicons](#)
- [Geospatial Lexicons](#)

3.4.1 Value Lexicons For Elements and Attributes of All Datatypes

MarkLogic Server 4.0 expands the lexicon APIs to allow `cts:element-values` and the other lexicon API calls on elements of any type, as long as there is a range index on that element or attribute. For example, if you have an `xs:date` range index on a date element, you can now use the lexicon APIs to return all of the unique dates (and use `cts:frequency` to find the counts of how many times each date occurs). Previously the value lexicon APIs only operated on `xs:string` elements and attributes.

3.4.2 Sample=N Option for Lexicon APIs

There is now a `sample=N` option for each lexicon API, which allows you to return matches from the first N fragments while performing the analytics on matches from the entire data set. The new `sample` option is complementary to the `truncate=N` option (which existed in 3.2 as well) and it limits the number of fragments from which matches are returned as well as limits the fragments from which the analytics are calculated. For details, see the *Mark Logic Built-In and Module Functions Reference*.

3.4.3 Element and Attribute Co-Occurrence Lexicons

With MarkLogic Server 4.0, you can now find pairs of values that appear together. To support this, element and attribute range indexes now have range value positions which, when enabled, allow you to use the following APIs:

- `cts:element-attribute-value-co-occurrences`
- `cts:element-value-co-occurrences`

There are options allowing you to specify the proximity (how many words apart the values are), the order, and other factors. You can also use the accessor `cts:frequency` on the co-occurring pairs to see how many times they occur in a fragment.

3.4.4 Range Lexicons

MarkLogic Server 4.0 includes new lexicon support for ranges of values. These APIs return ranges of values divided into buckets. The buckets are determined by bounds passed into the following functions:

- `cts:element-attribute-value-ranges`
- `cts:element-value-ranges`

3.4.5 Geospatial Lexicons

MarkLogic Server 4.0 includes lexicons on geospatial values. These lexicons return geospatial values and values of boxes. There are also geospatial co-occurrence lexicon functions, which return XML elements containing pairs of geospatial values. You can use `cts:frequency` on these geospatial lexicon functions to provide counts of the values returned. The following are the geospatial lexicon APIs:

- `cts:element-attribute-pair-geospatial-boxes`
- `cts:element-attribute-pair-geospatial-value-match`
- `cts:element-attribute-pair-geospatial-values`
- `cts:element-attribute-value-geospatial-co-occurrences`
- `cts:element-child-geospatial-boxes`
- `cts:element-child-geospatial-value-match`
- `cts:element-child-geospatial-values`
- `cts:element-geospatial-boxes`
- `cts:element-geospatial-value-match`
- `cts:element-geospatial-values`
- `cts:element-pair-geospatial-boxes`
- `cts:element-pair-geospatial-value-match`
- `cts:element-pair-geospatial-values`
- `cts:element-value-geospatial-co-occurrences`
- `cts:geospatial-co-occurrences`

3.5 Entity Enrichment

MarkLogic Server 4.0 includes support for enriching the XML of a document based on an analysis of the content. It includes technology to identify `entities` in text and then wrap those entities with meaningful tags. Some typical entities are people and places. With XML that includes entity enrichment, you can perform fine-grained search and analytics, such as finding all of the occurrences of a word near an entity (for example, all of the documents that have the word “revolution” near the person “George Washington”). Entities are more than simple word searches, as they take into account the way a term is used. Entity enrichment includes the `cts:entity-highlight` builtin API and the entity XQuery library module.

Note: A valid entity enrichment license key is required to use `cts:entity-highlight`; without a valid license key, it throws an exception. If you have a valid license for entity enrichment, you can entity enrich text in English and in any other languages for which you have a valid license key. For languages in which you do not have a valid license key, `cts:entity-highlight` finds no entities for text in that language.

3.6 Positions on Attribute Values

MarkLogic Server 4.0 stores position information on attribute values in the indexes. Because of this, `cts:near-query` expressions can now find positions of values in attributes using the range indexes.

3.7 Application Development Features

MarkLogic Server 4.0 includes the following features to support various aspects of application development:

- [Maps in XQuery](#)
- [Evaluate XQuery Expressions With `xdmp:value`](#)
- [SGML Entity Output Capability](#)
- [Choose Output Encoding](#)
- [Custom Error Pages For HTTP Servers](#)
- [Support for Multi-Part Mime Types \(for File Upload Applications\)](#)
- [`cts:element-query` Scope Now Includes Child Attributes](#)
- [`xdmp:get-request-body` Now Allows Binary, XML, or Text Content](#)
- [More Built-In XQuery Functions](#)

3.7.1 Maps in XQuery

MarkLogic Server 4.0 adds a set of XQuery built-ins to support name-value maps. Maps are in-memory structures that hold a set of name-value pairs, and it is a convenient way to store a large number of these pairs in a program. Maps are very useful for certain programming tasks, and they provide a convenient way to store and update temporary data in memory. In some programming languages, maps are implemented using hash tables. You can also take a map and store it in a database, which creates a serialized XML version of the map. For details on maps, see the map functions (`map:clear`, `map:count`, `map:delete`, `map:get`, `map:keys`, `map:map`, and `map:put`) in the *Mark Logic Built-In and Module Functions Reference* and the chapter on using the map functions in the *Developer's Guide*.

3.7.2 Evaluate XQuery Expressions With `xdmp:value`

The new XQuery built-in `xdmp:value` allows you to evaluate an expression while keeping all of the context from the query in which it is called. It is similar to `xdmp:eval`, but it retains the context (for example, the namespace bindings, module imports, and variable definitions) from the calling XQuery statement. It makes it more convenient to construct expressions as strings and then evaluate them to get their results.

3.7.3 SGML Entity Output Capability

MarkLogic Server 4.0 introduces the ability to output SGML entities instead of outputting their codepoints. You can set up an App Server to always output characters that have a corresponding SGML entity as their entities, or you can specify at an individual query level (overriding whatever is set in the App Server configuration) by using the `<output-sgml-character-entities>` option to `xdmp:save` or `xdmp:quote`. For details, see the “Controlling App Server Output and Errors” chapter of the *Developer's Guide*.

3.7.4 Choose Output Encoding

You can now specify the output encoding in which MarkLogic Server serializes query output. You can specify a default output encoding at the App Server level, and you can specify the output at an individual query level (overriding whatever is set in the App Server configuration) with the XQuery built-in function `xdmp:set-response-encoding`. Previously, in MarkLogic Server 3.2, the output encoding was always UTF-8. MarkLogic Server 4.0 still stores everything encoded as UTF-8, but now allows you to output results in other encodings.

3.7.5 Custom Error Pages For HTTP Servers

With MarkLogic Server 4.0, you can now specify an XQuery module for use as a custom error page on an HTTP App Server. You can catch 400 and 500 series HTTP errors and pass them to an XQuery module. The XQuery module can evaluate arbitrary XQuery, providing a large amount of flexibility for how you display errors.

3.7.6 Support for Multi-Part Mime Types (for File Upload Applications)

MarkLogic Server 4.0 now includes support in the HTTP App Server for multi-part mime types. Multi-part mime types are typically used to provide file-upload functionality to web applications. Such applications, supported by most modern browsers, provide an interface to navigate a local file system, select a file to upload, and then upload it to the server. Use the following APIs to use this functionality in an application:

- `xdmp:get-request-field`
- `xdmp:get-request-field-content-type`
- `xdmp:get-request-field-filename`

3.7.7 `cts:element-query` Scope Now Includes Child Attributes

You can now constrain a `cts:element-query` on attributes that are direct children of the element specified in the `cts:element-query`. Previously, the scope of a `cts:element-query` did not include attributes that are direct children of the specified element, only attributes on child elements. The following is an example of this new behavior:

```
(: returns true in 4.0, false in 3.2 :)
let $x := <a attr="something">hello</a>
return
cts:contains($x, cts:element-query(xs:QName("a"),
  cts:and-query((
    cts:element-attribute-word-query(xs:QName("a"),
      xs:QName("attr"), "something"),
    cts:word-query("hello") )) ) )
```

This feature might also introduce an incompatibility from your 3.2 applications, as described in “`cts:element-query` Now Includes Attributes of Element in its Scope” on page 29.

3.7.8 `xdmp:get-request-body` Now Allows Binary, XML, or Text Content

In 4.0, there is a new optional parameter to the `xdmp:get-request-body` function. The option allows you to get binary, text, or XML content from the API. This also introduces a minor incompatibility, as described in “`xdmp:get-request-body` Now Returns an `item()*`” on page 32. See the *Mark Logic Built-In and Module Functions Reference* for more details.

3.7.9 More Built-In XQuery Functions

There are many more built-in XQuery functions to perform a variety of tasks, including the following functions:

- `xdmp:filesystem-directory` Returns an XML node containing the file properties under a directory (for example, the filenames, directory names, and so on).
- `xdmp:md5` Calculates the md5 hash of the input parameter.
- `xdmp:unpath` Returns the node given a simple XPath expression of the form outputted from `xdmp:path`.

See the *Mark Logic Built-In and Module Functions Reference* for descriptions and signatures of all of the XQuery functions.

3.8 Modular Documents

MarkLogic Server 4.0 provides XQuery libraries and CPF pipelines to expand documents that include other documents with XInclude elements. Such *modular documents* can be used to import other documents by reference. The CPF pipelines and supporting XQuery modules will expand these modular documents into their full form, and will keep the expanded versions up-to-date as the original documents and/or their included parts are updated.

3.9 Administrative Enhancements

MarkLogic Server 4.0 includes the following administrative enhancements:

- [Scriptable Admin API](#)
- [Non-Insertable \(Read and Delete Only\) Forests](#)
- [Forest-Level Failover For High Availability](#)
- [Disable/Enable Databases, Forests, and App Servers](#)
- [Improvements to the Admin Interface](#)
- [Scheduled Backups](#)
- [XQuery Built-Ins for Admin Operations \(Including Backup and Restore\)](#)

3.9.1 Scriptable Admin API

MarkLogic Server 4.0 provides an API that allows you to perform many administrative tasks such as creating or modifying a database, forest, or App Server. The API is extensive, with over 400 functions, and it is written in XQuery so you can write XQuery modules to script your administrative tasks. In 3.2, most of these tasks required using the Admin Interface. You can still use the Admin Interface, but you can now also perform the tasks programmatically for greater flexibility.

Because you can script arbitrarily complex configurations, OEMs can use the Admin API to script a configurable installation, and developers can create scripts to easily recreate a configuration. Also, there are some convenience functions to do things like create a new database with the same configuration as an existing database.

For details on the Admin API, see the “Scripting Administrative Tasks With the Admin API” chapter in the *Administrator’s Guide* and the Admin Library section of the *Mark Logic Built-In and Module Functions Reference*.

3.9.2 Non-Insertable (Read and Delete Only) Forests

MarkLogic Server 4.0 provides an option on a forest configuration to disallow inserts and updates, allowing only reads and deletes for that forest. This option is useful for stopping updates to a particular forest, and can also be an effective way to throttle back the resource load on a system performing large-scale loads with multiple forests, without the need to change the loading application so it does not load into a particular forest. For example, if you have four forests on a host and you make two of them non-insertable, you can lower the amount of free disk space needed for update operations.

3.9.3 Forest-Level Failover For High Availability

MarkLogic Server 4.0 includes forest-level failover, which allows you to specify another host to take over a forest in the event that the host is disconnected from the cluster (due to hardware failure, software failure, or power failure, for example). For details on forest-level failover and on other high-availability features in MarkLogic Server, see *Scalability, Availability, and Forest-Level Failover*.

3.9.4 Disable/Enable Databases, Forests, and App Servers

In MarkLogic Server 4.0, each database, forest, and App Server configuration now has a setting to disable or enable it. When a database, forest, or App Server is disabled, no queries can be issued against it. Disabling a configuration is useful when you need to temporarily stop use of that database, forest, or App Server, but you do not want to delete the configuration.

3.9.5 Improvements to the Admin Interface

There are many other improvements to the Admin Interface in MarkLogic Server 4.0, some of which are:

- The ability to add as many items as you want on various pages (for example, fragment roots and parents, range indexes, and so on). Previously, you could only add three items at a time.
- The ability to attach and detach forests in any combination at the same time from the same Admin Interface page. Previously, you had to attach and detach forests from a database as separate steps.

3.9.6 Scheduled Backups

The Admin Interface now has a mechanism for scheduling backup operations, so you can automatically back up databases and/or forests at given time intervals.

3.9.7 XQuery Built-Ins for Admin Operations (Including Backup and Restore)

There are now XQuery Built-In functions for many more administrative functions, including `xdmp:database-backup`, `xdmp:database-restore`, `xdmp:forest-backup`, `xdmp:forest-restore`, `xdmp:shutdown`, and `xdmp:restart`. With these functions and the Admin API (see “Scriptable Admin API” on page 16), you can automate most administrative tasks using XQuery.

3.10 Enhanced Security Features

MarkLogic Server 4.0 includes many features to enhance security and security reporting. This section describes the following security features:

- [Auditing](#)
- [Request Blackouts](#)
- [Show and Change Permissions in the Admin Interface](#)
- [New Privilege for `xdmp:add-response-header`](#)

3.10.1 Auditing

In 4.0, you can now set up a MarkLogic Server cluster to audit events to an audit log. The setup is done at the group level via the Admin Interface. You can set up auditing of various system activities, and the events are logged to the `<marklogic-data-directory>/Logs/AuditLog.txt` file.

3.10.2 Request Blackouts

In 4.0, you can now configure an App Server to have blackout periods where requests will not be allowed. You can set up blackouts for specific users and/or specific roles, and you can configure the blackouts as either recurring or one time events. Each App Server in the Admin Interface now has a Request Blackouts page to configure these blackouts.

3.10.3 Show and Change Permissions in the Admin Interface

In 4.0, each database now has a page for displaying and changing permissions on documents in the database.

3.10.4 New Privilege for `xdmp:add-response-header`

To protect against people adding potentially malicious HTTP headers to a request, the `xdmp:add-response-header` function now has a privilege associated with it. This feature also introduces an incompatibility from 3.2 (see “Functions that Now Require Execute Privileges” on page 30).

3.11 Function Mapping

Function mapping is an extension to XQuery that allows you to pass a sequence to a function parameter that is typed to take a singleton item, and it will invoke that function once for each item in the sequence. Function mapping is enabled by default in 1.0-m1, and can be enabled with a prolog declaration in 1.0 (or disabled with a prolog declaration in 1.0-m1). Function mapping is not available in 0.9-m1. For details about function mapping, see the chapter on “MarkLogic Server Enhanced XQuery Language” in the *XQuery Reference Guide*. Function mapping can also introduce an incompatibility with 3.2 code, as described in “Function Mapping” on page 23.

4.0 Known Incompatibilities with Previous Releases

The vast majority of applications implemented on MarkLogic Server 3.2-* will run either without modifications or with very minor modifications in Release 4.0. There are, however, a number of changes that will cause compatibility issues with 3.2 applications. This section describes those incompatibilities and includes the following topics:

- [XQuery Compatibility](#)
- [Functional Behavior Changes](#)
- [Upgrade and Reindex Required](#)

4.1 XQuery Compatibility

MarkLogic Server 4.0 supports three dialects of XQuery. You can configure an App Server to use one of the dialects by default, and when you upgrade from 3.2, any existing App Servers will set the default to 0.9-m1, which is compatible with 3.2. You can also set the XQuery version in your XQuery modules, and that version is used no matter what the value of the XQuery version setting on the App Server. This section describes some of the XQuery compatibility issues, and includes the following topics:

- [XQuery Version Declaration](#)
- [0.9-m1 Syntax Remains the Same as 3.2](#)
- [New Prolog Syntax for 1.0-m1 and 1.0](#)
- [Triggers Now Default to 1.0-m1, So Trigger Modules Must Be Modified](#)
- [Effective Boolean Value Changes](#)
- [Function Mapping](#)
- [New Privilege on xdmp:add-response-header](#)
- [element\(\) Test in 0.9-m1 Equivalent to schema-element\(\) Test in 1.0-m1](#)
- [Namespace Declaration Attributes \(xmlns\) in Element Constructors Must Now be Literal Values](#)
- [MarkLogic Server err Prefix Bound to a Different Namespace in 1.0-m1](#)
- [XML Serializaton Change for Default Element Namespace With Prefix](#)
- [Can No Longer Bind No Namespace to a Prefix](#)
- [Ampersand Character \(&\) Must Be Escaped in String Literals in 1.0 and 1.0-m1](#)
- [p:state-transition and p:status-transition Now Have Priority Parameter](#)
- [Less-Than Character \(<\) No Longer Allowed in Attribute Values](#)

For more details on XQuery 1.0 support in MarkLogic Server 4.0, see the *XQuery Reference Guide*.

4.1.1 XQuery Version Declaration

The first line of an XQuery main or library module can optionally have an XQuery version declaration. In MarkLogic Server 4.0, you can set the XQuery version to one of the three dialects, and the query is then evaluated in that dialect.

The version declaration is optional, but it is a good practice to use it, as it disambiguates the dialect in which your code is written. Code from MarkLogic Server 3.2 is in the `0.9-ml` dialect. The following table shows the valid XQuery version declarations along with a description of each:

XQuery Version Declaration	Description
<code>xquery version "1.0-ml";</code>	Uses the XQuery 1.0 recommendation with MarkLogic Server enhancements such as <code>try/catch</code> . This is the recommended setting for the version declaration, as it allows you to most easily use features specific to MarkLogic Server.
<code>xquery version "0.9-ml"</code>	Uses the May 2003 XQuery draft version used in MarkLogic Server 3.2. To ensure your old 3.2 code will run on all configurations (regardless of the App Server XQuery version setting), put this declaration on the first line of your 3.2 XQuery modules.
<code>xquery version "1.0";</code>	Uses the strict version of the XQuery 1.0 Recommendation. This version does not include Mark Logic-specific enhancements to XQuery, but you can still use the MarkLogic Server built-in functions as long as you import the namespace bindings. In general, the strict version is only required if you need your code to be compatible with other XQuery processors, and any of the Mark Logic-specific built-in functions (<code>cts:search</code> , for example) are not available on other XQuery processors.

4.1.2 0.9-ml Syntax Remains the Same as 3.2

You can still run XQuery code exactly as it was written in 3.2. If the default XQuery version setting on your App Server configuration is set to `0.9-ml`, and if your XQuery modules have no version declaration, then your queries will run in 3.2 compatibility mode, just like in 3.2.

Note: If your XQuery code has a version declaration, it must be set to one of the three values shown above. Also, if you have previously set it to `1.0`, then you will need to change it to `0.9-ml`, or else MarkLogic Server will treat the code as strict XQuery 1.0.

4.1.3 New Prolog Syntax for 1.0-ml and 1.0

Although most of the syntax changes are minor between the May 2003 XQuery draft and the January 2007 XQuery recommendation, the changes in syntax in the prolog will cause syntax errors if you try and run the May 2003 code (0.9-ml) in a January 2007 XQuery parser (1.0-ml or 1.0). The main changes are:

- Prolog statements end in a semicolon (;).
- Statements that previously used `declare now` use `define` (except for `declare namespace`, which still uses `declare`).
- The `declare variable` statement now uses `:=` instead of curly braces.

The following table lists examples of syntax changes in the prolog:

xquery version "0.9-ml"	xquery version "1.0-ml" (and "1.0")
<code>import namespace foo="foo"</code>	<code>import namespace foo="foo";</code>
<code>declare namespace foo="foo"</code>	<code>declare namespace foo="foo";</code>
<code>define variable \$foo {"foo"}</code>	<code>declare variable \$foo := "foo";</code>
<code>define function foo() {"foo"}</code>	<code>declare function local:foo() {"foo"};</code>

For more details about XQuery 1.0, see the *XQuery Reference Guide*.

4.1.4 Triggers Now Default to 1.0-ml, So Trigger Modules Must Be Modified

For triggers that spawn actions on the task server, the default XQuery version is now 1.0-ml. If you have any triggers you created in 3.2, the trigger action modules will now run as 1.0-ml, which means that in most cases, they will fail with a syntax error. To fix this, add the 0.9-ml declaration as the first line of the action modules as follows:

```
xquery version "0.9-ml"
```

The version declaration will make the code run as it did in 3.2. You can also migrate the code in your modules to 1.0-ml if you desire. For details on migrating your XQuery code, see [Strategies For Migrating Code to Enhanced Dialect](#) in the *XQuery Reference Guide*.

4.1.5 Effective Boolean Value Changes

There are some changes to effective boolean value rules in the January 2007 XQuery recommendation that might cause some changes in the behavior of your 3.2 applications if you migrate them to 1.0-ml or to 1.0 (they will run the same if you leave them as 0.9-ml). The behavior of effective boolean values on sequences is different; for example, two `false` values have an effective boolean value of `true` in 0.9-ml, have a value of `false` in 1.0-ml, and is an error in 1.0.

For example, in 3.2 (0.9-m1):

```
xquery version "0.9-m1"

fn:boolean((fn:false(), fn:false()))

(: returns true :)
```

For example, in 1.0-m1:

```
xquery version "1.0-m1";

fn:boolean((fn:false(), fn:false()))

(: returns false :)
```

For example, in 1.0:

```
xquery version "1.0";

fn:boolean((fn:false(), fn:false()))

(: returns:
  XDMP-EFFBOOLVALUE: (err:FORG0006) boolean((false(), false())) --
  Effective Boolean Value is undefined for (false(), false())
:)
```

If you have applications that relied on this behavior, you will have to rewrite them to follow the new effective boolean value rules if you are using 1.0-m1 or 1.0. For more details about effective boolean value rules, see the XQuery 1.0 Recommendation (<http://www.w3.org/TR/xquery/#id-ebv>).

4.1.6 Function Mapping

The 1.0-m1 XQuery dialect includes function mapping, which is an extension to XQuery that allows you to pass a sequence to a function parameter that is typed to take an item, and it will run that function once for each item in the sequence. If you migrate code to 1.0-m1, function mapping can cause some functions that are typed as taking a single item to run multiple times, where the code in 0.9-m1 would have thrown an exception. Function mapping can be disabled in 1.0-m1 with a prolog declaration, and it can be enabled in 1.0 with a prolog declaration; function mapping is not available in 0.9-m1. For details on function mapping, see the *XQuery Reference Guide*.

4.1.7 New Privilege on `xdmp:add-response-header`

To enhance security, there is now a privilege on the `xdmp:add-response-header` function. If you have code that uses this function, you need to either add this privilege to a role assigned to users who run the code or create and use an amp to run this code. If you do not add the privilege to a role or add an amp, then code from 3.2 that uses this function will throw an exception in 4.0 when run by unauthorized users.

4.1.8 `element()` Test in 0.9-ml Equivalent to `schema-element()` Test in 1.0-ml

In the 1.0 and 1.0-ml XQuery dialects, the `element()` test only matches elements with the same name. In 0.9-ml (and in 3.2), it also matches elements that have names of substitution elements. To have the same behavior of the 0.9-ml `element()` test in 1.0 or 1.0-ml, use the `schema-element()` test. For example, in 3.2, the following returns `true` if the element `bar` is a substitution element of `foo`, but returns `false` in 4.0 using the 1.0 and 1.0-ml dialects:

```
element(foo) eq <bar/>
```

In 1.0 or 1.0-ml, the following returns `true` if the element `bar` is a subtype of `foo`:

```
schema-element(foo) eq <bar/>
```

For your 1.0 and 1.0-ml code to behave the same way as your 0.9-ml code, change any `element()` tests to `schema-element()` tests.

4.1.9 Namespace Declaration Attributes (`xmlns`) in Element Constructors Must Now be Literal Values

In 4.0, any namespace declaration attribute (`xmlns`, for example) within an element constructor (direct or computed) must now use a literal value for its value; you cannot compute it in a constructor. For example, the following will throw an exception in 4.0 (all XQuery dialects):

```
let $x := "my.namespace" return <a xmlns={$x}/>
```

In 4.0, the above throws `XDMP-NSLITERAL`. If you want to construct the value, you can do so using a computed element constructor in the specified namespace. For example:

```
let $x := "my.namespace" return element {fn:QName($x,"a")} {}
```

If you have any code that does not use literal values for `xmlns` namespace declarations, you need to either rewrite it to use computed constructors or replace the computed part in the direct constructor with a literal value.

4.1.10 MarkLogic Server `err` Prefix Bound to a Different Namespace in 1.0-ml

The `err` prefix is bound to the MarkLogic Server error namespace (<http://marklogic.com/xdmp/error>) in 0.9-ml, and is bound to the namespace for XQuery and XPath errors (<http://www.w3.org/2005/xqt-errors>) in 1.0-ml.

The `error` prefix is bound to the MarkLogic Server error namespace (<http://marklogic.com/xdmp/error>) in both 1.0-m1 and 0.9-m1. For a list of predefined namespaces for each XQuery dialect, see the [Understanding XML Namespaces in XQuery](#) chapter in the *Developer's Guide*.

If you have code from 3.2 that used the `err` prefix for the MarkLogic Server error namespace (used for the XML returned when an exception is thrown), you should change it so it uses the `error` prefix (which is bound to the same namespace). The following demonstrates the namespace bindings for the `err` and `error` prefixes in 1.0-m1 and in 0.9-m1:

```
xquery version "1.0-m1";
<err:error/>,
<error:error/>
(: returns:
<err:error xmlns:err="http://www.w3.org/2005/xqt-errors"/>
<error:error xmlns:error="http://marklogic.com/xdmp/error"/>
:.)

xquery version "0.9-m1"
<err:error/>,
<error:error/>
(: returns:
<error:error xmlns:error="http://marklogic.com/xdmp/error"/>
<error:error xmlns:error="http://marklogic.com/xdmp/error"/>
:.)
```

4.1.11 XML Serializaton Change for Default Element Namespace With Prefix

In 4.0 in all XQuery dialects, the way XML serializes in queries that both declare a default element namespace and bind that namespace to a prefix is slightly different than in 3.2. In 4.0, the prefix is not included in the serialization, and in 3.2 it is. For example:

```
xquery version "0.9-m1"
default element namespace ="my-namespace"
declare namespace my="my-namespace"

<some-element/>

(: returns:
   In 3.2: <my:some-element xmlns:my="my-namespace"/>
   In 4.0: <some-element xmlns="my-namespace"/>
:)
```

Note that these two serializations are equivalent in the XML data model, they are just changes in the textual representation of the XML. For example. the following returns true:

```
xquery version "0.9-m1"
default element namespace ="my-namespace"
declare namespace my="my-namespace"

<some-element/> eq <my:some-element/>

(: returns true :)
```

4.1.12 Can No Longer Bind No Namespace to a Prefix

In 4.0, you can no longer bind a namespace prefix to the empty namespace, sometimes called “no namespace.” In fact, the prolog declaration for this has a different meaning in 4.0; it now un-binds that prefix from a namespace. For example, consider the following:

```
declare namespace no = ""
```

In 3.2, this would bind the prefix `no` to the empty namespace (no namespace). In 4.0, it ensures that the prefix `no` is not bound to any namespace; that is, it *un-binds* the prefix. This is true in all XQuery dialects: 0.9-m1, 1.0-m1, and 1.0.

If you have any code that uses this design pattern, you have to rewrite it. If you are using direct element constructors to construct an element in no namespace with a namespace prefix (`<no:hello/>`, for example), you can either take the prefix off if the default element namespace is no namespace or use a computed element constructor and use a function like `fn:QName` to specify the namespace of the element.

4.1.13 Ampersand Character (&) Must Be Escaped in String Literals in 1.0 and 1.0-m1

In 4.0 using the 1.0 and 1.0-m1 XQuery dialects, you can no longer use a literal ampersand (&) character in a string literal. This is not legal XQuery, and it will now throw an `XDMP-ENTITYREF` exception. You can express an ampersand character either with the XML entity (`&`), in a CDATA element, or you can “repair” it using the repair option to `xdmp:document-load`, `xdmp:document-get`, or `xdmp:unquote`. If you have code that uses the ampersand character in string literal expressions, you must now use the XML entity or one of the other ways in 1.0 and 1.0-m1. In 0.9-m1, you can still use a literal ampersand character in a string literal.

4.1.14 p:state-transition and p:status-transition Now Have Priority Parameter

The signatures for the CPF pipeline functions `p:state-transition` and `p:status-transition` have changed, and they now have a new parameter to pass in for the priority of the transition. Higher priorities are executed before lower priorities when pipelines act on the same state or status. If you have any code that uses these functions, you will need to add the `$priority` parameter to those function calls. For the signatures, see the *Mark Logic Built-In and Module Functions Reference*.

4.1.15 Less-Than Character (<) No Longer Allowed in Attribute Values

The less-than character (<) is no longer allowed in attribute values. This is true in all XQuery dialects (0.9-m1, 1.0-m1, and 1.0). In 3.2, this character was allowed. If you have any attribute values that contain less-than characters, you must change them to use the XML entity for a less-than character (`<`). For example, the following throws an `XDMP-ATTRVALCHAR` exception in 4.0, but worked in 3.2:

```
<a b="<" />
```

To correct this, change it to the following:

```
<a b="&lt;" />
```

Ampersand characters (&) are also illegal in attribute values in 1.0 and 1.0-m1 (except in as entity escape characters). You must use the XML entity for an ampersand instead (`&`). Ampersand characters are allowed in 0.9-m1, although you should change those to use the XML entites as well for forward compatibility.

4.2 Functional Behavior Changes

This section lists functional behavior changes that might cause results to be different in 4.0 than in 3.2. In some cases, the differences are small, but if your application depends on the old behavior than you might need to make code changes in your application to keep the same behavior.

- [Ordered `cts:near-query` With Distance of 0 Now Matches Itself](#)
- [`cts:element-query` Now Includes Attributes of Element in its Scope](#)
- [Lexicon Functions Now Default to Document Values, Was Any Value](#)
- [Functions that Now Require Execute Privileges](#)
- [Maximum Length of `xs:float` and `xs:double` Casts](#)
- [`xdmp:describe` Differences Due to the `xs` Namespace Prefix Replacing the `xdt` Prefix](#)
- [Function Calls Now Check Return Types](#)
- [Can No Longer Load XML Documents With Multiple Root Nodes](#)
- [`xdmp:get-request-body` Now Returns an `item\(\)*`](#)
- [`repair` Option of `xdmp:document-load`, `xdmp:unquote`, and Others Now Default to `none` in 1.0 and 1.0-ml](#)
- [Attribute Whitespace No Longer Normalized](#)
- [HTTP and WebDAV Servers Now Default to Digest Authentication](#)
- [CPF State Changes for Default Conversion Option](#)
- [XCC for Java SPI ConnectionProvider `getConnection` Method Has Different Signature](#)
- [XCC Duration Type References that Used XDT Now Use XS](#)

4.2.1 Ordered `cts:near-query` With Distance of 0 Now Matches Itself

When using `cts:near-query` with the `ordered` option, items will now match themselves when specifying a distance of 0. Previously, `ordered` with a distance of 0 did not match itself. Queries that used the `unordered` option also match themselves with a distance of 0 (they also matched in 3.2). If you have applications that use `cts:near-query` with the `ordered` option, you might find some cases where a query matches in 4.0 but did not match in 3.2.

The following example demonstrates the change in behavior:

```
let $x := <foo>This is a test</foo>
return
cts:contains($x, cts:near-query(("test","test"),0,"ordered"))
(: returns true in 4.0, false in 3.2 :)
```

4.2.2 `cts:element-query` Now Includes Attributes of Element in its Scope

When using `cts:element-query` with `cts:element-attribute-query` as one of the `cts:query` constructors in the second argument, the scope of the attributes now includes attributes on the element specified in the `cts:element-query`. The following example demonstrates this behavior.

```
(: returns true in 4.0, false in 3.2 :)
let $x := <a attr="something">hello</a>
return
cts:contains($x, cts:element-query(xs:QName("a"),
  cts:and-query((
    cts:element-attribute-word-query(xs:QName("a"),
      xs:QName("attr"), "something"),
    cts:word-query("hello") )) ) )
```

While this behavior allows you to constrain on attributes of the element specified in the `cts:element-query`, it is a change in behavior from 3.2. Previously, it only searched attributes of descendant elements. If you have any code that relied on the old behavior, it will now match in cases where it did not before, and you have to either decide if that is acceptable to the application or rewrite the code so it behaves like it did before. This feature is also described in “Positions on Attribute Values” on page 13 and in “`cts:element-query` Scope Now Includes Child Attributes” on page 15.

4.2.3 Lexicon Functions Now Default to Document Values, Was Any Value

In MarkLogic Server 4.0, all of the lexicon functions now default to the `document` option, which constrains the results to values found in document fragments only. In 3.2, the default was `any`, which returned values found in any type of fragment (document, property, or lock). If you have code that relies on the old default behavior, you should rewrite those lexicon calls to explicitly use the `any` option. For example, consider the following lexicon call:

```
cts:element-values(xs:QName("myElement"))
```

In 3.2, this returns all unique values for the specified element, whether they occur in documents, properties, or locks. In 4.0, this returns only the element values that occur in documents. In many cases, it will not be any different (because the element might only exist in documents). If you need it to search through all fragments, you should rewrite this lexicon call as follows:

```
cts:element-values(xs:QName("myElement"), "", "any")
```

4.2.4 Functions that Now Require Execute Privileges

The following functions are now protected by execute privileges. In 3.2, no privileges were required to execute these functions. If you have code that uses any of these functions, any user that runs that code will need to have (via a role) the execute privilege corresponding to the newly protected function. The following functions table lists the newly protected functions and the URI of their corresponding execute privileges:

Function	Execute Privilege
<code>xdmp:get-session-field</code>	<code>http://marklogic.com/xdmp/privileges/xdmp-get-session-field</code>
<code>xdmp:get-session-field-names</code>	<code>http://marklogic.com/xdmp/privileges/xdmp-get-session-field-names</code>
<code>xdmp:set-session-field</code>	<code>http://marklogic.com/xdmp/privileges/xdmp-set-session-field</code>
<code>xdmp:add-response-header</code>	<code>http://marklogic.com/xdmp/privileges/xdmp-add-response-header</code>
<code>xdmp:invoke-modules-change</code>	<code>http://marklogic.com/xdmp/privileges/xdmp-invoke-modules-change</code>
<code>xdmp:invoke-modules-change-file</code>	<code>http://marklogic.com/xdmp/privileges/xdmp-invoke-modules-change-file</code>
<code>xdmp:spawn-modules-change</code>	<code>http://marklogic.com/xdmp/privileges/xdmp-spawn-modules-change</code>
<code>xdmp:spawn-modules-change-file</code>	<code>http://marklogic.com/xdmp/privileges/xdmp-spawn-modules-change-file</code>

Users who run application code containing these functions will need to have the corresponding execute privilege(s). In the case of the new `xdmp:invoke` and `xdmp:spawn` modules and filesystem privileges, those are only needed when changing the modules database and when changing the filesystem root, respectively. You can grant the needed privileges to users who run your application via a role (either in a new role or by modifying an existing role already granted to the user). Another possibility is to use an amp to run these functions as a more privileged user. For detail on security administration, see the *Administrator's Guide* and *Understanding and Using Security*.

4.2.5 Maximum Length of `xs:float` and `xs:double` Casts

The maximum length of the string you can cast to an `xs:float` or `xs:double` type is 127 characters in MarkLogic Server 4.0. In 3.2, the maximum was 511 characters. If you have any casts to these datatypes that will attempt to use a string representation of a number with more than 127 digits, they will fail with an invalid lexical representation of a number (`XDMP-LEXNUM`) exception. If you have code that allows for such large representations of numbers, you will have to modify it (or it might throw an exception if a large number is input).

4.2.6 `xdmp:describe` Differences Due to the `xs` Namespace Prefix Replacing the `xdt` Prefix

In 4.0, the `xdt` namespace prefix is replaced with the `xs` prefix (the XML schema types prefix). In 4.0, the `xdt` prefix is mapped to the `xs` prefix; therefore, anywhere you could use the `xdt` prefix in 3.2, you can now use either the `xdt` or the `xs` prefix. These changes are because the `xdt` prefix is deprecated in the XQuery 1.0 specification.

A side effect of this change is that `xdmp:describe` output of expressions using the `xdt` prefix now will return output that has the `xs` prefix. This is true for all XQuery dialects. For example:

```
xdmp:describe(xdt:yearMonthDuration("P20Y"))
(:
  returns xdt:yearMonthDuration("P20Y") in 3.2
  returns xs:yearMonthDuration("P20Y") in 4.0
:)
```

4.2.7 Function Calls Now Check Return Types

In 3.2, some function calls do not check their return types. For example, if you call a function that has a return type in a `case` statement within a `typeswitch` expression, 3.2 does not check the return type. In 4.0, it does check the return type. This can cause some 3.2 code to fail with an `XDMP-AS (Invalid coercion)` error. If you have any code that has erroneous return types, you will need to fix that code in 4.0 (the fixed code will work in 3.2 as well).

For example, the following code has return types of `element()*`, which does not match a text node (a text node is a node, not an element):

```
xquery version "0.9-m1"
define function passthru($x as node()) as element()*
{ for $z in $x/node() return dispatch($z) }

define function dispatch($x as node()) as element()*
{ typeswitch ($x)
  case text() return $x
  default return <temp>{passthru($x)}</temp> }

let $x := <a>hello</a>
return dispatch($x)
```

The return type `element()*` does not match the text node containing `hello`. Changing the return types of the functions to `node()*` fixes the problem in this example, as follows.

```
define function passthru($x as node()) as node()*
{ for $z in $x/node() return dispatch($z) }

define function dispatch($x as node()) as node()*
{ typeswitch ($x)
  case text() return $x
  default return <temp>{passthru($x)}</temp> }
```

Also, functions that are called within element content do not check the return type in 3.2. For example:

```
xquery version "0.9-m1"
define function how-many() as xs:string
{ fn:count(//p) }

<how-many>{how-many()}</how-many>
```

The above example throws an `xDMP-AS` exception in 4.0, as expected, because the return is an `xs:unsignedLong` number, not an `xs:string` as specified in the function definition. In 3.2, this does not throw an exception. If you have functions with incorrect return types, you must correct them in 4.0.

4.2.8 Can No Longer Load XML Documents With Multiple Root Nodes

In 4.0, you can no longer load XML documents in the database with multiple root nodes. Previously, if you had a document with multiple root nodes (for example, with a text node followed by an element node), it will load in 3.2. In 4.0, this is no longer allowed (this is true in all XQuery dialects: 0.9-m1, 1.0-m1, and 1.0).

If you have documents with a text root node followed by an element root node that you were loading as XML documents in 3.2 (where you only needed the element node) and want to load them in 4.0, you can use the `input-xml` option to `xdmp:tidy` to clean them up in 4.0, as in the following example:

```
xdmp:tidy("foo<bar/>",
  <options xmlns="xdmp:tidy">
    <input-xml>yes</input-xml>
  </options>) [2]
```

If you want to see what tidy cleaned up, you can look at the first node of the tidy output which lists all of the parts that tidy repaired.

4.2.9 xdmp:get-request-body Now Returns an item()*

The `xdmp:get-request-body` function now returns an `item()*`. In 3.2, it returned a string. Typically, the item it returns is a document node, with the contents being the contents of the POST body for the request, either as a text node, as an element node, or as a binary node. If you have code that uses the `xdmp:get-request-body` function, you should add a `/node()` XPath step to it as follows to get the same behavior in 4.0 as you did in 3.2:

```
xdmp:get-request-body()/node()
```

There is also a new optional parameter to the `xdmp:get-request-body` function that allows you to specify binary, text, or XML format for the request body. For more details, see “`xdmp:get-request-body` Now Allows Binary, XML, or Text Content” on page 15.

4.2.10 repair Option of xdm:document-load, xdm:unquote, and Others Now Default to none in 1.0 and 1.0-ml

In the 1.0 and 1.0-ml dialects, the default for XML repairing option is now `none` instead of `full`. If you have code that relied on the old default, you will need to explicitly set repair to `full`, as in the following examples:

```
xquery version "1.0-ml";
xdmp:unquote('<html xmlns="http://www.w3.org/1999/xhtml">
<p>content</html>', "repair-full");

xquery version "1.0-ml";
xdmp:document-load("c:/tmp/myfile.xml",
  <options xmlns="xdmp:document-load">
    <repair>full</repair>
  </options> );
```

In 0.9-ml, the default for repair is still `none`. These changes apply to `xdmp:document-load`, `xdmp:document-get`, `xdmp:unquote`, and `xdmp:zip-get` (also for the deprecated functions `xdmp:load` and `xdmp:get`).

4.2.11 Attribute Whitespace No Longer Normalized

In 4.0 in all XQuery dialects, attribute whitespace is only normalized if the attribute uses a schema type that includes whitespace normalization. In 3.2, all attribute whitespace was normalized. The following illustrates this change:

```
fn:string(<my-element my-att=" hello there ">
  content</my-element>/@*)
(: returns the string " hello there " in 4.0 and
  returns the string "hello there" in 3.2 :)
```

4.2.12 HTTP and WebDAV Servers Now Default to Digest Authentication

When you create a new HTTP or WebDAV Server, the default value for `authentication` is `digest` in 4.0. Previously, the default was `basic`. Digest authentication uses encrypted passwords and requires a client that is capable of encrypting the passwords (most modern browsers and WebDAV clients support digest authentication). If you need to use basic authentication for a new HTTP or WebDAV Server, you will need to set the configuration manually in the Server configuration, either when you create it or after it is created. For HTTP or WebDAV servers upgraded from 3.2 to 4.0, the upgrade will retain whatever setting with which each App Server is configured.

4.2.13 CPF State Changes for Default Conversion Option

In previous releases, the end state for conversion options was `http://marklogic.com/states/final`. In 4.0, there are different ending states depending on which actions are run. This change allows you to add more complex logic to accommodate moving between more states. If you have any code that relies on the `final` state, you might need modify that code to use the new state.

4.2.14 XCC for Java SPI ConnectionProvider obtainConnection Method Has Different Signature

The signature of the `com.marklogic.xcc.spi.ConnectionProvider.obtainConnection` method in XCC for Java is different in 4.0 than it was in 3.2. The `obtainConnection` method now requires a `request` parameter as the second parameter. If you have any code that uses this method, you will need to add the `request` parameter to it and recompile your application before using it with XCC 4.0.

4.2.15 XCC Duration Type References that Used XDT Now Use XS

XCC 4.0 has new types for the duration values. Previously, they referenced XDT value types, and now they reference XS value types. This is true no matter what dialect the XQuery is written in. You can still access 3.2 servers with XCC 4.0, but you must change your Java or .Net code to use the XS value types instead of the XDT value types.

4.3 Upgrade and Reindex Required

When you log into the Admin Interface after installing 4.0, you will be prompted to upgrade the security database and the configuration files. The 4.0 security database and configuration files are not backwards compatible with 3.2. Make sure to do a full backup of your databases before upgrading. If you do not want to reindex after upgrading, turn off reindexing for each database in 3.2 before installing 4.0. For details and index compatibility, see the *Installation Guide*.

Additionally, if you are upgrading from 3.2, it must be from release 3.2-6 or later.

Warning Upgrading from a 4.0 Early Access release to 4.0 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).

Typically, MarkLogic Server requires 3 times the disk space of the source content to be loaded. For example, if you plan on loading 10 GB of content into the datastore, you should reserve 30 GB of disk space for the datastore.

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 10 GB of memory, you should ideally configure the swap space to be 20 GB (and at least 10 GB). This is true on Windows systems also, but the system is normally set up to grow the swap (page) file as needed.

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 4.0, 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>Developer's Guide</i>	Provides procedures, methodologies, and conceptual information about application development tasks.
<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, useful to application developers and administrators.

Documentation	Description
<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>4.0 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 an 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 36. 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 for Internet Explorer 6, Mozilla 1.6, and Firefox.

Authentication functions in MarkLogic Server are supported for Internet Explorer 6 or newer and Mozilla 1.6 or newer (including 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.