

---

# MarkLogic Server

---

## Release Notes

Release 3.2  
Last updated: 3.2-9, October, 2008

## Copyright

© Copyright 2002-2008 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](http://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.

---



---

## Table of Contents

---

### Release Notes

Copyright .....	2
1.0 Introduction .....	7
2.0 Installation and Upgrade .....	8
3.0 New Features in Release 3.2 .....	9
3.1 Language and Collation Support .....	9
3.1.1 Supported Languages .....	10
3.1.2 Collation Support .....	10
3.1.3 Language Support at the Element Level .....	10
3.1.4 Improved Support for Non-UTF-8 Characters .....	10
3.1.5 NFC Normalized Characters for Greater Recall .....	11
3.1.6 Support For Characters Outside the Basic Multilingual Plane .....	12
3.2 Index, Search, and Query Enhancements .....	13
3.2.1 Range Query Constructors .....	13
3.2.2 cts:fitness .....	13
3.2.3 Forest-Specific Search and XPath Operations .....	13
3.2.4 Fast Performance of Frequency Analysis on Element Values .....	14
3.2.5 Field Definitions and Field Query Constructors .....	14
3.2.6 Collection and URI Lexicon APIs .....	14
3.2.7 Tokenization Enhancements .....	14
3.3 Administrative and Operational Enhancements .....	15
3.3.1 Server Administration Built-In XQuery Functions to Provide System Status 15	15
3.3.2 Ability to Set Merge Size Limit in Merge Blackout Periods .....	15
3.3.3 Extended Functionality of xdm:merge .....	16
3.3.4 Forcing Reindexing at a Timestamp .....	16
3.3.5 Built-In xdm:request-cancel XQuery Function .....	16
3.3.6 Option for Cluster-Wide License Key Acceptance .....	16
3.4 Other Features .....	16
3.4.1 Performance Profiler API .....	17
3.4.2 Zip Functions to Support Microsoft Office 2007 Conversions .....	17
3.4.3 More PDF Pipelines for Better Control Over PDF Conversion .....	17
3.5 More Languages Introduced in 3.2-4 and 3.2-5 .....	17
3.6 Improved PDF Conversion in 3.2-7 .....	17
3.7 New Privileges Added in 3.2-9 .....	17

<b>4.0</b>	<b>Known Incompatibilities with Previous Releases</b>	<b>18</b>
4.1	Content With Existing xml:lang Attributes	18
4.2	Unstemmed Searches Now Require Word Search Indexes	19
4.3	Deprecated Options to xdm:unquote Removed and Replaced	19
4.4	Minor Changes to xdm:describe Output	19
4.5	Non-Breaking Spaces Now Treated as Space, Not as Punctuation	20
4.6	Adjacent Punctuation Now Treated as Separate Tokens	20
4.7	Handling of Empty Text and Binary Nodes	21
4.8	Whitespace Now Preserved in xdm:tidy By Default	21
4.9	App Server Default Collation Change	21
4.10	Upgraded Range Indexes Use Codepoint Collation, Can Affect Value Lexicon APIs	22
4.11	Non-ASCII Characters Now Serialize as the UTF-8 Characters	23
4.12	Cannot Load Non-XML Character Entities With Repair Set to NONE	24
4.13	NFC Unicode Normalization	24
4.14	Demo Server on Port 8000 Now Requires Authentication	25
4.15	New Privileges Added in 3.2-9	25
<b>5.0</b>	<b>Changes Made to W3C Functions and Operators in MarkLogic Server 3.1</b>	<b>26</b>
5.1	New Functions	26
5.1.1	Replacements For fn:escape-uri	26
5.1.2	Renamed Functions	27
5.1.3	Other New Functions	28
5.2	New Operators	28
5.3	New Or Expanded Parameters For Functions	29
5.4	Expanded Functionality	30
5.4.1	Function Overloading	30
5.4.2	Computed Element and Attribute Constructor Names Atomized	31
5.4.3	Document Content in Element/Document Constructors	31
5.4.4	Some Functions That Previously Returned xs:string Now Return xs:anyURI and Do Atomic Promotion to xs:string	31
5.5	New Errors	32
<b>6.0</b>	<b>Features Introduced in Release 3.1</b>	<b>33</b>
6.1	Indexing, Search, and Query Enhancements	33
6.1.1	Improved Performance	34
6.1.2	Diacritic-Insensitive Search	34
6.1.3	Trailing Wildcard Indexes	34
6.1.4	Lexicon of Words in the Database	34
6.1.5	Using Lexicons, Trailing Wildcard, and Character Indexes to Resolve Wildcard Queries	35
6.1.6	Constrain cts:query Expressions By Documents, Collections, or Directories	35
6.1.7	Query Constructor API Enhancements	35
6.1.8	Registered cts:queries	35

6.1.9	Alternate Scoring Methods .....	36
6.1.10	Bounded Relevance Values: cts:confidence .....	36
6.1.11	Fast Pagination .....	36
6.1.12	Efficient Estimates When Performing Searches .....	36
6.1.13	Specify a Cap For xdm:estimate and fn:count .....	36
6.1.14	Point-In-Time Queries .....	36
6.1.15	New Built-In Functions and Operators Added for XQuery 1.0 Forward-Compatibility 37	
6.1.16	Select XQuery F&O Built-In Functions Enhanced for XQuery 1.0 Forward-Compatibility 37	
6.1.17	Select Enhancements to the XQuery Evaluator .....	37
6.2	Administrative and Operational Enhancements .....	37
6.2.1	Cache Size Maximum Values Increased to 16GB .....	38
6.2.2	Cancel a Request .....	38
6.2.3	Cancel a Merge .....	38
6.2.4	Merge Policy .....	38
6.2.5	Resilience to Running Out of Disk Space During a Merge .....	38
6.3	Other Features .....	38
6.3.1	XML Classifier API .....	39
6.3.2	Pre-Commit Triggers .....	39
6.3.3	CPF With Pre-Commit Triggers .....	39
6.3.4	Additional Platform Support .....	39
6.4	XCC—New Client-Side Connectors for Java and .NET .....	39
6.4.1	XML Contentbase Connector (XCC) .....	40
6.4.2	XCC Retryable Exceptions .....	40
7.0	Features Introduced in Release 3.0 .....	41
7.1	Features to Support Content Processing .....	41
7.1.1	Content Processing Framework .....	41
7.1.2	Document Conversion Option .....	41
7.1.3	Expanded HTTP Functions .....	41
7.1.4	HTTP-Enabled Load and Get Functions .....	42
7.1.5	HTML Tidy .....	42
7.1.6	xdmp:invoke and xdm:invoke-in .....	42
7.1.7	Post-Commit Triggers .....	42
7.2	Query, Performance, and Indexing Enhancement .....	42
7.2.1	Proximity Queries .....	43
7.2.2	Text Highlighting .....	43
7.2.3	Side Effects With xdm:set .....	43
7.2.4	Reindexing and Refragmenting .....	43
7.2.5	String Range Indexes .....	43
7.2.6	Wildcard Index Enhancements .....	43
7.2.7	Order By Optimization .....	43
7.2.8	Property Axis .....	43
7.2.9	Ordered cts:and-query .....	44
7.2.10	cts:element-query .....	44

7.2.11	Element Word Query Through Rules .....	44
7.2.12	XPath Union Optimization .....	44
7.3	Administrative Enhancements and Other Features .....	44
7.3.1	Status Screens and Other Admin Interface Changes .....	45
7.3.2	Moving Documents Between Forests .....	45
7.3.3	xdmp:subbinary .....	45
7.3.4	Base-64 Encoding/Decoding .....	45
7.3.5	Last Modified Property .....	45
7.3.6	Permissions, Collections, and Quality Inheritance .....	45
7.3.7	XDBC: Multiple Documents in DocInsertStream() .....	45
7.3.8	XDBC API for Microsoft .NET .....	45
7.3.9	Support Request Facility in Admin Interface .....	46
8.0	Other Notes .....	47
8.1	Memory and Disk Space Requirements .....	47
8.2	Compatibility with XQuery Drafts .....	48
8.3	XQuery Extensions .....	48
8.4	Documentation .....	48
8.5	Browser Requirements .....	50
8.6	Support .....	50

## 1.0 Introduction

MarkLogic Server 3.2 is a feature release. The new features are described in “New Features in Release 3.2” on page 9. The following table lists the major features and where they are described:

“Language and Collation Support” on page 9	“Index, Search, and Query Enhancements” on page 13	“Administrative and Operational Enhancements” on page 15	“Other Features” on page 16
<a href="#">Supported Languages</a>	<a href="#">Range Query Constructors</a>	<a href="#">Server Administration Built-In XQuery Functions to Provide System Status</a>	<a href="#">Performance Profiler API</a>
<a href="#">Collation Support</a>	<a href="#">cts:fitness</a>	<a href="#">Ability to Set Merge Size Limit in Merge Blackout Periods</a>	<a href="#">Zip Functions to Support Microsoft Office 2007 Conversions</a>
<a href="#">Language Support at the Element Level</a>	<a href="#">Forest-Specific Search and XPath Operations</a>	<a href="#">Extended Functionality of xdm:merge</a>	<a href="#">More PDF Pipelines for Better Control Over PDF Conversion</a>
<a href="#">Improved Support for Non-UTF-8 Characters</a>	<a href="#">Fast Performance of Frequency Analysis on Element Values</a>	<a href="#">Forcing Reindexing at a Timestamp</a>	
<a href="#">NFC Normalized Characters for Greater Recall</a>	<a href="#">Field Definitions and Field Query Constructors</a>	<a href="#">Built-In xdm:request-cancel XQuery Function</a>	
<a href="#">Support For Characters Outside the Basic Multilingual Plane</a>	<a href="#">Collection and URI Lexicon APIs</a>	<a href="#">Option for Cluster-Wide License Key Acceptance</a>	
	<a href="#">Tokenization Enhancements</a>		

If you are upgrading from 3.1, some applications will require minor changes to run correctly on 3.2. For details, see “Known Incompatibilities with Previous Releases” on page 18.

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 release of MarkLogic Server runs on the following platforms:

- Microsoft Windows XP SP2\*, Microsoft Windows 2003 Server (x86)
- Windows 2003 Server 64-bit Edition (x64)
- Sun Solaris 8, 9, and 10 (64-bit SPARC)
- Sun Solaris 10 (x64)
- Red Hat Enterprise Linux 3.0 and 4.0 (x86)
- Red Hat Enterprise Linux 3.0 and 4.0 (x64)

\* Microsoft Windows XP is supported for development only.

See the *Installation Guide* for details on installing MarkLogic Server.

Upgrading from an Early Access Release of 3.2 is not supported; if you loaded data in an Early Access Release, you must reload it in the production release of 3.2.

Upgrading is supported from 3.1-5 or later to 3.2. If you are running a release prior to 3.1-5, you must first upgrade to 3.1-5 or later before upgrading to 3.2. Additionally, the 3.1 release must be 3.1-5 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 3.1 to 3.2 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 or 3.1 compatibility mode, depending on the version of MarkLogic Server in which they were last loaded or reindexed. Running in compatibility mode will disable certain 3.2 features (for example, frequency analysis on element values) and will 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 3.2 and 3.1. You might need to make some minor code changes to your 3.1 applications before they can run correctly in 3.2. For details on the incompatibilities, see “Known Incompatibilities with Previous Releases” on page 18. For instructions on upgrading to 3.2, including information about database compatibility between 3.1 and 3.2, see the *Installation Guide*.

## 3.0 New Features in Release 3.2

This chapter describes the new features in Release 3.2 of MarkLogic Server. The feature descriptions are divided into the following categories:

- [Language and Collation Support](#)
- [Index, Search, and Query Enhancements](#)
- [Administrative and Operational Enhancements](#)
- [Other Features](#)
- [More Languages Introduced in 3.2-4 and 3.2-5](#)
- [Improved PDF Conversion in 3.2-7](#)
- [New Privileges Added in 3.2-9](#)

### 3.1 Language and Collation Support

The following features provide enhancements to the search and query capabilities in MarkLogic Server:

- [Supported Languages](#)
- [Collation Support](#)
- [Language Support at the Element Level](#)
- [Improved Support for Non-UTF-8 Characters](#)
- [NFC Normalized Characters for Greater Recall](#)
- [Support For Characters Outside the Basic Multilingual Plane](#)

For details on languages, collations, and how languages effect searches in MarkLogic Server, see the [Language Support in MarkLogic Server](#) chapter of the *Developer's Guide*.

### 3.1.1 Supported Languages

MarkLogic Server 3.2 includes support for the following languages:

- English
- French
- Italian
- German
- Spanish
- Russian
- Arabic
- Chinese (Simplified and Traditional)
- Korean
- Persian (Farsi)
- Dutch

### 3.1.2 Collation Support

MarkLogic Server 3.2 supports a wide range of collations for different languages. Collations specify the character order for a character set.

### 3.1.3 Language Support at the Element Level

MarkLogic Server 3.2 allows you to specify languages in content at the element level. An element in one language can contain child elements in another language. Queries in a given language with stemming will only match results in that language. For example, a stemmed search for the French word *chat* (meaning cat) will not find the English word *chat* (meaning to converse lightly).

### 3.1.4 Improved Support for Non-UTF-8 Characters

MarkLogic Server 3.2 allows you to convert non-UTF-8 characters to UTF-8 characters directly from XQuery. There are `encoding` options to many APIs including `xdmp:document-load`, `xdmp:document-get`, and `xdmp:http:get`. Previously, you needed to convert the content to UTF-8 before using it in MarkLogic Server; now you can convert it on the fly while loading or getting the content. MarkLogic Server supports the encodings for character sets commonly used with the [Supported Languages](#) in 3.2. Other encodings are not currently supported.

### 3.1.5 NFC Normalized Characters for Greater Recall

MarkLogic Server 3.2 normalizes characters using Unicode NFC (normalization form combined). The NFC form normalizes characters that have diacritics as a single character, even if it is represented as multiple characters. MarkLogic Server 3.2 treats the multi-character representation and the single-character representation of the same character as equivalent. Therefore, searches for either form of the character will each match. In previous versions of MarkLogic Server, these were not normalized and treated as separate characters, and a search for one would not match the other. For more details on the Unicode normalization forms, see <http://unicode.org/reports/tr15/>.

One of the advantages of this normalization is improved support of diacritic characters, both in diacritic-sensitive and diacritic-insensitive searches. In 3.1, diacritic-sensitivity was only for single-codepoint diacritic characters. The NFC normalized characters in MarkLogic Server 3.2 will therefore improve recall for both diacritic-sensitive and diacritic-insensitive searches.

The following sample (with comments) demonstrates how the NFC normalization works in 3.2. Note that this is a change from 3.1, and some searches that did not match in 3.1 will now match in 3.2. If you have upgraded from 3.1, you must either reindex or reload your data in order to get this behavior for documents in a database.

```
(: NFD and NFC - 3.2 :)
```

```
(:
```

```
In Unicode, many diacritics can be represented in two ways.
1) as a separate "combining" codepoint
2) as a codepoint that represents both the base character and
   the diacritic
```

```
For example, "ç" (small c with cedilla diacritic) can be represented
as:
```

```
1) "&#x063;&#x327;" ("latin small letter c" followed by "combining
   cedilla")
```

```
or
```

```
2) "&#x0e7;" ("latin small letter c with cedilla")
```

```
Similarly, there are 2 possible representations for e-acute,
e-grave, etc.
```

```
One can think of (1) as NFD (Normalized Form D (for Decomposed)) and
(2) as NFC (Normalized Form C (for Composed)).
```

```
In 3.1, a word containing "ç" represented as (1) will *not* match
the same word containing "ç" represented as (2).
```

```
In 3.2, it will!
```

```
Run the following example:
```

```
:)
```

```
(: latin small letter c :)
" &#x063; ",
```

```
(: combining cedilla (may not display) :)
"#{x327};",
(: latin small letter c with cedilla :)
"#{x0e7};",
(: "latin small letter c" followed by "combining cedilla" :)
"#{x063;#{x327};",

(: both content and search term represent ç the way your input
device decided :)
cts:contains( <foo>garçon</foo>, "garçon"),
(: content represents ç composed, search term represents ç composed :)
cts:contains( <foo>{ fn:concat( "gar", " #{x0e7};", "on" ) }</foo>,
fn:concat( "gar", " #{x0e7};", "on")),
(: content represents ç composed, search term represents ç
decomposed :)
cts:contains( <foo>{ fn:concat( "gar", " #{x0e7};", "on" ) }</foo>,
fn:concat( "gar", " #{x063;#{x327};", "on")),
(: content represents ç decomposed, search term represents ç
composed :)
cts:contains( <foo>{ fn:concat( "gar", " #{x063;#{x327};", "on" ) }</foo>,
fn:concat( "gar", " #{x0e7};", "on")),
(: content represents ç decomposed, search term represents ç
decomposed :)
cts:contains( <foo>{ fn:concat( "gar", " #{x063;#{x327};", "on" ) }</foo>,
fn:concat( "gar", " #{x063;#{x327};", "on"))

(:
  In 3.2, all these cts:contains() expressions return true().
  In 3.1, only the first two and the last one return true(), the
  others return false().
:)
```

### 3.1.6 Support For Characters Outside the Basic Multilingual Plane

MarkLogic Server 3.2 supports characters beyond the Basic Multilingual Plane. These are characters with codepoints above 10000 hex (65536). Previously, any of these characters were discarded.

## 3.2 Index, Search, and Query Enhancements

MarkLogic Server 3.2 includes enhancements to searches and queries. This section briefly describes the following enhancements:

- [Range Query Constructors](#)
- [cts:fitness](#)
- [Forest-Specific Search and XPath Operations](#)
- [Fast Performance of Frequency Analysis on Element Values](#)
- [Field Definitions and Field Query Constructors](#)
- [Collection and URI Lexicon APIs](#)
- [Tokenization Enhancements](#)

### 3.2.1 Range Query Constructors

MarkLogic Server 3.2 includes new `cts:query` constructors that allow you to compose queries that explicitly use range indexes that you have defined on elements and attributes. The following XQuery APIs are included in the range query constructors:

- `cts:element-attribute-range-query`
- `cts:element-range-query`
- corresponding accessor functions

For details of these APIs and for their signatures, see the *Mark Logic Built-In and Module Functions Reference*.

### 3.2.2 cts:fitness

A new API, `cts:fitness`, returns a number between 0 and 1 representing how well a search result item matches the `cts:query`. Fitness is a normalized measure of relevance that is based on how well a node matches the query issued, not taking into account the number of documents in which the query term(s) occur. This is an alternate to `cts:score`, which does take into account the number of documents in which the query term(s) occur.

### 3.2.3 Forest-Specific Search and XPath Operations

You can now specify one or more forests in which to constrain a search. You can constrain search operations to a set of forests in `cts:search`, the various lexicon APIs (`cts:words`, `cts:element-words`, the URI and collection APIs, and so on), and in an XPath. To constrain an XPath to a set of forests, you must use `cts:search` with an empty `cts:and-query` and a `quality-weight` of zero as in the following example:

```
cts:search(/some/xpath, cts:and-query( () ), (), 0.0,
          xdmp:forest("myForest") )
```

### 3.2.4 Fast Performance of Frequency Analysis on Element Values

MarkLogic Server 3.2 adds the `cts:frequency` API, which allows you to get the counts of distinct values from elements and attributes. The `cts:frequency` API is used with the value lexicon APIs (for example, `cts:element-values`). You must have Range Indexes defined for the elements or attributes from which you want to get the distinct value counts.

### 3.2.5 Field Definitions and Field Query Constructors

MarkLogic Server 3.2 adds the ability to define named *fields*, which are sets of elements you define that can be queried as a single unit. There are query constructors which allow you to use these field queries anywhere you can specify a `cts:query` expression (in a `cts:search`, for example). You can create complex field definitions that include and/or exclude certain elements, add indexing options specific to a field, and add weighting specifications specific to one or more elements. Fields not only make it easier to specify certain elements in your searches, but also improve the speed of the searches. The following APIs have been added to support fields:

- `cts:field-word-query`
- `cts:field-word-match`
- `cts:field-words`

As well as the accessor functions for `cts:field-word-query`:

- `cts:field-word-query-field-name`
- `cts:field-word-query-options`
- `cts:field-word-query-text`
- `cts:field-word-query-weight`

### 3.2.6 Collection and URI Lexicon APIs

MarkLogic Server 3.2 adds URI and Collection lexicons, which respectively list all of the document URIs and all of the collection URIs in a database. These lexicons are enabled or disabled at the database level. The following APIs have been added to support these new lexicons:

- `cts:collection-match`
- `cts:collections`
- `cts:uri-match`
- `cts:uris`

### 3.2.7 Tokenization Enhancements

In addition to being able to tokenize search text in different languages, MarkLogic Server 3.2 offers several other tokenization enhancements for all languages. Punctuation characters that are adjacent to each other are now each treated as a separate token. Previously, adjacent punctuation were coalesced into a single punctuation token. The result is you can now searches that include punctuation have greater recall. For example, the following now returns true, where previously it would return false:

```
(: returns true in 3.2, false in 3.1 :)  
let $x := <root>hello-!?!goodbye</root>  
return  
cts:contains($x, cts:word-query("hello-"))
```

Additionally, non-breaking spaces (&#160;) are now treated as spaces; previously they were treated as punctuatiou. Again, this will increase recall.

These changes will also cause some minor incompatibilities if you do not reindex any content loaded in pre-3.2 releases. For more details, see “Non-Breaking Spaces Now Treated as Space, Not as Punctuation” on page 20 and “Adjacent Punctuation Now Treated as Separate Tokens” on page 20.

### 3.3 Administrative and Operational Enhancements

The following features enhance the administrative and operational capabilities of MarkLogic Server:

- [Server Administration Built-In XQuery Functions to Provide System Status](#)
- [Ability to Set Merge Size Limit in Merge Blackout Periods](#)
- [Extended Functionality of xdmp:merge](#)
- [Forcing Reindexing at a Timestamp](#)
- [Built-In xdmp:request-cancel XQuery Function](#)
- [Option for Cluster-Wide License Key Acceptance](#)

#### 3.3.1 Server Administration Built-In XQuery Functions to Provide System Status

MarkLogic Server 3.2 includes XQuery functions which return status of various parts of the system. These APIs are used to create many of Status pages in the Admin Interface. These Server Administration XQuery APIs allow you to build custom programs to help you monitor your system.

#### 3.3.2 Ability to Set Merge Size Limit in Merge Blackout Periods

MarkLogic Server 3.2 adds functionality to the merge blackout configuration pages to allow you to specify the maximum size of a merge in during a blackout period. Setting a maximum size in a merge blackout allows merges to occur only if they are smaller than the specified setting, effectively eliminated very large merges while allowing smaller, less obtrusive merges to occur.

### 3.3.3 Extended Functionality of `xdmp:merge`

MarkLogic Server 3.2 adds an options node to the `xdmp:merge` API. With this options node, you can now specify the maximum size for a merge, the timestamp before which to merge, whether to merge even if there is only one stand in a forest, and the forests to merge. These options provide a much finer grain of control for explicit merges issued with the `xdmp:merge` function. Using the `xdmp:merge` function with no argument continues to force a complete merge of the system, regardless of the value of any other settings. For details, see *Mark Logic Built-In and Module Functions Reference*.

### 3.3.4 Forcing Reindexing at a Timestamp

MarkLogic Server 3.2 provides a new ability to force a reindex of all fragments in a database older than a specified timestamp (that is, all fragments with a timestamp less than or equal to the specified timestamp). With a timestamp reindex, only fragments that were modified at or before a specified timestamp are reindexed. This process not only reindexes, but refragments any qualifying fragments. If you specify the current timestamp, this effectively forces a reindex/refragment of every fragment in a database. Note that if you set the `reindex timestamp` in the database configuration, that setting remains active in the configuration until you change it, and therefore database restores that add fragments with timestamps lower than the specified timestamp will automatically be reindexed when they are attached to the database. For details on timestamps in MarkLogic Server, see the chapter on [Point-In-Time Queries](#) in the *Developer's Guide*.

### 3.3.5 Built-In `xdmp:request-cancel` XQuery Function

MarkLogic Server 3.2 adds the ability to cancel a request directly from an XQuery program. The `xdmp:request-cancel` function will attempt to cancel a specified request. For details, see *Mark Logic Built-In and Module Functions Reference*.

### 3.3.6 Option for Cluster-Wide License Key Acceptance

When you install or upgrade a cluster, you now have the option to accept the license key for all nodes in the cluster from a single Admin Interface. Previously, you had to accept the licence for each host from its own Admin Interface.

## 3.4 Other Features

The following are administrative and other miscellaneous new features:

- [Performance Profiler API](#)
- [Zip Functions to Support Microsoft Office 2007 Conversions](#)
- [More PDF Pipelines for Better Control Over PDF Conversion](#)

### 3.4.1 Performance Profiler API

MarkLogic Server 3.2 introduces an XQuery API to help you build tools to profile performance characteristics of MarkLogic Server requests. You can use the profile APIs to quantify how much time it takes to evaluate various portions of a query request against MarkLogic Server.

Additionally, you can gather information about how many times an expression is evaluated, providing important information for query tuning.

### 3.4.2 Zip Functions to Support Microsoft Office 2007 Conversions

MarkLogic Server 3.2 includes the ability to zip and unzip documents directly from XQuery. The zip APIs are `xdmp:zip-create`, `xdmp:zip-get`, and `xdmp:zip-manifest`. Because Microsoft Office 2007 uses a zip format to package up XML content, you can use these APIs to write applications that use Word 2007 content in a MarkLogic Server database.

### 3.4.3 More PDF Pipelines for Better Control Over PDF Conversion

MarkLogic Server 3.2 includes several new PDF pipelines as part of the Conversion option of the Content Processing Framework (CPF). CPF is included with the standard MarkLogic Server license, and Microsoft Office and Adobe PDF Conversion is a licence-key enabled feature. For more information on conversion, including descriptions of each of these PDF pipelines, see the chapter on the [The Default Conversion Option](#) in the *Content Processing Framework* manual.

## 3.5 More Languages Introduced in 3.2-4 and 3.2-5

MarkLogic Server 3.2-4 and 3.2-5 adds the following languages to the list of supported languages:

- Korean
- Persian (Farsi)
- Traditional Chinese (in addition to Simplified Chinese)
- Dutch

## 3.6 Improved PDF Conversion in 3.2-7

In 3.2-7 and later release, the PDF conversion is improved. The improvements include support for PDF files with JBIG2 images as well as many bug fixes.

## 3.7 New Privileges Added in 3.2-9

There are several new privileges added in 3.2-9 to protect changing the root and modules database for `xdmp:eval`, `xdmp:invoke`, and `xdmp:spawn`. For details, see “New Privileges Added in 3.2-9” on page 25.

## 4.0 Known Incompatibilities with Previous Releases

The vast majority of applications implemented on MarkLogic Server 3.1-\* will run either without modifications or with very minor modifications in Release 3.2. There are, however, a number of changes that will cause compatibility issues with 3.1 applications. This section describes those incompatibilities and includes the following topics:

- [Content With Existing xml:lang Attributes](#)
- [Unstemmed Searches Now Require Word Search Indexes](#)
- [Deprecated Options to xdm:unquote Removed and Replaced](#)
- [Minor Changes to xdm:describe Output](#)
- [Non-Breaking Spaces Now Treated as Space, Not as Punctuation](#)
- [Adjacent Punctuation Now Treated as Separate Tokens](#)
- [Handling of Empty Text and Binary Nodes](#)
- [Whitespace Now Preserved in xdm:tidy By Default](#)
- [App Server Default Collation Change](#)
- [Upgraded Range Indexes Use Codepoint Collation, Can Affect Value Lexicon APIs](#)
- [Non-ASCII Characters Now Serialize as the UTF-8 Characters](#)
- [Cannot Load Non-XML Character Entities With Repair Set to NONE](#)
- [NFC Unicode Normalization](#)
- [Demo Server on Port 8000 Now Requires Authentication](#)
- [New Privileges Added in 3.2-9](#)

### 4.1 Content With Existing xml:lang Attributes

When you upgrade from MarkLogic Server 3.1 to MarkLogic Server 3.2, your databases will take on a default language of English. If you have any content that has `xml:lang` attributes with values other than an English locale, then that content, when it is reindexed, will no longer be English content and will therefore no longer match stemmed searches in English.

## 4.2 Unstemmed Searches Now Require Word Search Indexes

MarkLogic Server 3.1 allowed you to specify an unstemmed search against a database that had stemmed searches enabled but word searches disabled. In MarkLogic Server 3.2, because the database is language-aware, the stemmed search index includes language information but the unstemmed word search does not include language information. Therefore, an unstemmed search will search through all languages. For this reason, unstemmed searches with word searches disabled fail with an `XDMP-WORDSEARCH` exception in 3.2.

In 3.2, if you need to perform unstemmed searches with word searches disabled, you can do so in the filter stage using `cts:contains`. You first do the stemmed search in your specified language (or in the default language if you do not explicitly specify a language) in a `let` binding, and then filter the result using `cts:contains` with an `unstemmed` query, as in the following example:

```
let $search := cts:search(doc(), cts:word-query("my words", "stemmed"))
for $x in $search
where cts:contains($x, cts:word-query("my words", "unstemmed"))
return $x
```

For details on how languages (even default languages) affect stemmed and unstemmed searches, see [Querying Documents By Languages](#) in the *Developer's Guide*.

## 4.3 Deprecated Options to `xdmp:unquote` Removed and Replaced

The `xdmp:unquote` built-in function uses the options `repair-none` and `repair-full` to specify whether to perform tag repair while converting a string to a node. In older releases of MarkLogic Server (2.1 and older), these options were named `full` and `none`. The `full` and `none` options were deprecated in Release 2.2, and for MarkLogic Server 3.2, they have now been removed. If you have any code that uses the `full` or `none` options of `xdmp:unquote`, that code will now fail with a syntax error, and you must replace those options with `repair-full` and `repair-none`, respectively.

## 4.4 Minor Changes to `xdmp:describe` Output

The `xdmp:describe` built-in XQuery function behaves slightly differently in MarkLogic Server 3.2 than it did in Release 3.1. In 3.2, there is an optional 3rd parameter to specify the length of the string to print out for each item, with a default length of 64. Previously, it would always print out the entire string. Additionally, for constructed nodes, Release 3.1 prints out the path to the node. Release 3.2 prints out a textual representation of the node, with the content truncated according to the optional third parameter (truncated to 64 characters by default).

If you have some code that relies on the old behavior, you will need to modify it to account for the new behavior.

## 4.5 Non-Breaking Spaces Now Treated as Space, Not as Punctuation

In MarkLogic Server 3.2, non-breaking space characters (&#160;) are treated as whitespace; previously, they were treated as punctuation. If the non-breaking space is adjacent to a word, this will cause searches for that word to match 3.2 (they would not match in 3.1). The new behavior is preferable, as it will increase recall for your searches. For example, the following returns true in 3.2, but returned false in 3.1:

```
(: Returns true in 3.2, false in 3.1 :)
let $x := <root>hello. &#160;goodbye</root>
return
cts:contains($x, cts:word-query("hello. goodbye" ) )
```

To get the 3.2 behavior for searches against content in a database, you must reindex any content loaded in a previous release. If you do not reindex, any searches in 3.2 are still tokenized the new way, while the content in the database is tokenized the old way. Depending on your content, this might cause some searches that include non-breaking space characters that previously matched to no longer match. Reindexing the content will correct that, as well as improve the search recall.

## 4.6 Adjacent Punctuation Now Treated as Separate Tokens

In MarkLogic Server 3.2, multiple punctuation characters that are next to each other are tokenized as separate tokens. Previously, they were coalesced into a single token. The new behavior is preferable, as it will increase recall for your searches. For example, the following returns true in 3.2, but returned false in 3.1:

```
(: Returns true in 3.2, false in 3.1 :)
let $x := <root>hello-!?!goodbye</root>
return
cts:contains($x, cts:word-query("hello-"))
```

This can cause some minor incompatibilities for searches that include one or more adjacent punctuation characters where the content includes two or more adjacent punctuation characters.

To get the 3.2 behavior for searches against content in a database, you must reindex any content loaded in a previous release. If you do not reindex, any searches in 3.2 are still tokenized the new way, while the content in the database is tokenized the old way. Depending on your content, this might cause some searches that include punctuation characters that previously matched to no longer match. Reindexing the content will correct that, as well as improve the search recall.

## 4.7 Handling of Empty Text and Binary Nodes

An empty text or binary node now returns the empty sequence. Previously, they returned an empty string. Therefore, in 3.2, the following return `true`:

```
(: returns true in 3.2, returned false in 3.1 :)
fn:empty( text{} ) ;

(: returns true in 3.2, returned false in 3.1 :)
fn:empty( binary{} )
```

Additionally, `text{""}` and `binary{""}` are now empty when added into a document or element node. However, they return an empty text or binary node if used outside of the context of another node. The following code and comments illustrate some of these changes:

```
(: this is the same in 3.1 and 3.2 :)
<foo>{ text{""} }</foo>
=> <foo/>

(: Returns false in 3.1 and true in 3.2 :)
fn:empty(document{ text{""} }/text())

(: Returns true in 3.1 and false in 3.2 :)
fn:empty( text{ " " } )

(: Returns false in both 3.1 and 3.2 :)
fn:empty( text{""} )
```

If you have any code that relied on the old behavior, you will need to modify the logic so it works with this new behavior.

## 4.8 Whitespace Now Preserved in `xdmp:tidy` By Default

The `xdmp:tidy` function now correctly preserves whitespace when you have `xml:space="preserve"` attributes in the HTML document. Previously, it would sometimes add or remove whitespace.

## 4.9 App Server Default Collation Change

When you create a new App Server, the default collation is the UCA Root Collation (<http://marklogic.com/collation/>) in MarkLogic Server 3.2. Previously, it was the Unicode Codepoint Collation (<http://marklogic.com/collation/codepoint>). While this in itself does not cause an incompatibility, note that code that you run on a new App Server in 3.2 will run in this new collation by default. If you are expecting that code to run in the Unicode Codepoint Collation, you need to specify that, either at the App Server level or in your XQuery code.

## 4.10 Upgraded Range Indexes Use Codepoint Collation, Can Affect Value Lexicon APIs

In MarkLogic Server 3.2, range indexes of type `xs:string` are defined by their QNames *and* by a specified collation. In 3.1, there was only a single collation (codepoint), and all range indexes used that collation, so there was no need to specify a collation to provide uniqueness. Therefore, in 3.2, the collation is necessary to specify a unique `xs:string` range index, both at index creation time and at query time.

When you upgrade from MarkLogic Server 3.1 and have range indexes of type `xs:string`, those indexes remain in the 3.1 default collation, Unicode Codepoint Collation (<http://marklogic.com/collation/codepoint>). When you create new `xs:string` range indexes in MarkLogic Server 3.2, the default collation is now the UCA Root Collation (<http://marklogic.com/collation/>), which is a change from 3.1.

If you are using the value lexicon APIs (for example, `cts:element-values`, `cts:element-attribute-values`), these APIs now take an optional collation parameter. If you do not specify the collation in your query, it will default to the App Server default collation. If the App Server was created in 3.2, it has a different default than if the App Server was created in 3.1. You can override these defaults in the query, either in the XQuery prolog or in the value lexicon API call. Depending on how the combination of your App Server and range indexes are set up, the value lexicon APIs might not specify the range index with the collation you have configured. In such cases, the lexicon APIs throw an exception similar to the following:

```
XDMP-ELEMRIXNOTFOUND: cts:element-values(xs:QName("one"), "",
    "collation=http://marklogic.com/collation/") -- No string element
    range index for xs:QName("one") http://marklogic.com/collation/
```

If you get this error and have upgraded range indexes from 3.1 to 3.2, try changing your code to specify the `http://marklogic.com/collation/codepoint` collation in the value lexicon call, as in the following example:

```
cts:element-values(xs:QName("one"), "",
    "collation=http://marklogic.com/collation/codepoint")
```

For details about collations, including information about how defaults are determined, see the chapter [Language Support in MarkLogic Server](#) in the *Developer's Guide*.

## 4.11 Non-ASCII Characters Now Serialize as the UTF-8 Characters

In previous releases, certain non-ASCII characters were turned into character entities when the XML was serialized out as text. In MarkLogic Server 3.2, all characters are serialized as the actual UTF-8 characters. For characters that do not have a graphic, such as non-breaking spaces, you will no longer see the character entity in the output. For example, the following query returns the character entities for the non-breaking space characters in 3.1, but returns the actual non-breaking space character in 3.2:

```
(: The following query returns
  <mytag>Hello&#160;&#160;&#160;World</mytag> in 3.1,
  returns <mytag>Hello World</mytag> in 3.2
: )
let $x := <mytag>Hello&nbsp;&nbsp;&nbsp;World</mytag>
return $x
```

If you have any code that relies on the entities being serialized, you will have to rework that code. Also, if you are viewing the output of any serialized nodes, it will no longer contain the entities.

The output from MarkLogic Server is UTF-8, so if you display it in a browser that is not set to display UTF-8, it might not display properly. This can happen if the `charset` is set to something other than UTF-8 (for example, if the `charset` is set to ISO-8859-1, the non-breaking space UTF-8 character will not display properly). By default, MarkLogic Server sends an HTTP header specifying UTF-8 for the `charset`, but it is possible to override that header value using the `xdmp:set-response-content-type` API. If you see this issue in your application, explicitly setting the `charset` to UTF-8 as follows might solve the problem (depending on your environment, and assuming you are outputting HTML):

```
xdmp:set-response-content-type("text/html; charset=utf-8")
```

Furthermore, if you are saving the output of a request to an HTML file and then opening the file using `file://` protocol in a browser, some browsers (especially Internet Explorer on Windows) will always default to a Windows character set, causing display problems for any UTF-8 characters that are different than the Windows character equivalents. In these cases, if you explicitly set the browser to display UTF-8 (for example, View > Encodings > UTF-8), they will display correctly. If you are outputting static HTML and expecting people to open it in Internet Explorer browsers using `file://` protocol, consider running a script on the saved files to replace the problem UTF-8 characters with a character entity (for example, to replace the `&#160;` character with `&nbsp;`).

## 4.12 Cannot Load Non-XML Character Entities With Repair Set to NONE

In MarkLogic Server 3.2, you can no longer load documents with non-XML character entities, unless you have repair enabled on the load (for example, `<repair>full</repair>` in the options node for `xdmp:document-load`). Previously, these entities were automatically mapped to a corresponding codepoint.

If you have content that has these non-XML entities (any entity other than `&amp;`, `&apos;`, `&quot;`, `&lt;`, and `&gt;`), you will need to change those to a character entity. For example, if you have `&nbsp;` as part of your XML code to represent a non-breaking space character, you will have to change those instances to the character entity `&#160;` (or an equivalent character entity).

It is also possible to use `xdmp:tidy` with the `<input-xml>yes</input-xml>` option to change these entities while you load them. For example, consider a document on disk with the following content:

```
<myElement>&nbsp;</myElement>
```

If you try to access this file with `xdmp:document-load` or `xdmp:document-get` using the `repair` option set to `none`, it will fail with an `XDMP-DOCENTITYREF` exception because it contains a non-XML entity. If you do the following, however, Tidy will clean up your entity references for you, and you can successfully load the document:

```
xdmp:document-insert("/tidied.xml",
  xdm:tidy(
    xdm:document-get("c:\tmp.xml",
      <options xmlns="xdmp:document-get">
        <format>text</format>
        <repair>none</repair>
      </options>),
    <options xmlns="xdmp:tidy">
      <input-xml>yes</input-xml>
    </options>)[2])
```

Note that it uses the second node from the `xdmp:tidy` output; the second node contains the new XML document, cleaned of any non-XML entities.

## 4.13 NFC Unicode Normalization

MarkLogic Server 3.2 normalizes unicode characters, resulting in some diacritic-sensitive searches matching in 3.2 that would not match in 3.1. This is new feature for 3.2, but if you have code that relies on the old behavior, it will no longer behave the same. For details on the NFC normalization, see “NFC Normalized Characters for Greater Recall” on page 11.

#### 4.14 Demo Server on Port 8000 Now Requires Authentication

In previous releases, the sample demo server configured on port 8000 was set up by default to run as the Admin user. In MarkLogic Server 3.2, it is set up to require authentication. If you are upgrading to 3.2, however, the settings will remain as they were prior to the upgrade. If you have upgraded to 3.2, make sure application-level authentication is disabled for the App Server configured on port 8000 before deploying any production applications.

#### 4.15 New Privileges Added in 3.2-9

Version 3.2-9 adds the following execute privileges for protecting the `modules` or `root` options of `xdmp:eval`, `xdmp:invoke`, and `xdmp:spawn`:

Function	Execute Privilege
<code>xdmp:eval</code> with the <code>modules</code> option	<a href="http://marklogic.com/xdmp/privileges/xdmp-eval-modules-change">http://marklogic.com/xdmp/privileges/xdmp-eval-modules-change</a>
<code>xdmp:invoke</code> with the <code>modules</code> option	<a href="http://marklogic.com/xdmp/privileges/xdmp-invoke-modules-change">http://marklogic.com/xdmp/privileges/xdmp-invoke-modules-change</a>
<code>xdmp:spawn</code> with the <code>modules</code> option	<a href="http://marklogic.com/xdmp/privileges/xdmp-spawn-modules-change">http://marklogic.com/xdmp/privileges/xdmp-spawn-modules-change</a>
<code>xdmp:eval</code> with the <code>modules</code> option to change the filesystem <code>root</code>	<a href="http://marklogic.com/xdmp/privileges/xdmp-eval-modules-change-file">http://marklogic.com/xdmp/privileges/xdmp-eval-modules-change-file</a>
<code>xdmp:invoke</code> with the <code>modules</code> option to change the filesystem <code>root</code>	<a href="http://marklogic.com/xdmp/privileges/xdmp-invoke-modules-change-file">http://marklogic.com/xdmp/privileges/xdmp-invoke-modules-change-file</a>
<code>xdmp:spawn</code> with the <code>modules</code> option to change the filesystem <code>root</code>	<a href="http://marklogic.com/xdmp/privileges/xdmp-spawn-modules-change-file">http://marklogic.com/xdmp/privileges/xdmp-spawn-modules-change-file</a>

If you have applications that use the `modules` or `root` options of `xdmp:eval`, `xdmp:invoke`, or `xdmp:spawn`, you will have to add these privileges to a role and assign it to any users evaluating these options, otherwise an exception is thrown. Previously, no extra privilege was required to use these options.

## 5.0 Changes Made to W3C Functions and Operators in MarkLogic Server 3.1

MarkLogic Server 3.1 included enhancements to selected existing XQuery-standard built-in functions and operators, as well as some XQuery-standard functions and operators from the latest version of the XQuery standard, where such enhancements and additions are expected in the XQuery 1.0 standard and where they do not conflict with existing functionality as defined by the May 2003 XQuery library. In most cases, these changes will not cause any incompatibilities with your existing applications; there are, however, a few cases where they might subtly change the behavior of certain functions in degenerate cases. This chapter lists these changes and includes the following sections.

- [New Functions](#)
- [New Operators](#)
- [New Or Expanded Parameters For Functions](#)
- [Expanded Functionality](#)
- [New Errors](#)

For details on all of the W3C functions implemented in MarkLogic Server 3.1, see the “W3C Built-In Functions” section of the *Mark Logic Built-In and Module Functions Reference*.

### 5.1 New Functions

This section lists new XQuery-standard functions introduced in MarkLogic Server 3.1, and includes the following parts:

- [Replacements For fn:escape-uri](#)
- [Renamed Functions](#)
- [Other New Functions](#)

For the signatures of these functions, see the “W3C Built-In Functions” section of the *Mark Logic Built-In and Module Functions Reference*.

#### 5.1.1 Replacements For fn:escape-uri

In the latest version of the XQuery specification, there are three functions to replace `fn:escape-uri`. The following functions are replacements for the `fn:escape-uri` function:

```
fn:encode-for-uri
```

```
fn:iri-to-uri
```

```
fn:escape-html-uri
```

The `fn:escape-uri` function remains available in MarkLogic Server 3.1.

## Examples:

```
fn:encode-for-uri("http://example.com/Weather/Los%20Angeles#ocean")
=> "http%3A%2F%2Fexample.com%2FWeather%2FLos%2520Angeles%23ocean"
```

```
fn:iri-to-uri("http://example.com/Weather/Los%20Angeles#ocean")
=> "http://example.com/Weather/Los%20Angeles#ocean"
```

```
fn:escape-html-uri("http://example.com/Weather/Los Angeles#ocean")
=> "http://example.com/Weather/Los Angeles#ocean"
```

### 5.1.2 Renamed Functions

The following functions have been renamed. Both the new and the old function names work in MarkLogic Server 3.1.

```
fn:local-name-from-QName (was fn:get-local-name-from-QName)
fn:namespace-uri-from-QName (was fn:get-namespace-from-QName)
fn:namespace-uri-for-prefix (was fn:get-namespace-uri-for-prefix)
fn:in-scope-prefixes (was fn:get-in-scope-namespaces)
fn:years-from-duration (was fn:get-years-from-yearMonthDuration)
fn:months-from-duration (was fn:get-months-from-yearMonthDuration)
fn:days-from-duration (was fn:get-days-from-dayTimeDuration)
fn:hours-from-duration (was fn:get-hours-from-dayTimeDuration)
fn:minutes-from-duration (was fn:get-minutes-from-dayTimeDuration)
fn:seconds-from-duration (was fn:get-seconds-from-dayTimeDuration)
fn:QName (was fn:expanded-QName)
fn:year-from-dateTime (was fn:get-year-from-dateTime)
fn:month-from-dateTime (was fn:get-month-from-dateTime)
fn:day-from-dateTime (was fn:get-day-from-dateTime)
fn:hours-from-dateTime (was fn:get-hours-from-dateTime)
fn:minutes-from-dateTime (was fn:get-minutes-from-dateTime)
fn:seconds-from-dateTime (was fn:get-seconds-from-dateTime)
fn:timezone-from-dateTime (was fn:get-timezone-from-dateTime)
fn:year-from-date (was fn:get-year-from-date)
```

```
fn:month-from-date (was fn:get-month-from-date)
fn:day-from-date (was fn:get-day-from-date)
fn:timezone-from-date (was fn:get-timezone-from-date)
fn:hours-from-time (was fn:get-hours-from-time)
fn:minutes-from-time (was fn:get-minutes-from-time)
fn:seconds-from-time (was fn:get-seconds-from-time)
fn:timezone-from-time (was fn:get-timezone-from-time)
```

The date/time component extraction functions have new semantics with the new names: they now use localized value, not normalized value. For example:

```
fn:hours-from-time( xs:time('15:57:52.185-08:00') )
=> 15

fn:get-hours-from-time( xs:time('15:57:52.185-08:00') )
=> 23
```

### 5.1.3 Other New Functions

The following are other new XQuery-standard functions for MarkLogic Server 3.1.

```
fn:abs (returns the absolute value of a specified argument)
fn:reverse (reverses the order of items in a sequence)
fn:codepoint-equal (equality of strings using codepoint collation)
fn:doc-available (equivalent to fn:exists(fn:doc($x)) )
fn:static-base-uri (a replacement for fn:base-uri with no arguments, which remains the same for 3.1, but will have different semantics in the newest XQuery 1.0 specification)
```

## 5.2 New Operators

The following new operators have been added. Previously, expressions using these operators would throw an exception.

```
xdt:yearMonthDuration div xdt:yearMonthDuration
xdt:dayTimeDuration div xdt:dayTimeDuration
xs:duration eq xs:duration
xs:date - xs:date
```

```
xs:dateTime - xs:dateTime
```

```
xs:duration casting is now allowed. For example:
  xs:duration("PT10H") cast as xdt:dayTimeDuration
```

```
xs:hexBinary and xs:base64Binary are castable to each other
```

### 5.3 New Or Expanded Parameters For Functions

This section lists changes to the parameters for existing functions. Note that in some cases, functions that take a string or node argument no longer throw an error when the parameter is the empty sequence.

- `fn:concat` takes any atomic arguments, not just strings. For example:

```
fn:concat( "this ", 4, <myNode>a</myNode> )
=> "this 4a"
```

- `fn:matches`, `fn:replace`, and `fn:tokenize` use the new flags `x` and `s`.

```
's' enables dot-all mode
```

```
'x' ignore (most) spaces in regular expressions (for example, spaces
for pretty-printing)
```

- `fn:collection()` with no arguments returns everything (like `fn:input()`). This anticipates the removal of the `fn:input` function.
- The following previously returned errors, but now return the empty sequence:

```
fn:node-name( () )
=> ()
```

```
fn:base-uri( () )
=> ()
```

```
fn:document-uri( () )
=> ()
```

```
fn:root( () )
=> ()
```

```
fn:data( () )
=> ()
```

```
fn:string-to-codepoints( () )
=> ()
```

```
fn:resolve-QName( (), $e )
=> ()
```

- The following previously were `xs:decimal` only; `xs:double` arguments would throw an exception:

```
xdt:yearMonthDuration * xs:double
xdt:yearMonthDuration div xs:double
xdt:dayTimeDuration * xs:double
xdt:dayTimeDuration div xs:double
```

- The following functions now have an optional second argument, which is a node that provides the context for the lookup.

```
fn:id
fn:idref
```

## 5.4 Expanded Functionality

This section describes the expanded XQuery functionality in MarkLogic Server 3.1, in preparation for conformance with the latest release of the XQuery 1.0 specification. It includes the following parts:

- [Function Overloading](#)
- [Computed Element and Attribute Constructor Names Atomized](#)
- [Document Content in Element/Document Constructors](#)
- [Some Functions That Previously Returned `xs:string` Now Return `xs:anyURI` and Do Atomic Promotion to `xs:string`](#)

### 5.4.1 Function Overloading

You can now define functions with the same name, as long as the signatures have different numbers of arguments. For example, you can now have the following function definitions in a single module:

```
define function echo ( $x as xs:string )
{
  $x, $x
}

define function echo ()
{
  echo("hello")
}
```

### 5.4.2 Computed Element and Attribute Constructor Names Atomized

Computed element and attribute constructor names are now atomized. For example, the following is now permitted (assuming `qNameNode` is an `xs:QName`):

```
element {$e/qNameNode} {"foo"}
```

Previously, such expressions would throw an exception.

### 5.4.3 Document Content in Element/Document Constructors

You can now use document nodes in an element or document constructor. If you specify a document node in an element constructor, the children of the document node are used (you cannot have a document node child of an element). For example, the following is now permitted:

```
element example { doc("foo.xml") }
```

Previously, such expressions would throw an error saying that element nodes cannot have document node children.

### 5.4.4 Some Functions That Previously Returned `xs:string` Now Return `xs:anyURI` and Do Atomic Promotion to `xs:string`

Similar to how numeric values of `xs:int` are automatically promoted to `xs:double`, values of type `xs:anyURI` are promoted to `xs:string` values where appropriate. Therefore, the following is now permitted:

```
fn:doc( $xhtml-anchor/@href )
```

Previously, such expressions would throw a type error if the attribute `href` was of type `xs:anyURI`.

As a part of this change, the following functions whose natural return type is `xs:anyURI` now have a return type of `xs:anyURI` instead of `xs:string`:

```
fn:base-uri
fn:document-uri
fn:resolve-uri
fn:namespace-URI,
fn:namespace-uri-from-QName
fn:namespace-uri-for-prefix
```

**Note:** Because these functions now return `xs:anyURI` instead of `xs:string`, applications that use these functions and rely on a return type of `xs:string` might need some logic changes to ensure that they work correctly when atomic promotion to `xs:string` does not occur (for example, if you are returning directly to Java or .Net and your Java or .Net code relies on the value being `xs:string`).

## 5.5 New Errors

Some functions that default to use the context item now return errors if there is no context item (or if it is not a node when it needs to be). Previously, these functions returned values. For example:

```
fn:position() was () if there was no context item
```

```
fn:last() was () if there was no context item
```

```
fn:name() was "" if there was no context item or not a node
```

```
fn:local-name() was "" if there was no context item or not a node
```

```
fn:namespace-uri() was "" if there was no context item or not a node
```

```
fn:number() was NaN if there was no context item
```

```
fn:lang($lang) was false if there was no context item or not a node
```

These functions now return an error in these cases. If you have code that relies on these functions always returning values, you might need to enhance that code (for example, catch the errors using a `try/catch` block).

**Note:** If you have such code that relies on context node-related expressions returning values where there is no context node, it is likely not working the way you expect it to anyway. Therefore, if you encounter the new errors, it is likely pointing out an error in your code.

## 6.0 Features Introduced in Release 3.1

This chapter describes the features introduced in Release 3.1 of MarkLogic Server. The feature descriptions are divided into the following categories:

- [Indexing, Search, and Query Enhancements](#)
- [Administrative and Operational Enhancements](#)
- [Other Features](#)
- [XCC—New Client-Side Connectors for Java and .NET](#)

### 6.1 Indexing, Search, and Query Enhancements

The following features provide enhancements to the search and query capabilities in MarkLogic Server:

- [Improved Performance](#)
- [Diacritic-Insensitive Search](#)
- [Trailing Wildcard Indexes](#)
- [Lexicon of Words in the Database](#)
- [Using Lexicons, Trailing Wildcard, and Character Indexes to Resolve Wildcard Queries](#)
- [Constrain cts:query Expressions By Documents, Collections, or Directories](#)
- [Query Constructor API Enhancements](#)
- [Registered cts:queries](#)
- [Alternate Scoring Methods](#)
- [Fast Pagination](#)
- [Bounded Relevance Values: cts:confidence](#)
- [Efficient Estimates When Performing Searches](#)
- [Specify a Cap For xdmp:estimate and fn:count](#)
- [Point-In-Time Queries](#)
- [New Built-In Functions and Operators Added for XQuery 1.0 Forward-Compatibility](#)
- [Select XQuery F&O Built-In Functions Enhanced for XQuery 1.0 Forward-Compatibility](#)
- [Select Enhancements to the XQuery Evaluator](#)

### 6.1.1 Improved Performance

MarkLogic Server 3.1 included many improvements to performance of query requests. General performance improvements (in addition to the specific features described in this chapter) include:

- improved index resolution speeds
- improved node traversal speed of complex XML structures
- cached XQuery modules
- improved order by performance

### 6.1.2 Diacritic-Insensitive Search

MarkLogic Server 3.1 improved support for diacritic characters. By default, content is now indexed to enable broader access, so that searches can return terms with or without diacritics. MarkLogic Server 3.1 introduced a diacritic-sensitive/diacritic-insensitive query option. This option specifies whether searches that include characters that are the root of a diacritic character should return hits that contain the diacritic character. For example, a diacritic-insensitive search for `Jose` can now return hits for `José`. You can now also configure databases for fast diacritic-sensitive recall.

### 6.1.3 Trailing Wildcard Indexes

MarkLogic Server 3.1 introduced trailing wildcard indexes, which optimize wildcard searches when the wildcard is at the end of the search term (for example, `abc*`). You need to configure trailing wildcard indexes in the Admin interface to take advantage of the speedup.

### 6.1.4 Lexicon of Words in the Database

MarkLogic Server 3.1 introduced lexicons, which are dictionaries of the unique words and/or values that occur in a database. MarkLogic Server 3.1 incorporated word, element word, element-attribute word, element value, and element-attribute value lexicons. Lexicons are configured through the Admin interface, and they are accessed with the following new `cts:query` constructor built-in functions:

- `cts:word-match`
- `cts:words`
- `cts:element-word-match`
- `cts:element-words`
- `cts:element-attribute-word-match`
- `cts:element-attribute-words`
- `cts:element-attribute-value-match`
- `cts:element-attribute-values`
- `cts:element-value-match`
- `cts:element-values`

### 6.1.5 Using Lexicons, Trailing Wildcard, and Character Indexes to Resolve Wildcard Queries

MarkLogic Server 3.1 will use the most appropriate combination of available indexes and lexicons, including character indexes, trailing wildcard indexes, and word, element-word, and element-attribute word lexicons to resolve a wildcard query.

### 6.1.6 Constrain `cts:query` Expressions By Documents, Collections, or Directories

MarkLogic Server 3.1 provided the ability to constrain a `cts:query` expression by documents, collections, or directories. This allows you to incorporate document, collection, and/or directory constraints into the `cts:query` expression used as the second argument to a `cts:search` expression, while also taking full advantage of the full complement of boolean constructors. Previously, you could only constrain the documents upon which the search operates by constraining the set of nodes on which the search operates (the first parameter of `cts:search`).

### 6.1.7 Query Constructor API Enhancements

MarkLogic Server 3.1 extended most query constructor APIs so they take a sequence of QNames or a sequence of text arguments, making it easier to programmatically construct complex queries, and allowing MarkLogic Server to further optimize query evaluation. The following `cts:query` constructors have been extended:

- `cts:element-query`
- `cts:element-value-query`
- `cts:element-attribute-value-query`
- `cts:element-word-query`
- `cts:element-attribute-word-query`

### 6.1.8 Registered `cts:queries`

MarkLogic Server 3.1 introduced the capability to resolve a `cts:query` construct once, caching the resulting candidate fragment list for later use and reuse. This provides a mechanism to speed up queries with long and/or complicated query constructs that are used more than once. The optimization precomputes unfiltered data-node term lists, and it therefore introduces the possibility of including false positives, depending on the scenario. The APIs for registered queries are:

- `cts:register`
- `cts:deregister`
- `cts:registered-query` (and the corresponding accessor function)

### 6.1.9 Alternate Scoring Methods

MarkLogic Server 3.1 allowed you to specify a scoring methodology at query time. There are three scoring methods available:

- `score-logtfidf`, which is the classic search engine algorithm taking into account term frequency both in a document and throughout the content set
- `score-logtf`, which takes into account only the term frequency in the content set
- `score-simple`, which gives a score of 8 for every term that has 1 or more matches

### 6.1.10 Bounded Relevance Values: `cts:confidence`

MarkLogic Server 3.1 introduced `cts:confidence`, a relevance measure that returns a value between 0.0 and 1.0. `cts:confidence` ignores any quality settings that might be in effect on the documents.

### 6.1.11 Fast Pagination

MarkLogic Server 3.1 introduced fast pagination, which is the ability to efficiently jump to the *n*th result from a `cts:search` expression. This developer-supplied directive instructs the search not to process skipped parts of the result set. When a search is performed in this mode, results are resolved solely from the indexes, and no document filtering is performed to validate the results. Because results are returned unfiltered, there is a possibility of including false positives in the results, as well as omitting results. There is a new "unfiltered" option to `cts:search` to specify this behavior.

### 6.1.12 Efficient Estimates When Performing Searches

MarkLogic Server 3.1 introduced a new API, `cts:remainder`, which allows for faster estimates of the size of search results when you are also returning all or a subset of those search results.

### 6.1.13 Specify a Cap For `xdmp:estimate` and `fn:count`

MarkLogic Server 3.1 extended the APIs for `xdmp:estimate` and `fn:count` to include an optional new parameter which specifies a maximum bound for the count or estimate. This can save processing time when, for example, you want an estimate or a count of items, but if the count is higher than 1,000,000, than you will call that “more than a million.”

### 6.1.14 Point-In-Time Queries

MarkLogic Server 3.1 introduced point-in-time queries, which allow you to configure your database to retain older versions of documents and query the database based on a specified timestamp. This feature is designed to provide the ability to load the latest revisions of your content and only make it available when you are ready for that content to be released.

### 6.1.15 New Built-In Functions and Operators Added for XQuery 1.0 Forward-Compatibility

MarkLogic Server 3.1 extended the existing Functions & Operators library to include more than 30 new or renamed built-in functions that are expected to belong to the XQuery 1.0 standard, and which do not conflict with the May 2003 XQuery library. Sample builtins include `fn:abs`, `fn:reverse`, and `fn:encode-for-uri`. Sample operators include comparison for `xs:durations` and subtraction for `xs:date` and `xs:dateTime`. For details on the functions implemented in 3.1, see the “W3C Built-In Functions” section of the *Mark Logic Built-In and Module Functions Reference*. These enhancements makes it easier for developers to write XQuery code with an eye to the emerging 1.0 standard. For details on these XQuery additions, see “Changes Made to W3C Functions and Operators in MarkLogic Server 3.1” on page 26.

### 6.1.16 Select XQuery F&O Built-In Functions Enhanced for XQuery 1.0 Forward-Compatibility

MarkLogic Server 3.1 introduced enhancements to selected existing built-in functions and operators, where such enhancements are expected in the XQuery 1.0 standard and where those enhancements do not conflict with existing functionality as defined by the May 2003 XQuery library. Sample enhancements include `fn:concat` accepting any atomic arguments, rather than just arguments of type `xs:string`, and `fn:collection` with no arguments returning all documents. For details on the functions implemented in 3.1, see the “W3C Built-In Functions” section of the *Mark Logic Built-In and Module Functions Reference*. These enhancements makes it easier for developers to write XQuery code with an eye to the emerging 1.0 standard. For details on these XQuery changes, see “Changes Made to W3C Functions and Operators in MarkLogic Server 3.1” on page 26.

### 6.1.17 Select Enhancements to the XQuery Evaluator

MarkLogic Server 3.1 enhanced the XQuery evaluator in a number of ways that are compatible with existing implementations and are expected to be incorporated in the XQuery 1.0 standard. One example is that MarkLogic Server 3.1 allows developers to define multiple functions with the same name, provided that they take a different number of arguments. When the function is called, the number of parameters in the function call determines which function will be invoked. There are a number of other more transparent improvements, most of which simply make it easier to write code that works as expected in the first place.

## 6.2 Administrative and Operational Enhancements

The following features enhance the administrative and operational capabilities of MarkLogic Server:

- [Cache Size Maximum Values Increased to 16GB](#)
- [Cancel a Request](#)
- [Cancel a Merge](#)
- [Merge Policy](#)

- [Resilience to Running Out of Disk Space During a Merge](#)

### **6.2.1 Cache Size Maximum Values Increased to 16GB**

MarkLogic Server 3.1 increased the maximum size of the database and group caches from 4GB to 16GB. That limit only applies to 64-bit platforms; 32-bit platforms still have a cache size limit of 1GB (due to the memory limitations inherent in 32-bit systems).

### **6.2.2 Cancel a Request**

MarkLogic Server 3.1 introduced the capability to cancel a running query or update request, using the Admin interface.

### **6.2.3 Cancel a Merge**

MarkLogic Server 3.1 introduced the capability to cancel a running merge operation, using the Admin interface.

### **6.2.4 Merge Policy**

MarkLogic Server 3.1 introduced a set of controls in the Admin interface to refine merge policy, including maximum merge size and merge blackout periods.

### **6.2.5 Resilience to Running Out of Disk Space During a Merge**

MarkLogic Server 3.1 is less likely to run out of disk space during a merge than previous versions. In 3.1, MarkLogic Server checks the disk space available before a merge begins, and does not start the merge if it looks like not enough space is available. It is still possible to run out of disk space during a merge (for example, if some disk space is used up while a merge is running), but it is less likely for it to happen.

## **6.3 Other Features**

The following are administrative and other miscellaneous new features:

- [XML Classifier API](#)
- [Pre-Commit Triggers](#)
- [CPF With Pre-Commit Triggers](#)
- [Additional Platform Support](#)

### 6.3.1 XML Classifier API

MarkLogic Server 3.1 introduced an XML support vector machine (SVM) classifier API. The classifier API allows you to take a set of training documents that show positive and negative examples of a particular topic, or *class*, as an input, and then allows you run other documents against that training document to see if they belong to any of the classes that you have trained. Because the classifier is XML-aware, you can perform sophisticated classifications based on the content, the structure, or a combination of content and structure of a document. The classifier is highly tunable, and has the following APIs:

- `cts:classify`
- `cts:thresholds`
- `cts:train`

### 6.3.2 Pre-Commit Triggers

MarkLogic Server 3.1 introduced pre-commit triggers. A pre-commit trigger will commit at the same time as the calling transaction, and it will not commit if the entire transaction does not commit. In MarkLogic Server 3.0, there were only post-commit triggers; MarkLogic Server 3.1 includes both pre- and post-commit triggers.

### 6.3.3 CPF With Pre-Commit Triggers

MarkLogic Server 3.1 used the new pre-commit triggers in the Content Processing Framework to provide more robust and more efficient content processing pipeline transitions.

### 6.3.4 Additional Platform Support

MarkLogic Server 3.1 introduced support for the following new operating system platforms:

- Windows 2003 Server 64-bit Edition (x64)
- Red Hat Linux 4.0 (x86 and x64)
- Sun Solaris 10 (SPARC and x64)

For the complete list of supported platforms in MarkLogic Server 3.1, see “Installation and Upgrade” on page 8.

## 6.4 XCC—New Client-Side Connectors for Java and .NET

The following are new features related to XML Contentbase Connector (XCC), which is a new programming model for issuing queries against an XDBC server:

- [XML Contentbase Connector \(XCC\)](#)
- [XCC Retryable Exceptions](#)

### **6.4.1 XML Contentbase Connector (XCC)**

MarkLogic Server 3.1 introduced a new API for connecting to MarkLogic Server from your Java or .NET code. The new API is required to use the new features in 3.1 (for example, point-in-time queries). The old XDBC API is deprecated, but will still work against MarkLogic Server 3.1 (except for 3.1 features). XCC includes architectural changes to increase its robustness and efficiency, so you should migrate your applications to use the new API as quickly as is practical.

### **6.4.2 XCC Retryable Exceptions**

MarkLogic Server 3.1 added the ability for certain XCC exceptions (ones that are deemed *retryable*) to automatically retry the operation. With XDBC, you had to catch the exception and explicitly retry it.

## 7.0 Features Introduced in Release 3.0

This chapter describes the features introduced in Release 3.0 of MarkLogic Server, originally released in June 2005. The feature descriptions are divided into the following categories:

- [Features to Support Content Processing](#)
- [Query, Performance, and Indexing Enhancement](#)
- [Administrative Enhancements and Other Features](#)

### 7.1 Features to Support Content Processing

The following features support Content Processing in MarkLogic Server 3.0:

- [Content Processing Framework](#)
- [Document Conversion Option](#)
- [Expanded HTTP Functions](#)
- [HTTP-Enabled Load and Get Functions](#)
- [HTML Tidy](#)
- [xdmp:invoke and xdm:invoke-in](#)
- [Post-Commit Triggers](#)

#### 7.1.1 Content Processing Framework

MarkLogic Server enables a document to be automatically processed following a series of steps described in one or more processing pipelines. Pipelines may be cascaded, simplifying reuse, and may incorporate calls to external processing services. Pipelines are fully customizable, and are applied independently of the mechanism by which a document was ingested or updated.

#### 7.1.2 Document Conversion Option

With Release 3.0, Mark Logic enhanced its support for document conversion by offering an optional component that automatically converts popular document formats into queryable XML. The Document Conversion Option includes document conversion functions and a set of pre-built processing pipelines that automatically convert HTML, Microsoft Word, Excel, Powerpoint, and Adobe PDF documents into XHTML and a DocBook-like XML format. Prior to 3.0, MarkLogic Server performed all document conversion as part of an external ingestion component.

#### 7.1.3 Expanded HTTP Functions

`xdmp:http-get`, `xdmp:http-post`, `xdmp:http-put`, `xdmp:http-delete`, and `xdmp:http-head` provide low level access to HTTP protocols at query time. This functions can be used to access external web services.

### 7.1.4 HTTP-Enabled Load and Get Functions

Two new built-ins, `xdmp:document-load` and `xdmp:document-get`, have been added. These built-ins incorporate HTTP support for fetching UTF-8 encoded content directly from external HTTP servers. `xdmp:load` and `xdmp:get` are deprecated (but still available) in this release.

### 7.1.5 HTML Tidy

HTML Tidy is now built into MarkLogic Server. The `xdmp:tidy` function provides full HTML tidy functions from within XQuery.

### 7.1.6 `xdmp:invoke` and `xdmp:invoke-in`

These functions invoke a module at the specified path and return the results of the module evaluation. Evaluation can be parameterized using external variables. Variants of these built-ins are also accessible at the XDBC layer.

### 7.1.7 Post-Commit Triggers

MarkLogic Server 3.0 included post-commit data triggers and database restart triggers to support the content processing activities. Background processing can be initiated using the `xdmp:spawn` function.

## 7.2 Query, Performance, and Indexing Enhancement

The following features enhance the query functionality, performance, and indexing capabilities of MarkLogic Server:

- [Proximity Queries](#)
- [Text Highlighting](#)
- [Side Effects With `xdmp:set`](#)
- [Reindexing and Refragmenting](#)
- [String Range Indexes](#)
- [Wildcard Index Enhancements](#)
- [Order By Optimization](#)
- [Property Axis](#)
- [Ordered `cts:and-query`](#)
- [`cts:element-query`](#)
- [Element Word Query Through Rules](#)
- [XPath Union Optimization](#)

### 7.2.1 Proximity Queries

MarkLogic Server 3.0 provided document-level and element-level indexing options to enable breakthrough XML-aware proximity search operators. Developers can combine structure and content in ordered and unordered proximity relevance search expressions using the new `cts:near-query` operator. Proximity indexes are also used to improve search performance and relevance measures for multi-term phrases.

### 7.2.2 Text Highlighting

`cts:highlight` provides programmable support for highlighting search results or otherwise marking up complex text structures. It supports the same search query expressions used by `cts:search`.

### 7.2.3 Side Effects With `xdmp:set`

MarkLogic Server 3.0 introduced a new built-in function that allows you to set the value of a variable, enabling you to introduce side effects into a query by changing the value of a variable to something other than what it was originally bound. This is particularly useful for contextually-dependent highlighting.

### 7.2.4 Reindexing and Refragmenting

MarkLogic Server 3.0 introduced online (background) reindexing and refragmentation of existing databases. This capability can be enabled and disabled through the Admin interface.

### 7.2.5 String Range Indexes

MarkLogic Server 3.0 introduced support for configuring element and element-attribute range indexes for the `xs:string` data type. This allows the optimization of both string inequality predicates and string-based order by clauses.

### 7.2.6 Wildcard Index Enhancements

MarkLogic Server 3.0 included three kinds of wildcard (character) indexes, providing greater flexibility, as well as providing better performance and smaller index sizes when you only use the three character indexes (allowing wildcard searches with at least 3 consecutive characters, such as `abc*`).

### 7.2.7 Order By Optimization

Certain order-by clauses are now resolved directly out of range indexes (if available), providing significant speed-up.

### 7.2.8 Property Axis

MarkLogic Server 3.0 introduced a new XPath axis, the `property::` axis, that allows a single XPath expression to make step tests against a document and its properties. 3.0 also incorporates specific optimizations designed to speed the evaluation of certain uses of the `property::` axis.

### 7.2.9 Ordered cts:and-query

An additional parameter to `cts:and-query` allows you to specify that the and-ed subqueries must occur in order. Word Position indexes must be enabled to take advantage of this feature.

### 7.2.10 cts:element-query

MarkLogic Server includes `cts:element-query`, which is a new search query constructor that allows you to construct combinations of search query primitives that match within the same element. You must enable Word Position and Element Word Position indexes to use `cts:element-query`.

### 7.2.11 Element Word Query Through Rules

Like phrase-through rules, `element-word-query-through` rules allow searches to ignore specific elements boundaries when performing searches. Using these new rules, phrase searches can be performed against both the parent element and the element specified by the `element-word-query-through` rule. These rules are applied recursively.

### 7.2.12 XPath Union Optimization

Certain XPath expressions involving unions (for example, `/a/(b|c)`) have been optimized. In addition, these expressions are now considered searchable path expressions, and can be used in `cts:search`, `cts:highlight` and other constructs requiring searchable path expressions.

## 7.3 Administrative Enhancements and Other Features

The following are administrative and other miscellaneous new features:

- [Status Screens and Other Admin Interface Changes](#)
- [Moving Documents Between Forests](#)
- [xdmp:subbinary](#)
- [Base-64 Encoding/Decoding](#)
- [Last Modified Property](#)
- [Permissions, Collections, and Quality Inheritance](#)
- [XDBC: Multiple Documents in DocInsertStream\(\)](#)
- [XDBC API for Microsoft .NET](#)
- [Support Request Facility in Admin Interface](#)

### 7.3.1 Status Screens and Other Admin Interface Changes

MarkLogic Server 3.0 introduced a new look and feel for the Admin Interface, incorporating new cross-linkages to speed navigation within the admin environment. Summary screens provide at-a-glance reports of system configuration. The Admin Interface incorporates new reports for displaying system status, on a per resource or aggregate basis. The Admin Interface also includes a new tool for collecting system configuration and status information to simplify communication with Mark Logic's support team.

### 7.3.2 Moving Documents Between Forests

When inserting or loading a document, specifying a forest placement will now place the document into the specified forest, even if the document already exists elsewhere in the database. The previous version of the document will be deleted.

### 7.3.3 `xdmp:subbinary`

MarkLogic Server 3.0 introduced a function to extract a “substring” from within a binary document.

### 7.3.4 Base-64 Encoding/Decoding

MarkLogic Server 3.0 introduced built-in base64 encoding and decoding functionality, useful for some SOAP-style transactions.

### 7.3.5 Last Modified Property

With MarkLogic Server 3.0, MarkLogic Server will automatically update a newly added Last Modified Date document property if this feature is enabled.

### 7.3.6 Permissions, Collections, and Quality Inheritance

MarkLogic Server 3.0 introduced database configuration options which allow newly ingested documents to inherit their collections, their permissions, and their quality setting from their parent directory.

### 7.3.7 XDBC: Multiple Documents in `DocInsertStream()`

MarkLogic Server 3.0 introduced the ability to include multiple documents in an XDBC `XDMPDocInsertStream()`.

### 7.3.8 XDBC API for Microsoft .NET

MarkLogic Server 3.0 introduced the C# XDBC library available for Microsoft .NET, allowing you to create .NET applications which interact with MarkLogic Server.

### **7.3.9 Support Request Facility in Admin Interface**

MarkLogic Server 3.0 introduced a facility in the Admin Interface to save all of your configuration information for your installation. It gathers all of the configuration files and logs for your installation and allows you to save them to a file or display them in the browser. This information is useful for Mark Logic Support, and can help in solving any problems you might encounter with the software.

## 8.0 Other Notes

This section provides the following information about MarkLogic Server:

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

### 8.1 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. The automatic configuration only takes into account memory up to 16 GB. If your system has more than 16 GB of memory, or 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 one CPU per 16 GB of source content.

Pragmatically, we recommend running most configurations with a minimum of two CPUs.

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.

## 8.2 Compatibility with XQuery Drafts

This release implements the XQuery language, functions and operators specified in the May 02, 2003 W3C XQuery Working Group Draft Recommendations:

- <http://www.w3.org/TR/2003/WD-xquery-20030502>
- <http://www.w3.org/TR/2003/WD-xpath-functions-20030502>

Additionally, much of the added functionality in the January 2007 W3C XQuery Recommendation is implemented in MarkLogic Server 3.2.

## 8.3 XQuery Extensions

Working within the W3C Draft Recommendations for XQuery 1.0, 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 retrieval and data manipulation functions and administrative tools.

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

## 8.4 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>	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>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>3.2 Release Notes</i>	Contains a summary of new features, upgrade compatible information, and known issues.
XCC Javadoc	API documentation for the Mark Logic XML Contentbase Connector Java API (XCC/J).
<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>Introduction to XQuery</i>	A condensed overview of the XQuery language. This book includes some syntax information, but is primarily intended as an introduction 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 Drafts” on page 48. 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> for more details.

## 8.5 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 5.5 or newer and Mozilla 1.5 or newer (including Firefox).

## 8.6 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](mailto:support@marklogic.com)

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