MarkLogic Server

Release Notes

MarkLogic 8
February, 2015

Last Revised: 8.0-9, August, 2018

Copyright © 2018 MarkLogic Corporation. All rights reserved.

Table of Contents

Release Notes

1.0 Introduction 8

2.0 Installation and Upgrade 9
- 2.1 Supported Platforms 9
- 2.2 Supported Filesystems 9
- 2.3 Upgrade Support 9

3.0 New Features in MarkLogic 8 11
- 3.1 Server-Side JavaScript 11
- 3.2 Native JSON 12
- 3.3 Samplestack Sample Application 12
- 3.4 Temporal Documents 12
- 3.5 REST Management API Improvements 13
- 3.6 More Semantics Features, Including SPARQL 1.1, Inferencing, and SPARQL UPDATE 13
- 3.7 Node.js Client API 13
- 3.8 REST and Java Client API Improvements 13
- 3.9 Enhanced HTTP Server Features 14
- 3.10 Flexible Replication Enhancements 15
- 3.11 Incremental Backup 15
- 3.12 Document Library Services (DLS) Improvements 15
- 3.13 MLCP Enhancements 15
- 3.14 New Features in MarkLogic 8.0-3 16
  - 3.14.1 Compatibility With New Versions of Hadoop 16
  - 3.14.2 MarkLogic Connector for Hadoop Supports JavaScript Queries 16
  - 3.14.3 XCC Support for JSON Content 17
  - 3.14.4 HDFS Client Bundles Available for Download 17
  - 3.14.5 Arithmetic and Comparison Methods for dateTime and Duration Objects in Server-Side JavaScript 17
  - 3.14.6 Expanded instanceof Support in Server-Side JavaScript 17
  - 3.14.7 New Features in the Client APIs 17
- 3.15 New Features in MarkLogic 8.0-4 18
  - 3.15.1 Server-Side JavaScript Features 18
    - 3.15.1.1 Server-Side JavaScript Search API (jsearch) 18
    - 3.15.1.2 cts:parse 19
    - 3.15.1.3 Alerting API JavaScript Enhancements 19
    - 3.15.1.4 Thesaurus and Spelling API Enhancements 19
  - 3.15.2 Semantics Enhancements 19

3.15.2.1 SPARQL Negation 19
3.15.2.2 Java Client API Semantics Enhancements 20
3.15.2.3 MarkLogic Sesame API 20
3.15.2.4 MarkLogic Jena API 20
3.15.2.5 Semantics REST API Additions 20
3.15.2.6 mlcp Enhancements 20
3.15.3 Management REST API Enhancements 20
3.15.4 Geospatial Search Enhancements 20
3.15.4.1 Converting To/From Common Representations 21
3.15.4.2 New Geospatial Utility Functions 22
3.15.4.3 Support for More Recent KML and GML Versions 22
3.15.5 Support For Red Hat Enterprise Linux 7 23
3.15.6 Clock Skew Timeouts 23
3.15.7 Node.js Client API Supports Searching XML with QBE 23
3.15.8 New Features in mlcp 23
3.16 New Features in MarkLogic 8.0-6 24
3.16.1 MapR Support in mlcp and the Hadoop Connector 24
3.17 New Features in MarkLogic 8.0-7 24
3.17.1 Ability to Access and Modify Metadata in a Transformation 24
3.17.2 New Server-Side Transaction Controls 24
3.17.3 XCC: New Session Methods for Transaction Control 26
3.17.4 mlcp: Greater Control over Host Connections 26
3.17.5 REST Management API: Advancing LSQT 26
3.17.6 REST Client API: Advancing LSQT 27
3.18 New Features in MarkLogic 8.0-8 27
3.18.1 Separate Download and Installation of Converters and Filters 27
3.18.2 SQL Deep Analysis Trace Event 27

4.0 Known Incompatibilities with Previous Releases 28
4.1 JSON Related Incompatibilities 28
4.1.1 Documents Created as JSON With MarkLogic 7 REST API or MLCP Must Be Converted to Native JSON 29
4.1.2 json:unquotedString Primitive No Longer Available 30
4.1.3 xdmp:to-json and json:transform-to-json Now Returns a document-node() 31
4.1.4 Search, Java, REST: json-key Is Now json-property in Options and Structured Query 31
4.1.5 Java and REST: Specifying a Language for JSON Documents is Deprecated 32
4.1.6 Java and REST: Default Path Language for JSON Document Patches is Now XPath 32
4.1.7 Java and REST: New Restrictions on Patching JSON Content 33
4.1.8 Java and REST: Transforms and Extensions That Manipulate JSON Must Be Rewritten 33
4.1.9 Java and REST: JSON Array Items and Property Values No Longer Distinguishable in QBE 34

4.1.10 Field Range Query and Field Value Query on JSON May Behave Differently 34
4.2 Semantics Incompatibilites 35
4.2.1 Changed Function: sem:sparql 35
4.2.2 Changed Function: sem:sparql-values 35
4.2.3 Changed Function: sem:sparql-values 36
4.2.4 Deprecated Function: sem:sparql-triples 36
4.2.5 Changed Behavior: Graphs 36
4.3 REST and Java Client API Incompatibilites 36
4.3.1 Must Upgrade to Java Client API v3.0 37
4.3.2 REST API Instance Must Use the Declarative Rewriter on the App Server 37
4.3.3 Default Transaction Mode for the POST Method of Resource Service Extensions is Now Query 37
4.3.4 REST API: Empty Bulk Read by Query Now Returns 200 Status 37
4.3.5 Error Reporting Format and Detail Changes 38
4.3.5.1 Error Format Defaults to JSON and is a REST Instance Creation Property 38
4.3.5.2 Error Detail Element and Property Names Have Changed 39
4.3.5.3 Use RESTAPI-SRVEXERR to Report Errors from Transforms and Extensions 39
4.3.6 Deprecated Interface: Keyvalue Queries 39
4.3.7 Transaction ID Format Has Changed 40
4.3.8 A Transaction Can No Longer Be Shared Across Users 40
4.3.9 Java: QBE Search Results No Longer Automatically Match the Query Format 40
4.4 Document Library Services (DLS) Repositories Need To Perform A Bulk Upgrade Operation 41
4.5 Linux Now Requires Red Hat 6 43
4.6 mlsql On Linux No Longer Ships With Server 43
4.7 Cyrillic Tokenization Changes 43
4.8 Application Builder Applications Must Be Re-Deployed in MarkLogic 8 44
4.9 Application Builder and Information Studio Links Removed 44
4.10 Search API Incompatibilities 44
4.10.1 search:parse Output is Now Unannotated cts:query XML 45
4.10.2 Deprecated Option: extract-metadata 45
4.10.3 Deprecated Functions: search:unparse, search:remove-constraint 45
4.10.4 Structured Query: locks-query and properties-query Renamed 46
4.10.5 sort-order Query Option Requires an Index 46
4.11 Locks and Properties Query Built-In Functions Renamed 46
4.12 xdmp:uri-content-type Of an XML Document Now Returns application/xml, Can Affect CPF Applications 47
4.13 xdmp:function Signature Change 47
4.14 Incompatibilities Between 8.0-9 and 8.0-8 47
4.14.1 LDAP Connection Change 47
4.15 Incompatibilities Between 8.0-8 and 8.0-7 48

4.15.1 The ldapSearch Results Format Changed 48
4.16 Incompatibilities Between 8.0-6 and 8.0-7 48
4.16.1 Changes to jsearch.facets Output Structure 48
4.16.2 Array Type is Preserved in x509 Certificate with Array-Valued Properties 49
4.16.3 Changes to xdmp:update XQuery Prolog Option 49
4.16.4 map:new Retains Keys with Empty Values 50
4.17 Incompatibilities Between 8.0-5 and 8.0-6 50
4.17.1 Terms Matched by additional-query Are Highlighted in Snippets 50
4.17.2 Change to Required Software for mlcp and the Hadoop Connector 50
4.17.3 Change to Required Software for the Node.js Client API 51
4.17.4 Change to Rewriter Regex Code 51
4.18 Incompatibilities Between 8.0-3 and 8.0-4 51
4.18.1 xdmp.multipartDecode Now Returns a JSON Payload for Headers 51
4.18.2 In JavaScript, Some Thesaurus and Spelling Function Have Different Return Type 51
4.18.3 xdmp.databaseRestoreStatus Now Returns an Object 52
4.18.4 Serialization Error Code Changes 52
4.18.5 Change to Required Java Version 52
4.18.6 Deprecated mlcp Command Line Options 53
4.18.7 REST APIs That Have JSON or XML Payloads Cannot Have Empty Payloads 53
4.18.8 Geospatial Namespace and Data Version Changes 53
4.18.8.1 GML and KML Library Modules Moved to a New Namespace .. 53
4.18.8.2 Some Built-In Geospatial Functions Moved to geo Namespace ... 54
4.18.8.3 Older GML and KML Versions Deprecated 55
4.19 Incompatibilities Between 8.0-2 and 8.0-3 55
4.19.1 spell.suggestDetailed, xdmp.filesystemDirectory, and xdmp.encodingLanguageDetect Now Return ValueIterator 56
4.19.2 xdmp.databaseRestoreStatus Now Returns an Array 56
4.19.3 xdmp.gssServerNegotiate Now Returns a JavaScript Object 56
4.19.4 Use of String Transaction Ids in Node.js To Be Deprecated 56
4.19.5 CDH 4.3 is No Longer a Supported Hadoop Distribution 57
4.19.6 Changes to How MarkLogic Locates Java and Hadoop Libraries for HDFS Forest Storage 57
4.20 Incompatibilities Between 8.0-1 and 8.0-2 57
4.20.1 Array Input Differences in fn.distinctValues, fn.subsequence, and Other Functions 58
4.20.2 The Second Parameters of xdmp.eval, xdmp.invoke, xdmp.xqueryEval, and xdmp.spawn Now Take a Single Object 58
4.20.3 extract-document-data Results Now Inline By Default 58
4.20.4 XCC v8.0-2 May Require Config Change When Used with Older Versions of MarkLogic 59
4.20.5 Client APIs: JavaScript Extension and Transform Error Reporting

Convention Change 59
4.20.6 Some JavaScript Built-In Functions that Returned XML Structures Now Return JSON Structures 60
4.21 Planning for Future Upgrades 60
4.21.1 ValueIterator to be Replaced with Sequence 60
4.21.2 xdmp:transaction-mode XQuery Prolog Option Deprecated 61
4.21.3 Deprecation of transaction-mode Option to xdmp:eval 62
4.21.4 XCC Session.setTransactionMode is Deprecated 62
4.22 MarkLogic 7 Incompatibilites 63
4.22.1 Incompatibilities Between MarkLogic 7.0-3 and 7.0-2 63
4.22.1.1 HDP No Longer a Supported Hadoop Platform 64
4.22.1.2 Java API: ContentVersionRequest Property Deprecated 64
4.22.1.3 REST API: content-versions Property Deprecated 64
4.22.1.4 REST API: JSON documents Cannot be Retrieved as XML 65
4.22.2 Float Precision Greater in 7.0-3 65
4.22.3 Incompatibilities Between MarkLogic 7.0-2 and 7.0-1 65
4.22.3.1 Change to JSON Output from the REST API 65
4.22.3.2 Changes to the MarkLogic Connector for Hadoop API 66
4.22.4 Incompatibilities Between MarkLogic 7.0-1 and MarkLogic 6 66
4.22.4.1 XQuery HTTP Client Built-In Functions Now Require a Privilege 66
4.22.4.2 HTTP Client Functions Are Now HTTP 1.1 Compliant 67
4.22.4.3 xdmp:get-request-username and xdmp:get-request-user Changes 67
4.22.4.4 Specifying a Forest Now Only Works With Strict Locking 67
4.22.4.5 Custom Dictionaries for Japanese and Chinese Languages Need to be Re-saved 68
4.22.4.6 Default Attributes on XML Copy Changes 68
4.22.4.7 Serialization of Alerting, Reverse, and Path Range Queries Change 69
4.22.4.8 Java and REST Client API Incompatibilities 70
4.22.4.9 Namespace Change for Properties Persisted Using JSON 71
4.22.4.10 mlcp Incompatibilities 77
4.22.4.11 REST Management API Version Incremented to v2 78
4.22.4.12 Changes to the Configuration Manager 81
4.22.4.13 xdmp:plan Now Requires a Privilege 81
4.22.4.14 fn:analyze-string Now Returns Output in a Different Namespace 81
4.23 Incompatibilities Between MarkLogic 6 and 5.0 82
4.23.1 32-bit and Solaris SPARC Platforms are No Longer Supported 82
4.23.2 Default Database Settings Changes 82
4.23.3 Serialization Changes for xdmp:from-json and xdmp:to-json 83
4.23.4 fn:format-dateTime Functions Now Round to 7 Places Instead of 3 83
4.23.5 xdmp:sha1, xdmp:sha256, xdmp:sha512 Not Supported in FIPS Mode ..83
4.23.6 Values of Type xs:date Sometimes Serialize With Timezone 84
4.23.7 DocumentInputFormat Default Input Value Class is MarkLogicDocument

84
4.23.8 Application Builder Generated App Differences 85
4.23.9 app-user Role Now Has the rest-reader Role 85
4.23.10 fn:resolve-uri in XSLT Now Requires a Scheme When Using the Second Parameter 86
4.23.11 MarkLogic Content Pump 1.0-2 Requires MarkLogic Server 6.0-2 86
4.23.12 Starting With 6.0-2, XCC Has a New insertContentCollectErrors() Method on the Java Session Interface 86
4.23.13 New Aggregate Input Default URI in mlcp 1.0-3 87
4.23.14 As of 6.0-5, HDP is no longer a supported Hadoop platform 87

5.0 Other Notes 88
5.1 Memory and Disk Space Requirements 88
5.2 Compatibility with XQuery Specifications 89
5.3 XQuery Extensions 89
5.4 Documentation 90
5.5 Browser Requirements 94

6.0 Technical Support 95

7.0 Copyright 96
7.0 NOTICE 96

1.0 Introduction

MarkLogic 8 is a major release of MarkLogic Server that includes many new features. The new features are described in “New Features in MarkLogic 8” on page 11. The following lists some of the major features with links to where they are described:

- Server-Side JavaScript
- Native JSON
- Samplestack Sample Application
- Temporal Documents
- More Semantics Features, Including SPARQL 1.1, Inferencing, and SPARQL UPDATE

For a description of these and many more new features, see “New Features in MarkLogic 8” on page 11.

If you are upgrading from MarkLogic 7, some applications will require minor changes to run correctly on MarkLogic 8. For details, see “Known Incompatibilities with Previous Releases” on page 28.

For a list of bugs fixed in the latest maintenance release and a list of known bugs, see the MarkLogic 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
- Supported Filesystems
- Upgrade Support

2.1 Supported Platforms

For a complete list of support platforms, see Supported Platforms in the *Installation Guide*.

2.2 Supported Filesystems

For a complete list of supported filesystems, see Supported Filesystems in the *Installation Guide*.

2.3 Upgrade Support

This section describes upgrade support to MarkLogic 8. For details on installing MarkLogic Server and for the upgrade procedure, see the *Installation Guide*.

Upgrading is supported from 5.0-6 or later. If you are running a release prior to 5.0, you must first upgrade to MarkLogic 5, MarkLogic 6, or MarkLogic 7 before upgrading to MarkLogic 8. If you are upgrading a cluster, you must first upgrade the node in which the Security database forest is located before you upgrade other nodes in the cluster. MarkLogic Corporation strongly recommends performing a backup of your databases before upgrading to MarkLogic 8. Additionally, MarkLogic Corporation recommends that you first upgrade to the latest maintenance release of MarkLogic 5, MarkLogic 6, or MarkLogic 7 before upgrading to MarkLogic 8.

An upgrade from 5.0 to MarkLogic 8 will reindex any databases that have `reindex enable` set to `true`. Upgrading from MarkLogic 6 or later does not require a reindex. If you choose not to reindex your databases, they will run in either 5.0 compatibility mode, depending on the version of MarkLogic Server in which they were last loaded or reindexed. Running in compatibility mode will disable certain MarkLogic 8 features and may treat all content in the database as English language content. For details on database compatibility, see the *Installation Guide*.

MarkLogic 7 and later includes a new rebalancing feature with a more efficient document placement algorithm. Upon upgrade, databases from previous MarkLogic releases are set to use the `legacy` document assignment policy, which is the same as used in previous MarkLogic releases. If you do plan on reindexing an upgraded database, MarkLogic recommends that you consider setting your databases to use the new `bucket` document assignment policy. The `bucket` policy is more efficient for rebalancing your database across forests if you add or remove forests from your configuration. For more details, see Database Rebalancing in the *Administrator's Guide*.

There are some known incompatibilities between MarkLogic 8 and MarkLogic 7. You might need to make some minor code changes to your MarkLogic 7 applications before they can run correctly in MarkLogic 8. For details on the incompatibilities, see “Known Incompatibilities with Previous Releases” on page 28. For instructions on upgrading to MarkLogic 8, including information about database compatibility between MarkLogic 8 and MarkLogic 7, see the *Installation Guide*.

If you are upgrading to MarkLogic 8.0-8 or later version 8 release, you may have to install MarkLogic Converters package separately. For more details, see MarkLogic Converters Installation Changes in Version 8 Releases Starting at 8.0-8 in the *Installation Guide*.

3.0 New Features in MarkLogic 8

This chapter describes the new features in MarkLogic 8.

- Server-Side JavaScript
- Native JSON
- Samplestack Sample Application
- Temporal Documents
- REST Management API Improvements
- More Semantics Features, Including SPARQL 1.1, Inferencing, and SPARQL UPDATE
- Node.js Client API
- REST and Java Client API Improvements
- Enhanced HTTP Server Features
- Flexible Replication Enhancements
- Incremental Backup
- Document Library Services (DLS) Improvements
- MLCP Enhancements
- New Features in MarkLogic 8.0-3
- New Features in MarkLogic 8.0-4
- New Features in MarkLogic 8.0-6
- New Features in MarkLogic 8.0-7
- New Features in MarkLogic 8.0-8

3.1 Server-Side JavaScript

MarkLogic 8 includes JavaScript as a server-side programming language, allowing you to access all of the powerful search and data manipulation capabilities of MarkLogic in a language that is familiar to many developers. Combined with native JSON document support, you can easily build JSON-based data services in JavaScript using MarkLogic.

The Server-Side JavaScript is implemented using the Google V8 JavaScript engine, which is an open-source, high-performance JavaScript engine implemented in C++ that is compiled into MarkLogic Server. There are over 1,000 MarkLogic built-in functions available directly from Server-Side JavaScript, allowing you access to data and search in JavaScript.

The JavaScript support is in addition to the XQuery support, and in fact, you can import XQuery libraries into JavaScript programs, making all of your XQuery code immediately usable in a JavaScript program.

For details on the MarkLogic-specific extensions to JavaScript, see the *JavaScript Reference Guide*.

3.2 Native JSON

MarkLogic 8 includes support for JSON as a native document format. Along with the other document formats (XML, text, and binary), JSON documents are stored in the database with all of the enterprise features you expect in MarkLogic. They are indexed using the universal index (like XML and text documents), and, like XML documents, you can also create range indexes on JSON document properties and path indexes on JSON paths. Similarly, you can create fields on JSON documents. Therefore you can perform fast and complex searches across JSON documents.

JSON has become extremely popular in web applications, and is very popular with JavaScript developers. It is therefore an excellent match for the Server-Side JavaScript and for the Node.js Client API features. MarkLogic 8 also includes a set of constructors in XQuery to make it easy to work with JSON documents in XQuery.

You can choose to model your data in the format that makes sense for your application. Also, you do not have to model everything in one format. Use JSON where it makes sense, use XML where it makes sense, and use RDF triples where they make sense. All MarkLogic document formats are designed to co-exist, working together as your application requirements dictate. For more details on JSON, see Working With JSON in the *Application Developer's Guide*.

3.3 Samplestack Sample Application

Samplestack is a three-tier application that uses MarkLogic as its data layer, and demonstrates a reference architecture for a popular way of building applications. Samplestack is an open source project on GitHub available at the following URL:

https://github.com/marklogic/marklogic-samplestack

Samplestack is a demonstration application that uses data from Stack Overflow to create a question and answer application. For details about Samplestack, see the GitHub project. For an overview of the reference architecture, see Understanding the Reference Architecture in the *Reference Application Architecture Guide*.

3.4 Temporal Documents

MarkLogic 8 includes support for temporal documents, typically used for bitemporal applications. A temporal document stores information about the valid time as well as the system time for a document. This allows you to store date as it was known at various times throughout a document's lifecycle. This is useful in many "what you knew when you knew it" type of applications, often necessary in compliance applications.

For details about using temporal documents in MarkLogic, see the *Temporal Developer's Guide*.

3.5 REST Management API Improvements

The REST Management API has been significantly expanded in MarkLogic 8. You can now script almost any management task via the REST API, allowing you to create management scripts in whatever scripting language you like, whether that is python, bash, ruby, php, or anything else that allows you to make HTTP calls. For details about the REST Management API, see the *MarkLogic REST API Reference* and the *REST Application Developer's Guide*.

3.6 More Semantics Features, Including SPARQL 1.1, Inferencing, and SPARQL UPDATE

MarkLogic 8 extends the use of standard SPARQL, enabling you to perform analytics (aggregates) over triples; explore semantics graphs using property paths; and update semantic triples; all using the standard SPARQL 1.1 language over standard protocols. Specifically, MarkLogic 8 includes the following enhancements to semantics:

- SPARQL 1.1 Support
- Inferencing
- SPARQL Update Support
- Aggregates
- Property Paths

For details on using the enterprise triple store in MarkLogic, see the *Semantics Developer's Guide*. For details on inferencing, see Inference, for details on SPARQL Update, see SPARQL Update, for details on aggregates, see SPARQL Aggregates.

3.7 Node.js Client API

The MarkLogic Node.js Client API is an open source JavaScript library for Node.js, allowing you to quickly and reliably access MarkLogic from a Node application. Node.js (nodejs.org) is a popular platform for building three-tier applications, where Node.js is typically the middle tier. The MarkLogic Node.js Client API GitHub project is available at the following URL:

https://github.com/marklogic/node-client-api

The Node.js Client API is available on GitHub, and uses Node technologies you would expect such as `npm` (Node Packaging Manager). For details on getting started with the Node.js Client API, see the GitHub project and Introduction to the Node.js Client API the *Node.js Application Developer's Guide*.

3.8 REST and Java Client API Improvements

The REST Client API and the Java Client API have extensive enhancements to both search and CRUD (create, read, update, delete) operations. The Search enhancements include:

- `extract-document-data` query option

- `geo-json-*` support in structured query
- Temporal document support
- Native JSON query and update support

Additional REST and Java client enhancements include:

- JavaScript resource service extensions and content transformations.
- New endpoints for evaluating (eval) and invoking (invoke) code (both JavaScript and XQuery code).
- Runtime database selection: for the REST Client API, database selection is per request; for the Java Client API, database selection is per `DatabaseClient` (*not* per request).
- Bitemporal support.
- New config properties for configuring REST instances: `forests-per-host`, `error-format`, `xdbc-enabled`.
- REST instance available on port 8000 for all installations.
- Bulk operations for both REST and Java (write, read by URI, read by search).
- A new POJO data binding interface for the Java Client API, allowing you to persist plain old Java objects (POJOs) into MarkLogic for storage and retrieval.

Additionally, the Java Client API is now an open source project on GitHub available at the following URL:

https://github.com/marklogic/java-client-api

3.9 Enhanced HTTP Server Features

The HTTP App Server in MarkLogic 8 includes a declarative XML rewriter, and the default rewriters used by a REST API Instance (as well as the one on port 8000) allow you to use a single App Server for multiple applications, including application that use the REST API, The Java Client API, the Node.js Client API, MLCP, or any XCC application that previously required an XDBC App Server. There is a REST endpoint `POST /v1/rest-apis`) to create an instance.

For the App Server available in all installations on port 8000, the enhanced features in the HTTP App Server make is very easy for new users to run code without needing to create a separate REST API instance or a separate XDBC App Server; instead just point a REST client or an XCC program (like MLCP) to port 8000 of your MarkLogic installation. For details on REST instances, see Administering REST Client API Instances in the *REST Application Developer's Guide*, and for details on the declarative rewriter, see Creating a Declarative XML Rewriter to Support REST Web Services in the *Application Developer's Guide*.

3.10 Flexible Replication Enhancements

Flexible Replication is an existing feature in MarkLogic that makes it easy to copy some or all parts of your data to other MarkLogic clusters, whether they are in the same data center or geographically distributed (and possibly bandwidth or connectivity limited). Flexible Replication is different from Database Replication, as Database Replication is better suited for the purpose of high availability (for example, for failover and disaster recovery). Flexible Replication, on the other hand, is well-suited for applications that need to keep copies of subsets of their data for use by other applications.

In MarkLogic 8, Flexible Replication adds the ability to perform replication based on a saved query (an alert). This query-based flexible replication (QBFR) has the ability to be much faster for highly distributed systems to replicate subsets of their content and makes it efficient to maintain changes in that content. This is especially useful in applications where there are many replicas each replicating different parts of the data, and when some or all of those replicas might have bandwidth or connectivity constraints. For details on configuring QBFR, see Configuring Alerting With Flexible Replication in the *Flexible Replication Guide*.

3.11 Incremental Backup

In addition to the existing backup and journaled backup, MarkLogic 8 adds incremental backup, allowing you to create incremental backups at whatever cadence makes sense for your application. Incremental backups can save backup time and space because it only needs to back up the changes since the last full or incremental backup. You can combine incremental backups with journal archiving, allowing you to restore to the closest incremental backup and then rewind to any time using your journal archive. For details, see Incremental Backup and Restoring from an Incremental Backup with Journal Archiving in the *Administrator's Guide*.

3.12 Document Library Services (DLS) Improvements

Document Library Services (DLS) is an API that allows you to create applications to version documents, perform check-in and check-out operations, and other library services features. In MarkLogic 8, there are improvements to make the system significantly more efficient, especially if you have large DLS repositories.

The improvements require an upgrade operation on any existing DLS repositories, as described in "Document Library Services (DLS) Repositories Need To Perform A Bulk Upgrade Operation" on page 41 of these *Release Notes*.

For details on library services applications, see Library Services Applications in the *Application Developer's Guide*.

3.13 MLCP Enhancements

The MarkLogic Content Pump (MLCP) has many improvements in MarkLogic 8, including:

- The ability to specify an alternate database

- Temporal document support for loading
- JSON document support for loading

Additionally, because of the Enhanced HTTP Server Features, you no longer need to create an XDBC Server to use MLCP; you can target any REST API instance, including the built-in port 8000 instance.

3.14 New Features in MarkLogic 8.0-3

MarkLogic version 8.0-3 contains the following new features:

- Compatibility With New Versions of Hadoop
- MarkLogic Connector for Hadoop Supports JavaScript Queries
- XCC Support for JSON Content
- HDFS Client Bundles Available for Download
- Arithmetic and Comparison Methods for dateTime and Duration Objects in Server-Side JavaScript
- Expanded instanceof Support in Server-Side JavaScript
- New Features in the Client APIs

3.14.1 Compatibility With New Versions of Hadoop

MarkLogic features that leverage Apache Hadoop MapReduce and HDFS can now be used with the following Hadoop distributions:

- Cloudera's Distribution Including Apache Hadoop (CDH) version 5.4
- Hortonworks Data Platform (HDP) version 2.2

CDH version 4.3 is no longer included in the list of compatible distributions.

This change affects mlcp (MarkLogic Content Pump) in distributed mode, the MarkLogic Connector for Hadoop, and the use of HDFS for forest storage.

3.14.2 MarkLogic Connector for Hadoop Supports JavaScript Queries

You can now express input and output queries in either XQuery or Server-Side JavaScript. That is, you can use Server-Side JavaScript in the values of the configuration properties `mapreduce.marklogic.input.query` and `mapreduce.marklogic.output.query`. Previously, you could only use XQuery to express input and output queries. Use the properties `mapreduce.marklogic.input.queryLanguage` and `mapreduce.marklogic.output.queryLanguage` to indicate which scripting language is used in your input or output query. The default query language is XQuery.

For details, see the *MarkLogic Connector for Hadoop Developer's Guide*.

3.14.3 XCC Support for JSON Content

You can now use XCC and the Jackson libraries to insert, update, and read JSON content. For example, you can create a ContentFactory specifically for JSON, similar to the following:

```
JsonNode node = ...;
ContentCreateOptions options = new ContentCreateOptions();
options.setFormat(DocumentFormat.JSON);
Content content = ContentFactory.newContent(uri, node, options);
```

For details, see Working With JSON Content in the *XCC Developer's Guide*.

3.14.4 HDFS Client Bundles Available for Download

Previously, using HDFS for forest storage required you to assemble a set of Hadoop HDFS JAR files or install Hadoop on each MarkLogic host containing a forest on HDFS (or to install Hadoop in a well-known location).

You can now download a pre-packaged Hadoop HDFS client bundle from http://developer.marklogic.com/products/hadoop and install this bundle on your MarkLogic hosts. A bundle is available for each supported Hadoop distribution. Use of one of these bundles is required if you use HDFS for forest storage.

The availability of these bundles also changes how and where MarkLogic looks for the JDK and Hadoop libraries.

For details, see HDFS Storage in the *Query Performance and Tuning Guide*.

3.14.5 Arithmetic and Comparison Methods for dateTime and Duration Objects in Server-Side JavaScript

Beginning with 8.0-3, there are methods that allow you to compare, add, subtract, multiply, and divide duration and date objects in Server-Side JavaScript. These methods allow you to take advantage of richly typed date values available in MarkLogic from Server-Side JavaScript. For details of these new APIs, see JavaScript Duration and Date Arithmetic and Comparison Methods in the *JavaScript Reference Guide*.

3.14.6 Expanded instanceof Support in Server-Side JavaScript

Beginning in 8.0-3, you can use the `instanceof` operator in Server-Side JavaScript to test for any of the MarkLogic-typed values, including ValueIterator, cts.query, and so on. For details, see JavaScript instanceof Operator in the *JavaScript Reference Guide*.

3.14.7 New Features in the Client APIs

The REST Client API, Java Client API, and Node.js Client API now support deleting multiple documents by URI in a single operation.

- REST: Pass multiple `uri` parameters to the DELETE /v1/documents method.

- Java: Pass multiple URIs to `DocumentManager.delete`.
- Node.js: Pass multiple URIs to `Documents.remove` or `DatabaseClient.remove`.

The Java Client API now includes support for the `extract-document-data` query option on search operations. Use this option with `QueryManager.search` to include sparse document projections in your search results. Previously, this capability was only available for multi-document reads. For details, see Extracting a Portion of Matching Documents in the *Developing Applications With the Java Client API*.

3.15 New Features in MarkLogic 8.0-4

MarkLogic version 8.0-4 contains the following new features:

- Server-Side JavaScript Features
- Semantics Enhancements
- Management REST API Enhancements
- Geospatial Search Enhancements
- Support For Red Hat Enterprise Linux 7
- Clock Skew Timeouts
- Node.js Client API Supports Searching XML with QBE
- New Features in mlcp

3.15.1 Server-Side JavaScript Features

MarkLogic 8.0-4 includes the following Server-Side JavaScript new features:

- Server-Side JavaScript Search API (jsearch)
- cts:parse
- Alerting API JavaScript Enhancements
- Thesaurus and Spelling API Enhancements

3.15.1.1 Server-Side JavaScript Search API (jsearch)

MarkLogic 8.0-4 includes a new Server-Side JavaScript library to help create search applications. This new `jsearch` API uses common JavaScript design patterns to make it easy to create search applications that include search results with snippets, facets, suggestions, and other search application features. For details on the jsearch API, see Creating JavaScript Search Applications in the *Search Developer's Guide* and the *jsearch API Documentation*.

3.15.1.2 cts:parse

There is a new built-in function called `cts:parse` in XQuery and `cts.parse` in JavaScript. The `cts.parse` function is used by `jsearch`, but is also available to any XQuery or Server-Side JavaScript code. It returns a `cts:query` and it useful for parsing a Google-style search grammar that a user might type into a search box in an application, and converting that string into a cts:query to pass into a search. For details, see Creating a Query From Search Text With cts:parse in the *Search Developer's Guide*.

3.15.1.3 Alerting API JavaScript Enhancements

In 8.0-4, the alerting API is more convenient to use in JavaScript. The alerting API now allows you to use either XML or JSON format, and it accepts JavaScript objects when called from JavaScript.

It is a best practice to pass XML when the alert action is implemented by an XQuery module and a JavaScript object when the action is implemented by a JavaScript file.

3.15.1.4 Thesaurus and Spelling API Enhancements

In 8.0-4, the thesaurus and spelling APIs are more convenient to use in JavaScript. The spelling API now allows you to create dictionaries either in XML or JSON format, and both the thesaurus and spelling APIs accept JavaScript objects when called from JavaScript. For details, see Using the Spelling Correction Functions and Using the Thesaurus Functions in the *Search Developer's Guide*.

3.15.2 Semantics Enhancements

MarkLogic 8.0-4 includes the following enhancements to Semantics:

- SPARQL Negation
- Java Client API Semantics Enhancements
- MarkLogic Sesame API
- MarkLogic Jena API
- Semantics REST API Additions
- mlcp Enhancements

3.15.2.1 SPARQL Negation

SPARQL 1.1 Negation (using `EXISTS`, `NOT EXISTS`, and `MINUS`) is part of MarkLogic Semantics in 8.0-4. Used with the `FILTER` expression, negation operates on matching patterns to refine solution results. See Negation in Filter Expressions in the *Semantics Developer's Guide*.

3.15.2.2 Java Client API Semantics Enhancements

In 8.0-4, the Java Client API includes increased support for Semantics. You can use Java for managing graphs and triples, and accessing SPARQL query and SPARQL Update functionality in MarkLogic. MarkLogic now supports graph operations, SPARQL query, and SPARQL Update in the Java Client API. For more information, see Java Client API in the *Semantics Developer's Guide*, the *Developing Applications With the Java Client API*, and `GraphManager` and `SPARQLQueryManager` in the *Java Client API Documentation*. The Java Client project is available on GitHub.

3.15.2.3 MarkLogic Sesame API

In 8.0-4, MarkLogic Sesame API provides full-featured support for standard Sesame APIs. Java developers familiar with Sesame APIs now have access to MarkLogic Semantics, extensions, and combination queries, simplifying semantic application development. For more information, see MarkLogic Sesame API in the *Semantics Developer's Guide* and the Sesame project on GitHub.

3.15.2.4 MarkLogic Jena API

In 8.0-4, MarkLogic Jena API provides full-featured support for standard Jena APIs. Java developers familiar with Jena APIs now have access to MarkLogic Semantics, extensions, and search capabilities, simplifying semantic application development. For more information, see MarkLogic Jena API in the *Semantics Developer's Guide* and the Jena project on GitHub.

3.15.2.5 Semantics REST API Additions

The ability to use MarkLogic Semantics with the REST Client API to view, query, and modify triple data and graphs has been enhanced in 8.0-4 with variable bindings, ruleset configuration, and transaction support. For details, see Using Semantics with the REST Client API in the *Semantics Developer's Guide*.

3.15.2.6 mlcp Enhancements

When importing triples and quads with the mlcp command line tool, you can now use the new options `-output_graph` and `-output_override_graph` to control the graphs into which your semantic data is loaded. For details, see Loading Triples in the *mlcp User Guide*.

3.15.3 Management REST API Enhancements

The REST Management API has been expanded to include new alerting, mimetypes, and support endpoints.

3.15.4 Geospatial Search Enhancements

MarkLogic 8.0-4 introduces the following enhancements for working with and searching geospatial data:

- Converting To/From Common Representations
- New Geospatial Utility Functions

- Support for More Recent KML and GML Versions

3.15.4.1 Converting To/From Common Representations

MarkLogic 8.0-4 adds support for Well Known Binary (WKB) representation of geospatial data, as well as new functions for converting between common geospatial data serializations and the internal MarkLogic representation. For details, see Converting To and From Common Geospatial Representations in the *Search Developer's Guide* and the XQuery and Server-Side JavaScript API reference documentation.

The following new functions have been added to support this feature:

XQuery	JavaScript
`geo:parse`	`geo.parse`
`geo:validate-wkt`	`geo.validateWkt`
`geo:parse-wkb`	`geo.parseWkb`
`geo:to-wkb`	`geo.toWkb`
`geo:validate-wkb`	`geo.validateWkb`
`geogml:parse-gml`	`geogml.parseGml`
`geogml:to-gml`	`geogml.toGml`
`geokml:parse-kml`	`geokml.parseKml`
`geokml:to-kml`	`geogml.toKml`
`georss:parse-georss`	`georss.parseGeorss`
`georss:to-georss`	`georss.toGeorss`
`geojson:parse-geojson`	`geojson.parseGeojson`
`geojson:to-geojson`	`geojson.toGeojson`

3.15.4.2 New Geospatial Utility Functions

The following new geospatial utility functions have been added in 8.0-4. For details, see the XQuery and Server-Side JavaScript API Reference.

XQuery	JavaScript
`geo:polygon-to-linestring`	`geo.polygonToLinestring`
`geo:linestring-reverse`	`geo.linestringReverse`
`geo:linestring-concat`	`geo.linestringConcat`
`geo:region-affine-transform`	`geo.regionAffineTransform`
`geo:region-approximate`	`geo.regionApproximate`
`geo:region-clean`	`geo.regionClean`
`geo:circle-polygon`	`geo.circlePolygon`
`geo:remove-duplicate-vertices`	`geo.removeDuplicateVertices`
`geo:count-vertices`	`geo.countVertices`
`geo:count-distinct-vertices`	`geo.countDistinctVertices`
`geo:interior-point`	`geo.interiorPoint`
`geo:distance-convert`	`geo.distanceConvert`
`cts:match-regions`	`cts.matchRegions`

3.15.4.3 Support for More Recent KML and GML Versions

MarkLogic now supports KML 2.2 and GML 3.2. These are now the default versions for KML and GML data. Use the following namespace URIs in your data to identify the version and when converting between a cts point or region and a GML or KML node.

- KML 2.0: http://earth.google.com/kml/2.0
- KML 2.2: http://www.opengis.net/kml/2.2
- GML 3.1.1: http://www.opengis.net/gml
- GML 3.2: http://www.opengis.net/gml/3.2

When generating GML or KML constructs from a cts point or region, you can use the above namespace URIs in conversion functions like the XQuery `geogml:to-gml` function or JavaScript `geogml.toGml` function to request a specific version.

As a side-effect of this feature, the GML and KML geospatial library modules have been moved to a MarkLogic-specific namespace. For details, see “Geospatial Namespace and Data Version Changes” on page 53.

3.15.5 Support For Red Hat Enterprise Linux 7

MarkLogic 8.0-4 includes support for RHEL 7. The RHEL 7 package is separate from the RHEL 6 package on developer.marklogic.com. For details on MarkLogic platforms, see "Supported Platforms" on page 9, and for details on installation see the *Installation Guide*.

3.15.6 Clock Skew Timeouts

XDQP connections between hosts in either a local or foreign cluster will now drop if a host's clock is skewed by more than the host timeout. Attempts to connect will result in a warning message in the log when the first connection attempt is rejected, and every hour after that.

The host timeout is either from the host's group if it is an intra-cluster connection, or from the foreign cluster configuration if it is an inter-cluster connection.

3.15.7 Node.js Client API Supports Searching XML with QBE

You can now search XML documents with using Query By Example when using the Node.js Client API. For details, see Querying XML Content With QBE in the *Node.js Application Developer's Guide*.

3.15.8 New Features in mlcp

The mlcp command line tool added the following new capabilities in MarkLogic 8.0-4:

- Improved support for loading semantic triples and quads through the new `-output_graph` and `-output_override_graph` options. For details, see Loading Triples in the *mlcp User Guide*.
- The new `-query_filter` option enables you to select documents to export or copy using a cts query. For details, see Controlling What is Exported, Copied, or Extracted in the *mlcp User Guide*.
- Create JSON documents from delimited text files, such as CSV files, by using `-document_type json` with `-input_file_type delimited_text`. For details, see Creating Documents from Delimited Text Files in the *mlcp User Guide*.
- Create JSON documents from aggregate JSON files (one JSON object per line). For details, see Creating Documents from Line-Delimited JSON Files in the *mlcp User Guide*.
- Generate a report of the mlcp version, supported MarkLogic versions, and dependent software versions found in your environment using the new "version" command. For example:

```
$ mlcp.sh version
ContentPump version: 8.0
Java version: 1.7.0_45
Hadoop version: 2.6.0
Supported MarkLogic versions: 6.0 - 8.0
```

- Improved error reporting and job output summary. For details, see Understanding mlcp Output in the *mlcp User Guide*.

3.16 New Features in MarkLogic 8.0-6

MarkLogic 8.0-6 includes the following new features:

- MapR Support in mlcp and the Hadoop Connector
- New Server-Side Transaction Controls

3.16.1 MapR Support in mlcp and the Hadoop Connector

You can now use MapR as your Hadoop implementation with `mlcp` in distributed mode and with the MarkLogic Connector for Hadoop.

Using mlcp with MapR requires special setup. For details, see Required Software and Using mlcp With MapR in the *mlcp User Guide* or .

3.17 New Features in MarkLogic 8.0-7

MarkLogic 8.0-7 includes the following new features:

- Ability to Access and Modify Metadata in a Transformation
- New Server-Side Transaction Controls
- XCC: New Session Methods for Transaction Control
- mlcp: Greater Control over Host Connections
- REST Management API: Advancing LSQT
- REST Client API: Advancing LSQT

3.17.1 Ability to Access and Modify Metadata in a Transformation

Collections, permissions, document quality, and temporal collection metadata specified by the client are now available to a transformation function via the context parameter. In addition, a transformation function can set collections, permissions, quality, and temporal collection metadata for its output document(s).

For details, see Transforming Content During Ingestion in the *mlcp User Guide*.

3.17.2 New Server-Side Transaction Controls

As of MarkLogic 8.0-7, you can configure the commit mode (auto or explicit) and transaction type (query, update, or auto) independently when configuring a new transaction. This change manifests in the following ways:

- The `xdmp:update` XQuery prolog option accepts a new value, “auto”, which specifies that MarkLogic should determine the transaction/statement type (query or update) through static analysis. The pre-existing value “false” now means the transaction/statement type is query. Use this option plus `xdmp:commit` instead of the now-deprecated `xdmp:transaction-mode` prolog option.
- A new XQuery prolog option, `xdmp:commit`, has been added for specifying whether a main module should run as a single-statement, auto-commit transaction (the default) or a multi-statement explicit-commit transaction. Use this option plus `xdmp:update` instead of the now-deprecated `xdmp:transaction-mode` prolog option.
- New `commit` and `update` options have been added to the functions listed in the table below. Use these in preference to the `transaction-mode option`, which has been deprecated.

The following functions support the new commit and update options. For more details, see the function reference documentation for `xdmp:eval` (XQuery) or `xdmp.eval` (JavaScript).

XQuery	JavaScript
`xdmp:eval`	`xdmp.eval`
`xdmp:javascript-eval`	`xdmp.xqueryEval`
`xdmp:invoke`	`xdmp.invoke`
`xdmp:invoke-function`	`xdmp.invokeFunction`
`xdmp:spawn`	`xdmp.spawn`
`xdmp:spawn-function`	

For more details on the new capabilities, see Understanding Transactions in MarkLogic Server in the *Application Developer's Guide*, and xdmp:update and xdmp:commit in the *XQuery and XSLT Reference Guide*.

For details on transitioning from the old transaction controls to the new ones, see the following topics:

- “Changes to xdmp:update XQuery Prolog Option” on page 49
- “xdmp:transaction-mode XQuery Prolog Option Deprecated” on page 61
- “Deprecation of transaction-mode Option to xdmp:eval” on page 62

3.17.3 XCC: New Session Methods for Transaction Control

In MarkLogic 8.0-7, the following methods have been added to the `Session` class for configuring transactions and querying transaction configuration:

- `Session.setAutoCommit` and `Session.isAutoCommit`
- `Session.setUpdate` and `Session.getUpdate`

You should use these methods rather `Session.setTransactionMode`, which has been deprecated. For details, see "XCC Session.setTransactionMode is Deprecated" on page 62.

`Session.setAutoCommit` controls whether requests submitted during the session run in a transaction with auto-commit semantics (the default) or explicit commit semantics. Executing a request with commit set to explicit starts a multi-statement transaction.

`Session.setUpdate` controls whether requests submitted during the session run in a query transaction, an update transaction, or if the transaction type should be automatically detected by MarkLogic through analysis of the submitted code. Auto detection is the default behavior.

Note that if you override the `Session` transaction configuration in an ad hoc query, the behavior differs depending on whether you configure the session using `setTransactionMode` or `setAutoCommit` and `setUpdate`. With `setAutoCommit` and `setUpdate`, the transaction configuration reverts to the `Session` settings once the transaction involving the override completes. With `setTransactionMode`, the override persists and affects future transactions unless you explicitly change it.

3.17.4 mlcp: Greater Control over Host Connections

You can now specify multiple hosts for mlcp to connect to during `import`, `export`, and `copy` jobs. Used by itself, this feature enables mlcp to fall back an alternative host if the initial host is not available.

You can also use this capability in conjunction with the new `-restrict_hosts` option to prevent mlcp from connecting to any hosts except the ones on the initial host list.

For more details, see Controlling How mlcp Connects to MarkLogic in the *mlcp User Guide*.

3.17.5 REST Management API: Advancing LSQT

You can use the following new method of the REST Management API to advance LSQT on a temporal collection:

```
POST /manage/v2/databases/{id|name}/temporal/collections?collection=collname
```

For more details, see `POST:/manage/v2/databases/{id|name}/temporal/collections?collection={name}` in the *MarkLogic REST API Reference*.

3.17.6 REST Client API: Advancing LSQT

You can now use the following new method of the REST Client API to advance LSQT on a temporal collection:

```
POST /v1/temporal/collections/{name}
```

For more details, see `POST:/v1/temporal/collections/{name}` in the *MarkLogic REST API Reference*.

3.18 New Features in MarkLogic 8.0-8

MarkLogic 8.0-8 introduces the following new features:

- Separate Download and Installation of Converters and Filters
- SQL Deep Analysis Trace Event

3.18.1 Separate Download and Installation of Converters and Filters

In MarkLogic version 8 releases starting at 8.0-8, MarkLogic converters/filters are offered as a separate package (called *MarkLogic Converters* package) from MarkLogic Server package.

This change provides better flexibility and enables you to install/uninstall MarkLogic converters/filters separately from MarkLogic Server.

For more details, see MarkLogic Converters Installation Changes in Version 8 Releases Starting at 8.0-8 in the *Installation Guide*.

3.18.2 SQL Deep Analysis Trace Event

SQL uses cost-based optimization. With the default (shallow) table analysis, the cost estimates are likely to be very wrong, certain optimizations are disabled, and query executions will be longer. However, the cost of performing the deep analysis is expensive.

For long-lived ODBC connections, the cost of analysis is paid once when the connection is created and the better costing numbers will be available for the life time of the connection. The trace event SQL Deep Analysis causes the deep analysis to be run when the connection is initiated automatically. For `xdmp:sql`, deep analysis is only done if the `ANALYZE` command is used in the SQL, but the information is only kept until `xdmp:sql` returns. For example:

```
xdmp:sql("ANALYZE; SELECT whatever FROM example")
```

Enabling SQL Deep Analysis will enable full table analysis for each new ODBC connection.

Note: With shallow table analysis, a number of key optimizations are defeated and the costing functions may lead to very bad plans. Customers who use long-running ODBC connections will probably want to enable the SQL Deep Analysist trace event.

4.0 Known Incompatibilities with Previous Releases

The vast majority of applications implemented on MarkLogic Server 5.0-* will run either without modifications or with very minor modifications in MarkLogic 8. There are, however, a number of changes that will cause compatibility issues with applications developed using previous releases. This section describes those incompatibilities and includes the following topics:

- JSON Related Incompatibilities
- Semantics Incompatibilites
- REST and Java Client API Incompatibilites
- Document Library Services (DLS) Repositories Need To Perform A Bulk Upgrade Operation
- Linux Now Requires Red Hat 6
- mlsql On Linux No Longer Ships With Server
- Cyrillic Tokenization Changes
- Application Builder Applications Must Be Re-Deployed in MarkLogic 8
- Application Builder and Information Studio Links Removed
- Search API Incompatibilities
- Locks and Properties Query Built-In Functions Renamed
- xdmp:uri-content-type Of an XML Document Now Returns application/xml, Can Affect CPF Applications
- xdmp:function Signature Change
- Incompatibilities Between 8.0-9 and 8.0-8
- Incompatibilities Between 8.0-8 and 8.0-7
- Incompatibilities Between 8.0-6 and 8.0-7
- Incompatibilities Between 8.0-5 and 8.0-6
- Incompatibilities Between 8.0-3 and 8.0-4
- Incompatibilities Between 8.0-2 and 8.0-3
- Incompatibilities Between 8.0-1 and 8.0-2
- Planning for Future Upgrades
- MarkLogic 7 Incompatibilites
- Incompatibilities Between MarkLogic 6 and 5.0

4.1 JSON Related Incompatibilities

MarkLogic 8 includes Native JSON support. In MarkLogic 7, there was support for JSON via a set of libraries to convert between JSON and XML. If you are using the MarkLogic 7 JSON support, you will have to migrate your code to use the native JSON support. This should end up being more efficient, but will require you to do some minor code changes. This section lists the incompatibilites related to working with JSON documents:

- Documents Created as JSON With MarkLogic 7 REST API or MLCP Must Be Converted to Native JSON
- json:unquotedString Primitive No Longer Available
- xdmp:to-json and json:transform-to-json Now Returns a document-node()
- Search, Java, REST: json-key Is Now json-property in Options and Structured Query
- Java and REST: Default Path Language for JSON Document Patches is Now XPath
- Java and REST: Specifying a Language for JSON Documents is Deprecated
- Java and REST: New Restrictions on Patching JSON Content
- Java and REST: Transforms and Extensions That Manipulate JSON Must Be Rewritten
- Java and REST: JSON Array Items and Property Values No Longer Distinguishable in QBE
- Field Range Query and Field Value Query on JSON May Behave Differently

4.1.1 Documents Created as JSON With MarkLogic 7 REST API or MLCP Must Be Converted to Native JSON

In previous versions of MarkLogic, you can load JSON documents into MarkLogic using either the REST Client API or the Java Client API. When you do so, the documents are transformed and stored as XML, but are still queryable as and returned as JSON. In MarkLogic 8 this "XML facade" is no longer needed, and the REST and Java Client APIs in MarkLogic 8 do not do the translation to the XML facade anymore. Therefore, if you have any document that were loaded as JSON in MarkLogic 7 and earlier, you must convert those document to native JSON in order to query them as JSON using the REST API.

To help with the conversion, MarkLogic supplies a set of conversion scripts. These scripts do not handle every case, and for any case it does not handle you will have to convert the documents some other way. Because the conversion scripts do not handle all cases, it is very important to do a backup of your database before attempting the conversion. The scripts are located in the `Samples/migrate-scripts` directory under the MarkLogic installation directory (`/opt/MarkLogic` on Linux, `c:/Program Files/MarkLogic` on Windows, and `~/Library/MarkLogic` on Mac OS). The scripts require bash and curl, and on Windows platforms they also require cygwin.

For more details, see the *<marklogic-dir>*`/Samples/migrate-scripts/README` file.

Generally, the conversion scripts perform the following:

- Converts the documents to native JSON.
- Updates index configurations to reference the JSON content.
- Updates existing saved search options. This includes changing references to the JSON basic namespace to reference the new JSON content and changes occurrences of `json-key` to `json-property`.
- Updates any alerting rules that reference the JSON content.

The scripts do not upgrade your application code. The types of things you will need to change in your application include:

- Modify client code to update structured queries, combined queries, and query options that reference `json-key` and anything in the `http://marklogic.com/xdmp/json/basic` namespace. For more details see "Search, Java, REST: json-key Is Now json-property in Options and Structured Query" on page 31.
- Rewrite and reinsert any server-side code (for example, transformations or custom constraints) that operated over the internal XML representation of your JSON documents. For details see "Java and REST: Transforms and Extensions That Manipulate JSON Must Be Rewritten" on page 33.
- Modify client code that relies on the `/v1/keyvalues` endpoint for key/value searches over JSON.
- Modify client code to update patch specifications over JSON. For details see "Java and REST: New Restrictions on Patching JSON Content" on page 33.
- Review index settings and queries over document properties, update as needed (because you can no longer have JSON properties).

The basic steps to upgrade your MarkLogic 7 or earlier JSON to native JSON in MarkLogic 8 are as follows:

1. Backup the database in which your JSON documents exist.

2. Make copies and edit the connection details and other information in the various configuration files in the `Samples/migrate-scripts/conf` directory. This files have details about your configuration and index settings.

3. Run the `Samples/migrate-scripts/migrate` script.

4. Test your results. Make sure the index changes that the scripts made match your newly converted JSON data. It is especially important to review path and fields indexes to make sure they are including the same content in the converted JSON is they were previously.

If you have problems upgrading your application, contact MarkLogic Technical Support.

4.1.2 json:unquotedString Primitive No Longer Available

MarkLogic 7 had a primitive to convert an XQuery string to an unquoted String called `json:unquotedString`. In MarkLogic 8, that function is no longer available because it is no longer needed, as MarkLogic 8 has much more extensive support for JSON. For details on working with JSON in MarkLogic, see Working With JSON in the *Application Developer's Guide*.

4.1.3 xdmp:to-json and json:transform-to-json Now Returns a document-node()

In MarkLogic 8, the `xdmp:to-json` function returns a `document-node()`; previously, it returned a string. If you have code that expects a JSON string, you might need to modify your code to perform XPath on the `document-node()` to get the JSON node (which will serialize into a string); depending on what your code does, you might or might not need to do this. For example:

```
(: 7.0 :)
xdmp:to-json(("a",fn:false()))
=> ["a",false]

(: 8.0 :)
xdmp:to-json(("a",fn:false()))/node()
=> ["a",false]
```

Note: The `json:transform-to-json` function uses `xdmp:to-json`, so it also returns a `document-node()` in MarkLogic 8.

4.1.4 Search, Java, REST: json-key Is Now json-property in Options and Structured Query

All occurrences of `json-key` in query options and structured query are now `json-property`. If you use the constructs listed below to search JSON documents by key/property name, you will need to modify your query options (`search:options`, in XML) or structured queries.

The following query options are affected. For details, see `search:search` or Appendix: Query Options Reference in the *Search Developer's Guide*.

- `container-constraint`
- `extract-metadata`
- `range-constraint`
- `sort-order`
- `value-constraint`
- `word-constraint`

The following structured query components are affected. For more details, see the structured query Syntax Reference in the *Search Developer's Guide*.

- `container-query`
- `range-query`
- `value-query`
- `word-query`

The corresponding Java Client API structured query builder method name has also changed. `StructuredQueryBuilder.JSONKey` is now `StructuredQueryBuilder.JSONProperty`.

4.1.5 Java and REST: Specifying a Language for JSON Documents is Deprecated

Previously, you could specify a language when ingesting JSON documents. This was only possible because JSON documents were represented internally as XML.

This parameter is now deprecated and will be ignored when present. This affects the following interfaces:

- Java: `JSONDocumentManager.setLanguage` and `JSONDocumentManager.getLanguage` are deprecated. Calling `setLanguage` has no effect.
- REST: The lang request parameter of `PUT:/v1/documents` is deprecated and will be ignored.
- REST: The lang request parameter of `POST:/v1/documents` (all variants) is deprecated and will be ignored.

4.1.6 Java and REST: Default Path Language for JSON Document Patches is Now XPath

This section applies to applications that use the Java Client API or REST Client API patch (partial update) feature on JSON documents.

Previously, JSONPath was the default path language for identifying the target of update operations in a JSON patch. XPath is now the default path language for both XML and JSON patches.

Use of JSONPath is now deprecated. To convert your JSON patches to use XPath expressions instead of JSONPath, see Traversing JSON Documents Using XPath in the *Application Developer's Guide*.

To continue using JSONPath, you can explicitly override the default path language in one of the following ways:

- For a raw JSON patch, include a `pathlang` property as the sibling of the top level `patch` property. For example:

  ```
  { "pathlang": "jsonpath",
    "patch": [ ... ] }
  ```

  Raw patches are used by `POST:/v1/documents` and can be used with the Java Client API method `DocumentManager.patch`.

- When using the Java Client API, use `DocumentPatchBuilder.pathLanguage`. to set the path language to JSONPath, as shown in the following example:

```
DocumentPatchBuilder patchBldr = docMgr.newPatchBuilder();
patchBldr.pathLanguage(PathLanguage.JSONPATH);
```

4.1.7 Java and REST: New Restrictions on Patching JSON Content

The native JSON document model imposes some new restrictions on partial updates to JSON documents. Therefore, some patch operations that were previously supported will now be rejected or produce different results. For details, see Limitations of JSON Path Expressions in the *REST Application Developer's Guide* and Traversing JSON Documents Using XPath in the *Application Developer's Guide*.

For example, you cannot construct patch path expressions that address anonymous nodes. In the JSON document model, object nodes and array nodes are anonymous. The name in a property name-value pair addresses the value(s), not the containing node.

This means you cannot use "last-child" position to insert a new property or value under the root node of a document or in an array because the parent node is anonymous and cannot be selected by the context expression of the insert operation. Similarly, you cannot address an array node that is nested inside another array (`[1, [2, 3], 4]`) because it is unnamed.

You can no longer replace an entire property (name-value pair) in a single patch `replace` operation for the same reason. To replace a property, you must delete it and then insert a new one.

4.1.8 Java and REST: Transforms and Extensions That Manipulate JSON Must Be Rewritten

This section applies to REST Client API and Java Client API applications that use content transformations, resource service extensions, and other server-side code to manipulate the XML representation of JSON documents.

Previously, content transformations, resource service extensions, and other server-side code manipulated JSON content as XML in the `http://marklogic.com/xdmp/json/basic` namespace. Now, such code must operate on JSON document nodes instead of XML.

For example, previously, the JSON data { "key": "value" } was represented in the database as XML of the following form, and this is what your server-side transforms and extensions worked with:

```
<json type="object" xmlns="http://marklogic.com/xdmp/json/basic">
  <key type="string">value</key>
</json>
```

Thus, to access the value of the “key” property in XQuery, you could use a path expression like this following:

```
$someDocument/json:json/json:key
```

With a native JSON document you reference the same data using the following path expression:

```
$someDocument/key
```

You should understand the native JSON document model before rewriting your code. For details, see Working With JSON in the *Application Developer's Guide*.

4.1.9 Java and REST: JSON Array Items and Property Values No Longer Distinguishable in QBE

This change may affect applications that use QBE to search JSON documents using the Java Client API or REST Client API.

Previously, you could construct a QBE that included a word or value query explictly scoped to an array item or the . Now, it is not possible to distinguish between an array item and a property value.

For example, given a document with the following context:

```
{ "notArray": "blue", "array": ["azure", "blue"] }
```

Previously, the following QBE would only match the occurence of the value `"blue"` in the `"array"` property. Now it matches both the occurrence in `"array"` and the occurrence in `"notArray"`.

```
{"$query": [ {"$value": ["blue"]} ] }
```

This is because array nodes are unnamed in the native JSON document model, so they cannot be explicitly identified in a QBE.

4.1.10 Field Range Query and Field Value Query on JSON May Behave Differently

This difference only applies to applications using field range queries or field value queries on JSON documents.

JSON and XML are not indexed in exactly the same way. Some of the indexing differences affect the behavior of field range queries and field value queries over JSON. Since JSON documents were previously stored as XML, this means your field range queries and field value queries over JSON may behave differently.

For example, previously you could construct a field value query for “John Smith” that would match the following document by defining a field on the `name` property that excluded the `middle` property.

```
{ "name": {
    "first": "John",
    "middle": "NMI",
    "last": "Smith"
}
```

This was possible because the document was represented as XML and the text nodes in the field were concatenated together so that the field value in the above document was “John Smith”. In native JSON documents, this concatenation does not occur, and the values of the equivalent field are “John” and “Smith”. To get the same effect now, you would have to use a construct such as a near query.

For more details, see Creating Indexes and Lexicons Over JSON Documents and How Field Queries Differ Between JSON and XML in the *Application Developer's Guide*.

4.2 Semantics Incompatibilites

MarkLogic 8 introduces a number of new and changed Semantic features. This section describes those and includes the following changes that might cause incompatibilites:

- Changed Function: sem:sparql
- Changed Function: sem:sparql-values
- Changed Function: sem:sparql-values
- Deprecated Function: sem:sparql-triples
- Changed Behavior: Graphs

4.2.1 Changed Function: sem:sparql

In MarkLogic 8, the signature of `sem:sparql` has changed for the last parameters. The fourth parameter is now a `sem:store*`, where previously there where two parameters at the end, one to specify a `cts:query` and another to specify the forest ID. The old signature is still available, but is deprecated. If you have any code from MarkLogic 7 that uses the fourth or fifth arguments to `sem:sparql`, it will still work in MarkLogic 8, but you should migrate that code to use the new signature using `sem:store` as the fourth parameter.

4.2.2 Changed Function: sem:sparql-values

In MarkLogic 8, `sem:sparql-values` now serializes a string as a `cts:word-query` when used as part of an argument. In other words, string values will be passed as `cts:query` arguments. This is a change from MarkLogic 7.

4.2.3 Changed Function: sem:sparql-values

The `sem:sparql-values` function no longer accepts `forest-id` as an option. This is an incompatibility with MarkLogic 7 functionality.

4.2.4 Deprecated Function: sem:sparql-triples

The `sem:sparql-triples` function has been deprecated in favor of `sem:in-memory-store` in MarkLogic 8. See documentation for details - Querying Triples in Memory in the *Semantics Developer's Guide*.

4.2.5 Changed Behavior: Graphs

In MarkLogic 8, graph documents containing metadata are created when triples are ingested, whether they are ingested using SPARQL Update, mlcp, or SPARQL endpoints over REST. These named graphs inherit the permissions of the user, unless specified as part of the ingest process. The graph permissions are stored along with other metadata in the graph document.

When loading triples with mlcp, if the `output-permissions` parameter is set when loading RDF, the graph will inherit the default permissions just as it would in a `sem:sparql-update` operation. If the `output_collection` parameter is set when loading RDF, the graph document is only created for the first collection specified (because a managed triple can only belong to one graph).

In an upgraded system, graph metadata for managed triples created in MarkLogic 7 will be created the first time you add triples to the graph or modify triples in the graph. The graph will either have the user's permissions or the permissions specified as part of the operation. Make sure the document permissions of documents containing managed triples created in MarkLogic 7 are passed into `sem:sparql-update` as `default-permissions` to ensure the graph metadata created is consistent with the permissions for existing managed triples created in MarkLogic 7.

4.3 REST and Java Client API Incompatibilites

This section covers the incompatibilites for the REST API between MarkLogic 7 and MarkLogic 8. that are not related to JSON. If you work with JSON documents using the REST Client API or Java Client API, you should also see "JSON Related Incompatibilities" on page 28.

This section covers the following incompatibilities:

- Must Upgrade to Java Client API v3.0
- REST API Instance Must Use the Declarative Rewriter on the App Server
- Default Transaction Mode for the POST Method of Resource Service Extensions is Now Query
- REST API: Empty Bulk Read by Query Now Returns 200 Status
- Error Reporting Format and Detail Changes
- Deprecated Interface: Keyvalue Queries
- Transaction ID Format Has Changed

- A Transaction Can No Longer Be Shared Across Users
- Java: QBE Search Results No Longer Automatically Match the Query Format

4.3.1 Must Upgrade to Java Client API v3.0

You cannot use earlier versions of the Java Client API with MarkLogic 8. Update your application to use version 3.0 or later.

4.3.2 REST API Instance Must Use the Declarative Rewriter on the App Server

In MarkLogic 7, App Servers that are REST API Instances used a different URL rewriter than in MarkLogic 8. In MarkLogic 8, the App Server is configured to use the declarative rewriter. If you had not modified anything in your REST API Instance App Server setup, the upgrade to MarkLogic 8 will reconfigure your App Server to use the new rewriter. If, however, you have modified something in setup to use a different rewriter, then you will have to make similar changes to the new setup (or consider not using those changes in the REST API Instance). For details on the declarative rewriter, see Creating a Declarative XML Rewriter to Support REST Web Services in the *Application Developer's Guide*.

4.3.3 Default Transaction Mode for the POST Method of Resource Service Extensions is Now Query

In MarkLogic 7, the POST method of a resource service extension was always executed in update mode. In MarkLogic 8, POST methods in a single-statement transaction are executed in query mode. However, within a multi-statement transaction, they are in update mode. The new way is safer because, generally speaking, if a function is not doing an update, it is much more efficient for it to run as a query.

If you have a resource service extension that requires the transaction mode to be update, you need to modify the extension code to add an annotation to the function to explicitly force it into update mode. For example, to modify an extension that is a GET, add an annotation like the following to your get function:

```
declare %rapi:transaction-mode("update") function testrstxn:get(
   $context, $params) {
<Your code goes here>
};
```

The annotation `%rapi:transaction-mode("update")` forces the function to run as an update. For details, see Controlling Transaction Mode in the *REST Application Developer's Guide*.

4.3.4 REST API: Empty Bulk Read by Query Now Returns 200 Status

Previously, performing a bulk read by retrieving all documents that match a query would return a 404 Not Found response status if no documents matched the query. As of MarkLogic 8, such a request returns a 200 OK status with an empty response body.

This change applies to the following methods, when the Accept header is set to multi-part/mixed and the request does not ask for a search result summary in addition to the matching documents.

- `GET:/v1/search`
- `POST:/v1/search`
- `GET:/v1/qbe`
- `POST:/v1/qbe`

For more details on these interfaces, see Reading Multiple Documents Matching a Query in the *REST Application Developer's Guide*.

4.3.5 Error Reporting Format and Detail Changes

The changes in this section might affect your application if either of the following is true:

- Your REST or Java client application directly manipulates error details returned by MarkLogic through a REST API instance. This is unlikely for Java applications because the receive such errors as Java exceptions.
- Your application includes content transformations or resource service extensions that report errors to the client.

MarkLogic 8 introduces the following incompatible changes to error reporting for users of the

- Error Format Defaults to JSON and is a REST Instance Creation Property
- Error Detail Element and Property Names Have Changed
- Use RESTAPI-SRVEXERR to Report Errors from Transforms and Extensions

4.3.5.1 Error Format Defaults to JSON and is a REST Instance Creation Property

In previous versions, the default format for error messages returned by the REST Client API was XML, and you could change it by setting the `error-format` instance configuration property. As of MarkLogic 8, the default error format for new REST instances is JSON. You can now specify the format when you create the REST instance, and you can subsequently change it using the Admin Interface, `admin:appserver-set-default-error-format`, or the REST Management API.

This change has the following implications:

- The Java `setErrorFormat` and `getErrorFormat` methods of `com.marklogic.client.admin.ServerConfigurationManager` have been removed. Set the error message format when creating the REST instance instead.
- The REST GET and PUT `/v1/config/properties/error-format` methods are no longer available. Set the error message format when creating a REST instance instead.

- You cannot include an `error-format` XML element or JSON property in the payload to `PUT:/v1/config/properties`.
- The payload for `POST:/v1/rest-apis` can now include an `error-format` XML element or JSON property. This is an attribute of the App Server.
- You can use the Accept or X-Error-Accept HTTP headers to override the default error format for a particular request. For details, see Error Reporting in the *REST Application Developer's Guide*.

To set the error format when creating an instance, set the `error-format` configuration property. For details, see Creating an Instance in the *REST Application Developer's Guide*.

4.3.5.2 Error Detail Element and Property Names Have Changed

The error detail returned by the REST Client API has changed in the following ways. For examples of the new format, see Error Reporting in the *REST Application Developer's Guide*.

- XML: The root element of the error detail is an `<error-response>` element in the namespace `http://marklogic.com/xdmp/error`. Previously, it was an `<error/>` element in the namespace `http://marklogic.com/rest-api`.
- JSON: The top level property name is now `errorResponse`. Previously, it was `error`. Child property names that previously used dashes to separate "words" now use camel case. For example, `message-code` is now `messageCode` and `status-code` is now `statusCode`.

4.3.5.3 Use RESTAPI-SRVEXERR to Report Errors from Transforms and Extensions

Resource service extensions and transformations previously reported errors to the client using `RESTAPI-EXTNERR` and could specify a response payload in JSON or XML. The expected error response content type was controlled by a caller supplied parameter. This parameter is now ignored, and you should use `RESTAPI-SRVEXERR` instead of `RESTAPI-EXTNERR`. Your payload must be compatible with the MIME type expected by the caller, which can vary, so it is best to restrict the error detail to text. For details, see Reporting Errors in the *REST Application Developer's Guide*.

4.3.6 Deprecated Interface: Keyvalue Queries

This topic applies to applications that use the Java Client API class `KeyValueQueryDefinition` or the REST Client API method `GET:/v1/keyvalue`. These interfaces are now deprecated.

You can use Query By Example (QBE) or structured query to perform the same kind of search.

For example, to search for a JSON property named "author" with the value "Mark Twain", use a QBE such as the following:

```
{ "$query": { "author": "Mark Twain" } }
```

The following is a similar search for an XML element:

```
<q:qbe xmlns:q="http://marklogic.com/appservices/querybyexample">
  <q:query>
    <author>Mark Twain</author>
  </q:query>
</q:qbe>
```

With structured query, use `value-query` or `container-query`.

For details, see the following references:

- Searching Using Query By Example in the *Search Developer's Guide*.
- Searching Using Structured Queries in the *Search Developer's Guide*.
- `RawQueryByExampleDefinition` or `StructuredQueryBuilder` in the Java Client API javadoc.
- `GET:/v1/search` or `POST:/v1/search` in the *MarkLogic REST API Reference*.

4.3.7 Transaction ID Format Has Changed

Previously the transaction ids created using `DatabaseClient.openTransaction` (Java) or `POST:/v1/transactions` (REST) were of the form `hostId_transactionId`. The `hostId` segment has now been dropped.

As long as your application treats the transaction id as a "black box", this change is transparent.

4.3.8 A Transaction Can No Longer Be Shared Across Users

This change only affects Java Client API and REST Client API applications that use multi-statement transactions and share the resulting transaction id across multiple users.

Previously, it was possible to create a multi-statement transaction as one MarkLogic user and then perform operations within the transaction as another user. This is no longer possible. Now, all operations within a transaction must be performed as the same user who created the transaction.

4.3.9 Java: QBE Search Results No Longer Automatically Match the Query Format

Previously, using `QueryManager.search` with a Query By Example (QBE) automatically returned results in the same format as the query. That is, XML results were returned for an XML QBE, and JSON results were returned for a JSON QBE. Now, you must explicitly request JSON.

For example, if the `qbe` variable in the following statement contains a JSON QBE, then previously you would receive JSON results. Now, you will receive XML by default, instead.

```
queryMgr.search(qbe.newStringHandle()).get();
```

To achieve the same result as before, explicitly set the format on the result handle to JSON. For example:

```
queryMgr.search(qbe.newStringHandle().withFormat(Format.JSON)).get();
```

4.4 Document Library Services (DLS) Repositories Need To Perform A Bulk Upgrade Operation

MarkLogic 8 includes an enhancement to Document Library Services to make it significantly faster for large DLS repositories. This enhancement requires some metadata changes to the documents under DLS control.

If you have any DLS repositories created in MarkLogic 7 or earlier, you must first set compatibility mode for your repository, and then upgrade the documents in the repository, and finally set the repository to upgraded. The upgrade process will touch all of the documents under DLS control, so it will take a while, depending on the size of your DLS application. If you do not perform at least the first part of this upgrade, DLS functions might produce incorrect results in MarkLogic 8.

Because this is an upgrade that touches a large number of documents, MarkLogic strongly recommends that you first back up your database and that you thoroughly test your process on a development system before upgrading your production DLS repository.

To upgrade existing DLS repositories, perform the following steps:

1. Back up your database containing the documents under DLS control.

2. As either a user with the `admin` role, set compatibility mode for your DLS repository by running the following XQuery against your DLS database (for example, in Query Console):

```
xquery version "1.0-ml";

import module namespace dls = "http://marklogic.com/xdmp/dls"
  at "/MarkLogic/dls.xqy";

dls:set-upgrade-status(fn:false())
```

3. As a user with the `admin` role, run the following XQuery against your DLS database (for example, in Query Console):

```
xquery version "1.0-ml";
(: This starts a task which will run for a time proportional to
   the number of documents you have under DLS control. The
   function returns immediately though. It is safe to rerun
   this function if it is stopped or fails for any reason
   such as a system restart. :)

import module namespace dls = "http://marklogic.com/xdmp/dls"
  at "/MarkLogic/dls.xqy";

dls:start-upgrade()
```

   If you stop the upgrade (for example, if the server is restarted or if there are errors in the upgrade that you have fixed), you can restart the upgrade at any time by running the above query.

4. You can check the progress at any time by looking at the `upgrade-task-status.xml` document, ad in the following XQuery:

```
xquery version "1.0-ml";
(:
  this document is updated every few minutes to show the
  progress of the upgrade
:)
fn:doc("http://marklogic.com/dls/upgrade-task-status.xml")
```

5. You can check the progress at any time by running the following XQuery:

```
xquery version "1.0-ml";

import module namespace dls = "http://marklogic.com/xdmp/dls"
  at "/MarkLogic/dls.xqy";

dls:latest-validation-results()
```

6. The `dls:latest-validation-results` output has an element names `dls:validation-status`. When the value of that element is completed, the process is `complete`.

7. When you are satisfied that the process has completed (for example, if the previous step shows the process is complete), the set the upgrade status to true by running the following, either as a user with the `admin` role or a user with the `dls-admin` role:

```
xquery version "1.0-ml";

import module namespace dls = "http://marklogic.com/xdmp/dls"
  at "/MarkLogic/dls.xqy";

dls:set-upgrade-status(fn:true())
```

Once the process is successful, you can use DLS as usual.

It is also possible to run DLS in compatibility mode without running the upgrade on the repository (by not running the upgrade portion of the above procedure), but MarkLogic strongly recommends performing this upgrade procedure. If you have any problems or questions, contact MarkLogic Technical Support.

4.5 Linux Now Requires Red Hat 6

MarkLogic 8 Linux platforms now require Red Hat 6, and will no longer work on Red Hat 5; they will fail to install on Red Hat 5. So if you are running MarkLogic 7 on Red Hat 5, you will have to migrate that environment to Red Hat 6 in order to use MarkLogic 8.

4.6 mlsql On Linux No Longer Ships With Server

On Linux platforms, the `mlsql` utility no longer is packaged with the MarkLogic Server rpm binary. To get `mlsql` on Linux now, you must install the ODBC Driver for Linux. For details on the ODBC Driver for Linux, see Installing the ODBC Driver on Linux in the *SQL Data Modeling Guide*.

4.7 Cyrillic Tokenization Changes

The tokenization rules for Cyrillic script has changed such that mixed letter and number tokens are handled consistently with Latin letter and number tokens. That is, the following now tokenizes as a single token (previously it was two tokens):

```
&#x401;1 (Cyrillic A + the digit 1)
```

If you have Cyrillic content in your database, you should reindex the database or reload the Cyrillic content so that it is properly tokenized and indexed in accordance with the new rules. If you do not trigger a retokenization of any existing Cyrillic content, certain queries may behave inconsistently between old content and newly loaded content.

4.8 Application Builder Applications Must Be Re-Deployed in MarkLogic 8

Applications deployed with Application Builder in MarkLogic 7 and earlier will not work correctly in MarkLogic 8 until you fully re-deploy them.

If you have any applications built using Application Builder, you must remove the code from the modules database of the deployed application and then re-deploy the application. Specifically, for each Application Builder application that is upgraded from MarkLogic 7 or earlier:

1. Back up the application in case you need to restore it to its previous state.

2. In the modules database for the application, either clear the database or delete the following directories:

   ```
   /application
   /Default
   /marklogic.rest.resource
   /marklogic.rest.transform
   ```

   As well as the document at the following URI

   ```
   /index.html
   ```

3. Go into Application Builder and re-deploy the application. For details, see Deploy Page in the *Application Builder Developer's Guide*.

4. If you have extended your application with any customizations, re-deploy those customizations. For details on customizing Application Builder projects, see Extending Applications Built With Application Builder in the *Application Builder Developer's Guide*.

5. Test your re-deployed application.

4.9 Application Builder and Information Studio Links Removed

The links in the navigation bar from Query Console and other tools no longer have links to Application Builder and Information Studio. To use these applications, enter the URL directly (for example, http://localhost:8000/appservices).

4.10 Search API Incompatibilities

The following incompatible changes have been made to the Search API:

- search:parse Output is Now Unannotated cts:query XML
- Deprecated Option: extract-metadata
- Deprecated Functions: search:unparse, search:remove-constraint
- Structured Query: locks-query and properties-query Renamed

- sort-order Query Option Requires an Index

4.10.1 search:parse Output is Now Unannotated cts:query XML

Previously, `search:parse` produced XML representing an annotated `cts:query` that could be passed directly to `search:unparse`. As of MarkLogic 8, the default output from `search:parse` does not include annotations, so it cannot be passed to `search:unparse`.

You can get annotated output from `search:parse` in MarkLogic 8 using the `$output` parameter as shown below:

```
search:parse("myQueryText", options, "cts:annotated-query")
```

4.10.2 Deprecated Option: extract-metadata

The `extract-metadata` query option is now deprecated. Use `extract-document-data` instead. For details, see search:search or Extracting a Portion of Matching Documents in the *Search Developer's Guide*.

The new `extract-document-data` option does not support extracting specific metadata properties, but properties are available in other ways, such as using `xdmp:document-properties` or by fetching all properties metadata through one of the client APIs.

4.10.3 Deprecated Functions: search:unparse, search:remove-constraint

The `search:unparse` and `search:remove-constraint` functions are now deprecated. If you need to deconstruct a query, modify it, and "put it back together", use structured query or `cts:query` with `search:resolve` instead.

For example, if you previously did something similar to the following:

```
let $ctsquery := search:parse("myQueryString", $options)
(: ...modify $ctsquery... :)
return search:search(search:unparse($ctsquery), $options)
```

Then you can achieve the same result doing the following:

```
let $ctsquery := search:parse("myQueryString", $options)
(: ...modify $ctsquery... :)
return search:resolve($ctsquery, $options)
```

To generate a structured query instead of a `cts:query`, set the third parameter of `search:parse` to "search:query":

```
search:parse("myQueryString", $options, "search:query")
```

4.10.4 Structured Query: locks-query and properties-query Renamed

The following structured query elements have been renamed to more accurately reflect their purpose:

- `locks-query` is now `locks-fragment-query`
- `properties-query` is now `properties-fragment-query`

For details, see locks-fragment-query and properties-fragment-query in the *Search Developer's Guide*.

4.10.5 sort-order Query Option Requires an Index

If you use the `sort-order` query option to sort search results by something other than score, such as an XML element, JSON property, or field, then the database configuration must include a range index on the entity used for a sort key. Failing to create such an index causes `SEARCH-BADORDERBY` to be thrown when searching. Previously, the index requirement was not enforced.

For details, see sort-order in the *Search Developer's Guide*.

4.11 Locks and Properties Query Built-In Functions Renamed

The following built-in functions related to locks and document properties queries have been renamed to more accurately reflect their purpose.

- `cts:locks-query` is now `cts:locks-fragment-query`
- `cts:locks-query-query` is now `cts:locks-fragment-query-query`
- `cts:properties-query` is now `cts:properties-fragment-query`
- `cts:properties-query-query` is now `cts:properties-fragment-query-query`

4.12 xdmp:uri-content-type Of an XML Document Now Returns application/xml, Can Affect CPF Applications

In 8.0, the `xdmp:uri-content-type` function returns `application/xml`. In 7.0, it returns `text/xml`. Where MarkLogic previously returned `text/xml` for an xml document, it now returns `application/xml`. If you have applications that are expecting `text/xml`, you will either need to change the application to accept application/xml or you will have to modify your program to send a content type of `text/xml` (for example, using `xdmp:set-response-content-type`). MarkLogic will still accept `text/xml` for xml documents (in addition to `application/xml`). Similarly, JSON documents return `application/json` but MarkLogic accepts either `application/json` or `text/json`.

As a consequence of this change, if you have a CPF application that relies on the mimetype `text/xml` to identify an XML document, you must change that application to instead rely on `application/xml`. In the case of a CPF application, you will have to modify your CPF pipelines and change any `text/xml` mimetypes (that were referring to XML documents) to `application/xml`. If you are using the default pipelines, then reinstalling CPF for the database should correct this incompatibility. Otherwise, those CPF applications will not trigger the change actions for XML documents. Similar changes are required for CPF appliations that rely on `text/json` to identify JSON documents; they need the mimitype changed to `application/json`.

4.13 xdmp:function Signature Change

To allow for anonymous functions in Server-Side JavaScript, the first argument of the `xdmp:function` built-in function takes an `xs:QName?` (zero or 1 QNames) in 8.0; previously, it took an `xs:QName` (exactly one QName). If you have code that does not allow for zero or 1 QNames, you will need to modify that code.

Also, if you are relying on function mapping with `xdmp:function`, you can no longer use function mapping with it (because it no longer takes a singleton). If you have code that function maps with `xdmp:function`, you will need to rewrite it to not use function mapping (by calling `xdmp:function` in the return of a FLOWR statement for each item in your sequence of QNames, for example).

4.14 Incompatibilities Between 8.0-9 and 8.0-8

The following incompatibilities exist between MarkLogic 8.0-9 and MarkLogic 8.0-8:

- LDAP Connection Change

4.14.1 LDAP Connection Change

In 8.0-9, the behavior has changed for multiple LDAP connection requests. To eliminate concurrent duplicate lookup requests by the LDAP server, MarkLogic now creates a Connection Pool for any new incoming user verification request, while the outgoing LDAP request is pending. This changed behavior may affect an environment where a load balancer URI is the LDAP server.

Earlier... when we do not find user in LdapUserCahe, we create new LDAP Connection, and meanwhile if new request from same user comes in we do create another Connection to LDAP Server. Starting with v8.0-8, Bug Fix #47727, we changed the behavior and now we do Connection Pool for any new incoming user verification request till out going LDAP request is pending. This could break Customer environment if they have Load Balancer uri as LDAP.

4.15 Incompatibilities Between 8.0-8 and 8.0-7

The following incompatibilities exist between MarkLogic 8.0-7 and MarkLogic 8.0-8:

- The ldapSearch Results Format Changed

4.15.1 The ldapSearch Results Format Changed

When using server-side Javascript on MarkLogic, the `xdmp.ldapSearch` function used to return a `ValueIterator` that contained a flattened list of all `ldap` object attributes. In MarkLogic 8.0-8, the `xdmp.ldapSearch` results format has changed so that the `ldap` objects are no longer flattened. This change is not backwards compatible with earlier versions.

4.16 Incompatibilities Between 8.0-6 and 8.0-7

The following incompatibilities exist between MarkLogic 8.0-6 and MarkLogic 8.0-7:

- Changes to jsearch.facets Output Structure
- Array Type is Preserved in x509 Certificate with Array-Valued Properties
- Changes to xdmp:update XQuery Prolog Option
- map:new Retains Keys with Empty Values

4.16.1 Changes to jsearch.facets Output Structure

In previous versions of MarkLogic 8, calling `jsearch.facets` always produced a JSON object for each facet, with object properties of the form *`facetValue`*`:`*`count`*. This structure prevented proper sorting of facet values.

As of MarkLogic 8.0-7, the structure of each facet is an array of arrays instead of a JSON object if and only if you include an explicit `orderBy` clause in your facet definition. If you do not use `FacetDefinition.orderBy`, the output is unchanged.

For more details, see Sorting Facet Values with OrderBy in the *Search Developer's Guide*.

4.16.2 Array Type is Preserved in x509 Certificate with Array-Valued Properties

In MarkLogic 8.0-6 and earlier, if you use `xdmp.x509CertificateGenerate` to generate a certificate, and the configuration object includes array-valued properties, the array values were encoded as a single string. As of MarkLogic 8.0-7, the array type is preserved. This change applies to any Relative Distinguished Names (RDNs) within a Distinguished Name (DN), such as the `issuer` and `subject` DNs.

For example, in the following snippet, the `issuer.organizationName` property has an array value.

```
var certObj = {
  version: "2",
  serialNumber: "BA0195369CD6B679",
  issuer: {
    countryName: "US",
    stateOrProvinceName: "CA",
    localityName: "San Carlos",
    organizationName: ["MarkLogic", "Mark Logic"],
    organizationalUnitName: "Eng",
    emailAddress: "jdonner@marklogic.com",
    commonName: "JGD Certificate Authority",
  },...
};
var privateKey = ...;
xdmp.x509CertificatExtract(
  xdmp.x509CertificateGenerate(certObj, privateKey)
);
```

If you round trip the generated certificate through xdmp.x509CertificateExtract, you will see the following output for `issuer.organizationName` in MarkLogic 8.0-6 vs MarkLogic 8.0-7.

```
// Marklogic 8.0-6
organizationName: "[\"MarkLogic\", \"Mark Logic\"]"

// MarkLogic 8.0-7 and later versions of MarkLogic 8
organizationName: ["MarkLogic", "Mark Logic"]
```

If you do not have a certificate containing a multi-valued property, you will not notice any difference in behavior.

4.16.3 Changes to xdmp:update XQuery Prolog Option

Previously, setting the XQuery prolog option xdmp:update to “false” caused MarkLogic to automatically detect whether a module should be evaluated as an update or a query transaction.

As of MarkLogic 8.0-7, setting the option to “false” tells MarkLogic to treat the code as a query transaction. This could cause your program to get an error if you explicitly set the option to “false” and your code performs an update operation.

The new “auto” option value is equivalent to the previous behavior of “false”.

For related changes, see the following topics:

- “New Server-Side Transaction Controls” on page 24
- “xdmp:transaction-mode XQuery Prolog Option Deprecated” on page 61
- “Deprecation of transaction-mode Option to xdmp:eval” on page 62

4.16.4 map:new Retains Keys with Empty Values

As of MarkLogic 8.0-7, the `map:new` function retains any input key-value pair whose value is an empty sequence. Previous versions of MarkLogic 8 discarded such key-value pairs.

For example:

```
map:keys(map:new(map:entry("noval",())))

(: Now returns "noval". Previously, returned an empty sequence. :)
```

4.17 Incompatibilities Between 8.0-5 and 8.0-6

The following incompatibilities exist between MarkLogic 8.0-5 and MarkLogic 8.0-6:

- Terms Matched by additional-query Are Highlighted in Snippets
- Change to Required Software for mlcp and the Hadoop Connector
- Change to Required Software for the Node.js Client API
- Change to Rewriter Regex Code

4.17.1 Terms Matched by additional-query Are Highlighted in Snippets

Previously, the Search API documentation stated that terms matched by the query specified in an `additional-query` query option were not highlighted in search result snippets. This is no longer the case.

You should expect any terms matched by the `additional-query` option to be included in highlighted sections of snippets.

4.17.2 Change to Required Software for mlcp and the Hadoop Connector

To use mlcp or the MarkLogic Connector for Hadoop with HDP now requires HDP 2.4. However, there are no known incompatibilities with the previous minimum version, HDP 2.2.

To use mlcp or the MarkLogic Connector for Hadoop with CDH now requires CDH 5.8 and above. However, there are no known incompatibilities with the previous minimum version, CDH 5.4.

For more details, see Required Software in the *mlcp User Guide* or Requirements in the *MarkLogic Connector for Hadoop Developer's Guide*.

4.17.3 Change to Required Software for the Node.js Client API

The Node.js Client API now requires Node.js version 6.3.1 or later. Previously, the minimum required version was 0.10.26.

4.17.4 Change to Rewriter Regex Code

There were bug fixes made in the rewriter regex code for 8.0-6. Some regexes that were incorrectly considered invalid, are now correctly considered valid. And some regexes that were incorrectly considered valid, are now correctly considered invalid.

4.18 Incompatibilities Between 8.0-3 and 8.0-4

There are a few incompatibilities made to the Server-Side JavaScript implementation in 8.0-3. The following are the incompatibilities:

- xdmp.multipartDecode Now Returns a JSON Payload for Headers
- In JavaScript, Some Thesaurus and Spelling Function Have Different Return Type
- xdmp.databaseRestoreStatus Now Returns an Object
- Serialization Error Code Changes
- Change to Required Java Version
- Deprecated mlcp Command Line Options
- REST APIs That Have JSON or XML Payloads Cannot Have Empty Payloads
- Geospatial Namespace and Data Version Changes

4.18.1 xdmp.multipartDecode Now Returns a JSON Payload for Headers

In 8.0-4, the Server-Side JavaScript `xdmp.multipartDecode` function returns the headers (in the first item of the returned ValueIterator) as a JSON array; previously, it was returned as an XML element. If you have code that is expecting the XML element, you need to either rewrite your code to parse the JSON array or use the XQuery version (`xdmp:multipart-decode`).

4.18.2 In JavaScript, Some Thesaurus and Spelling Function Have Different Return Type

In 8.0-4, there are changes to the thesaurus and spelling function modules to make them more friendly to JavaScript users. If you have JavaScript code that imports these XQuery libraries, then the following functions now return JavaScript Object by default:

- `spell.makeDictionary`
- `thsr.lookup`

- `thsr.queryLookup`

For each of these function, you can set a new optional parameter to change its output. By default, these functions return XML output in XQuery and JavaScript Objects in JavaScript. If you have existing JavaScript code that uses these functions, you either need to add the new option to your code specifying the XML output or rework your code to accept the returned JavaScript Object. For details on these functions, see the API documentation for each function. In XQuery, the functions behave the same way they did in previous versions by default, but now allow you to specify the optional parameter to output JavaScript Objects instead of XML, if you so choose.

4.18.3 xdmp.databaseRestoreStatus Now Returns an Object

In 8.0-4, the Server-Side JavaScript API `xdmp.databaseRestoreStatus` now returns an `Object`; previously, it returned an `Array`. The `Array` that was previously returned is now the value of the `"forest"` key, and there is also a `"status"` key containing the current status of the restore. The `xdmp.databaseBackupStatus` also contains this new `"status"` key, but the return type is not changed. Similarly, the XQuery counterparts to these APIs (`xdmp:database-backup-status` and `xdmp:database-restore-status`) also contain information about the status, but their signature has not changed. If you have code that relies on any of the old behavior, you will need to modify that code to work with the changed output.

4.18.4 Serialization Error Code Changes

In 8.0-4, the names of some error exception codes for serialization, as well as the message text, have changed as shown in the following table:

Old Error Code	New Error Code
SER-DOCTYPESYSTEM2	SER-DOCTYPESYSTOPLEVTXT
SER-DOCTYPESYSTEM3	SER-DOCTYPESYSMULTIELEMROOT
SER-STANDALONE2	SER-STANDALONETOPLEVTXT
SER-STANDALONE3	SER-STANDALONEMULTIELEMROOT
SER-STANDALONE4	SER-STANDALONEOMITXMLDEC

If you have code that does a try/catch looking for one of the old exceptions or the old text, or if you have tests that use the old exceptions as keys, you will have to rewrite that code to look for the new exception.

4.18.5 Change to Required Java Version

The following tools and libraries that depend on a Java Runtime Environment (JRE) now require at least Java 7, rather than Java 6:

- mlcp

- MarkLogic Connector for Hadoop
- Java Client API
- XCC for Java (XCC/J)

4.18.6 Deprecated mlcp Command Line Options

The `-aggregate_uri_id` and `-delimited_uri_id` command line options are now deprecated. Use the more general `-uri_id` instead.

4.18.7 REST APIs That Have JSON or XML Payloads Cannot Have Empty Payloads

Starting in 8.0-4, any of the REST APIs that specifies a JSON or and XML `content-type` for its payload cannot have an empty payload. Previously, it allowed an empty payload. REST calls that specify a JSON or XML `content-type` with an empty payload throw a `MANAGE_EMPTYPAYLOAD` exception beginning in 8.0-4. For example, `POST:/admin/v1/init` previously allowed an empty payload with a JSON or XML content, but requires the payload in 8.0-4.

If you have code that does not send a payload, then you must either add an empty JSON or XML document as the payload or change the `content-type` to one that allows an empty payload (for example, `text/plain`).

4.18.8 Geospatial Namespace and Data Version Changes

The following changes have been made to some of the geospatial built-in and library functions in 8.0-4:

- GML and KML Library Modules Moved to a New Namespace
- Some Built-In Geospatial Functions Moved to geo Namespace
- Older GML and KML Versions Deprecated

4.18.8.1 GML and KML Library Modules Moved to a New Namespace

The GML library module is now in a different namespace, and the library module no longer uses the same namespace as GML and KML data.

Module	Old Module Namespace	New Module Namespace
GML	http://www.opengis.net/gml	http://marklogic.com/geospatial/gml
KML	http://earth.google.com/kml/2.0	http://marklogic.com/geospatial/kml

You must update your module import declarations in XQuery or require statements JavaScript to use the new namespace. In addition, since the module and the XML data of the same format no longer use the same namespace, you may need to change the namespace prefix you use for the module.

The following code snippets demonstrate the required changes for KML and GML in XQuery. The namespace URI in the module import declaration is changed to use the new URI, and the module namespace prefix is changed to distinguish names in the module from names in the data.

Old	New

`xquery version "1.0-ml";`
`import module namespace kml =`
`  "http://earth.google.com/kml/2.0" at`
`  "/MarkLogic/geospatial/kml.xqy";`
 	
`kml:box(`
`  <kml:LatLongBox>...</kml:LatLongBox>)`
	`xquery version "1.0-ml";`
	`import module namespace **geokml** =`
	`  "http://marklogic.com/geospatial/kml"`
	`  at "/MarkLogic/geospatial/kml.xqy";`
	`declare namespace kml =`
	`  "http://www.opengis.net/kml/2.2";`
	
	`geokml:box(`
	`  <kml:LatLongBox>...</kml:LatLongBox>)`

`xquery version "1.0-ml";`
`import module namespace gml =`
`  "http://www.opengis.net/gml"`
`  at "/MarkLogic/geospatial/gml.xqy";`
 	
`gml:box(`
`  <gml:Envelope>...</gml:Envelope>)`
	`xquery version "1.0-ml";`
	`import module namespace geogml =`
	`  "http://marklogic.com/geospatial/gml"`
	`  at "/MarkLogic/geospatial/gml.xqy";`
	`declare namespace gml =`
	`  http://www.opengis.net/gml/3.2";`
	
	`geogml:box(`
	`  <gml:Envelope>...</gml:Envelope>)`

4.18.8.2 Some Built-In Geospatial Functions Moved to geo Namespace

The following geospatial built-in functions are now deprecated and will be removed in a future release. You should use the corresponding built-in function with a “geo” prefix instead. For example, use `geo:distance` instead of `cts:distance` in XQuery, and use `geo.distance` instead of `cts.distance` in Server-Side JavaScript.

XQuery	Server-Side JavaScript
`cts:approx-center`	`cts.approxCenter`
`cts:arc-intersection`	`cts.arcIntersection`
`cts:bearing`	`cts.bearing`
`cts:box-intersects`	`cts.boxIntersects`

XQuery	Server-Side JavaScript
cts:circle-intersects	cts.circleIntersects
cts:complex-polygon-contains	cts.complexPolygonContains
cts:complex-polygon-intersects	cts.complexPolygonIntersects
cts:destination	cts.destination
cts:distance	cts.distance
cts:parse-wkt	cts.parseWkt
cts:polygon-contains	cts.polygonContains
cts:polygon-intersects	cts.polygonIntersects
cts:region-contains	cts.regionContains
cts:region-intersects	cts.regionIntersects
cts:shortest-distance	cts.shortestDistance
cts:to-wkt	cts.toWkt

4.18.8.3 Older GML and KML Versions Deprecated

As of MarkLogic 8.0-4, the default KML data version is 2.2 and the default GML data version is 3.2. Older versions are deprecated. You should convert your data.

To continue querying older versions of data with `geokml:geospatial-query` or `geogml:geospatial-query`, specify the namespace URI of the older version in your query constructor. For example:

```
GML
Old: gml:geospatial-query($regions, $options, $weight)
New: geogml:geospatial-query(
       $regions, $options, $weight, "http://www.opengis.net/gml")

KML
Old: kml:geospatial-query($regions, $options, $weight)
New: geokml:geospatial-query(
       $regions, $options, $weight, "http://earth.google.com/kml/2.0")
```

4.19 Incompatibilities Between 8.0-2 and 8.0-3

There are a few incompatibilites made to the Server-Side JavaScript implementation in 8.0-3. The following are the incompatibilities:

- spell.suggestDetailed, xdmp.filesystemDirectory, and xdmp.encodingLanguageDetect Now Return ValueIterator
- xdmp.databaseRestoreStatus Now Returns an Array

- xdmp.gssServerNegotiate Now Returns a JavaScript Object
- Use of String Transaction Ids in Node.js To Be Deprecated
- CDH 4.3 is No Longer a Supported Hadoop Distribution
- Changes to How MarkLogic Locates Java and Hadoop Libraries for HDFS Forest Storage

4.19.1 spell.suggestDetailed, xdmp.filesystemDirectory, and xdmp.encodingLanguageDetect Now Return ValueIterator

In 8.0-3, the `spell.suggestDetailed`, `xdmp.filesystemDirectory`, and `xdmp.encodingLanguageDetect` functions now return results in a `ValueIterator`. In 8.0-2 and earlier, they returned results in an `Array`. If you have any code that expects an Array, you must rework that code to accept a `ValueIterator`; for example, you can use `xdmp.arrayValues` to convert the `ValueIterator` result into an `Array` result. For details on the `ValueIterator` JavaScript type, see ValueIterator in the *JavaScript Reference Guide*.

4.19.2 xdmp.databaseRestoreStatus Now Returns an Array

In 8.0-3, the `xdmp.databaseRestoreStatus` function now returns a JavaScript `Array`. Previously, it returned an `ArrayNode`. If you have any code that expects the `ArrayNode`, you must rework that code to accept an `Array`.

4.19.3 xdmp.gssServerNegotiate Now Returns a JavaScript Object

In 8.0-3, the `xdmp.gssServerNegotiate` function (used for kerberos GSS authentication) returns a JavaScript object. Previously, it returned an XML element.

4.19.4 Use of String Transaction Ids in Node.js To Be Deprecated

The Node.js interfaces that open, manipulate, or pass around transaction ids now accept either a simple id (as before) or a transaction object. You obtain a transaction object rather than an id by passing `true` in for the new `withState` parameter of `DatabaseClient.transactions.open`. For example:

```
// old forms: return a transaction id, for backward compatibility
db.transactions.open();
db.transactions.open({timeLimit: limit, transactionName: name});

// new forms: return transaction object; preferred.
db.transactions.open(true);
db.transactions.open({withState: true, ...});
```

In a future release, the default of withState will be changed to true and the use of transaction ids will be deprecated.

If you do not modify your code to use transaction objects rather than ids, the session affinity required by multi-statement transactions is not guaranteed to be properly preserved in all cases.

For details, see Managing Transactions in the *Node.js Application Developer's Guide*.

4.19.5 CDH 4.3 is No Longer a Supported Hadoop Distribution

The MarkLogic features and tools that rely on Hadoop no longer support CDH 4.3. Instead, use one of the newer distributions listed in "Compatibility With New Versions of Hadoop" on page 16.

This change affects MarkLogic Content Pump (mlcp), the MarkLogic Connector for Hadoop, and forest storage on HDFS.

4.19.6 Changes to How MarkLogic Locates Java and Hadoop Libraries for HDFS Forest Storage

When you use HDFS for forest storage, MarkLogic must be able to find a suitable Java installation and Hadoop HDFS libraries. The algorithm for where MarkLogic looks has changed substantially as of 8.0-3.

You should now make the Hadoop libraries available to your MarkLogic hosts using one of the new Hadoop HDFS client bundles. You must unpack one of these bundles under `/opt`, `/usr`, or `/space`. For details, see "HDFS Client Bundles Available for Download" on page 17 and HDFS Storage in the *Query Performance and Tuning Guide*.

MarkLogic now locates a suitable Java installation using a different algorithm. You might need to either change the path to your JDK or set JAVA_HOME in the MarkLogic startup environment. You can now make JAVA_HOME available to MarkLogic in a way that is preserved across upgrades, using /etc/marklogic.conf. For details, see HDFS Storage in the *Query Performance and Tuning Guide*.

4.20 Incompatibilities Between 8.0-1 and 8.0-2

There are a few incompatible changes made to the Server-Side JavaScript implementation in 8.0-2. The following are the incompatibilities:

- Array Input Differences in fn.distinctValues, fn.subsequence, and Other Functions
- The Second Parameters of xdmp.eval, xdmp.invoke, xdmp.xqueryEval, and xdmp.spawn Now Take a Single Object
- extract-document-data Results Now Inline By Default
- XCC v8.0-2 May Require Config Change When Used with Older Versions of MarkLogic
- Client APIs: JavaScript Extension and Transform Error Reporting Convention Change
- Some JavaScript Built-In Functions that Returned XML Structures Now Return JSON Structures

4.20.1 Array Input Differences in fn.distinctValues, fn.subsequence, and Other Functions

In 8.0-2, the Server-Side JavaScript functions `fn.distinctValues` and `fn.subsequence` behave differently from 8.0-1 if you pass in an array. In 8.0-1, these functions will extract the values out of the array to make multiple inputs, one for each item in the array. In 8.0-2, these functions treat the array as a single item. Therefore, to get the same behavior in 8.0-2, you need to call `xdmp.arrayValues` on the array before you pass it into these functions. For example:

```
fn.distinctValues([1, 1, 2]);
// returns 1, 2 in 8.0-1
// returns [1, 1, 2] in 8.0-2
```

To modify the above code in 8.0-2 to return the same answer as in 8.0-1:

```
fn.distinctValues(xdmp.arrayValues([1, 1, 2]));
// returns 1, 2 in 8.0-2
```

The nature of the change is that there are fewer times when MarkLogic coerces an array into its values in 8.0-2 than there were in 8.0-1. The newer behavior is more natural to JavaScript developers.

In addition to `fn.distinctValues` and `fn.subsequence`, this applies to any JavaScript function that takes a `ValueIterator` as input to a parameter, including: `fn.exactlyOne`, `fn.head`, `fn.insertBefore`, `fn.reverse`, `fn.unordered`, `fn.empty`, `fn.remove`, `fn.reverse`, `fn.zeroOrOne`, `fn.oneOrMore`, `fn.deepEqual`, `fn.count`, `cts.contains`, `xdmp.setSessionField`, `xdmp.setServerField`, and others.

4.20.2 The Second Parameters of xdmp.eval, xdmp.invoke, xdmp.xqueryEval, and xdmp.spawn Now Take a Single Object

In 8.0-2, the `vars` parameter to the Server-Side JavaScript functions `xdmp.eval`, `xdmp.invoke`, `xdmp.xqueryEval`, and `xdmp.spawn` has been simplified so it only takes a single Object. If you have any code that you used in 8.0-1 that passes the external variables (`vars` parameter) as an array of Objects, as an array of stings, or as a ValueIterator, you must rewrite that code in 8.0-2 so it passes a single Object.

4.20.3 extract-document-data Results Now Inline By Default

The query option `extract-document-data` previously caused `search:search` and `search:resolve` to return a sequence consisting of the `search:response` and the extracted documents. Now, this option returns the extracted documents embedded in the `search:response` as `search:extracted` elements.

When you use the option with the REST Client API, the `/v1/search` service returns the extracted content inside the search response if the Accept header MIME type is `application/xml` or `application/json`. The extracted content is still returned as individual documents when the Accept header MIME type is multipart/mixed (a multi-document read).

For details, see Extracting a Portion of Matching Documents in the *Search Developer's Guide*.

4.20.4 XCC v8.0-2 May Require Config Change When Used with Older Versions of MarkLogic

This change affects XCC applications that set the transaction mode (`Session.setTransactionMode`) or transaction time limit (`Session.setTransactionTimeout`) and use XCC v8.0-2 or later with MarkLogic Server 8.0-1 or earlier.

If your XCC application meets the above conditions, you must set the property `xcc.txn.compatible` to `true`. If you do not do so, an exception is raised if you set the transaction time limit or set the transaction mode to a value other than `Session.TransactionMode.AUTO`.

You can set this system property on the java command line with an argument of the following form:

```
java -Dxcc.txn.compatible=true
```

You can also set the property programmatically by calling `System.setProperty`.

4.20.5 Client APIs: JavaScript Extension and Transform Error Reporting Convention Change

The following change affects applications that use the REST Client API, Java Client API, or Node.js Client API and that raise `RESTAPI-SRVEXERR` from a server-side JavaScript extension, transform, or custom snippeter, or other custom code.

In MarkLogic Server 8.0-1, when calling fn.error to raise `RESTAPI-SRVEXERR`, error detail such as the response status code and status text are passed to `fn.error` as an array. Starting with MarkLogic Server 8.0-2, the error details must be passed as a sequence. You can use `xdmp.arrayValues` to convert the array to a sequence.

For example, if you previously had an fn.error call in an extension similar to the following:

```
fn.error(null, 'RESTAPI-SRVEXERR',
         [statusCode, statusMsg, body])
```

Then you should now wrap the array that is the 3rd argument to fn.error in a call to xdmp.arrayValues, similar to the following:

```
fn.error(null, 'RESTAPI-SRVEXERR',
         xdmp.arrayValues([statusCode, statusMsg, body]))
```

4.20.6 Some JavaScript Built-In Functions that Returned XML Structures Now Return JSON Structures

The following APIs return JSON output in 8.0-2.

- `cts.plan`
- `cts.relevanceInfo`
- `xdmp.zipManifest`
- `xdmp.userExternalSecurity`
- `xdmp.userLastLogin`
- `xdmp.databasePathNamespaces`
- `cts.distinctiveTerms`
- `cts.cluster`
- `cts.train`
- `cts.classify`
- `cts.thresholds`

In 8.0-1, these APIs returned the same XML structures that their XQuery counterparts return. If you have any JavaScript code that relies on the XML structures, you will need to modify that code to use the new JSON output.

4.21 Planning for Future Upgrades

This section provides guidelines for preparing for changes expected in a future release. You are not required to make related changes at this time.

- ValueIterator to be Replaced with Sequence
- xdmp:transaction-mode XQuery Prolog Option Deprecated
- Deprecation of transaction-mode Option to xdmp:eval
- XCC Session.setTransactionMode is Deprecated

4.21.1 ValueIterator to be Replaced with Sequence

This topic applies only to users of Server-Side JavaScript.

The ValueIterator interface is a JavaScript iterator that encapsulates one or more MarkLogic values. Many built-in and library functions that return or accept a sequence in XQuery return or accept a ValueIterator in Server-Side JavaScript in MarkLogic 8.

The next major release of MarkLogic will deprecate ValueIterator in favor of a new interface named Sequence, which is a JavaScript Iterable. All MarkLogic interfaces that accept or return a ValueIterator in MarkLogic 8 will use a Sequence in MarkLogic 9. It will no longer be possible to create a ValueIterator or receive one as a function result.

This change can be mostly transparent to your application if you use ValueIterator objects in the recommended way in MarkLogic 8. For details, see Preparing to Switch from ValueIterator to Sequence in the *JavaScript Reference Guide*.

4.21.2 xdmp:transaction-mode XQuery Prolog Option Deprecated

The XQuery prolog option `xdmp:transaction-mode` is deprecated as of MarkLogic 8.0-7. Use the `xdmp:commit` and `xdmp:update` prolog options instead.

Note that the new prolog options differ from `xdmp:transaction-mode` in that they affect only the transaction created after their declaration, where as `xdmp:transaction-mode` settings persist across an entire session.

The following table illustrates the correspondence between the old and new options settings.

`xdmp:transaction-mode` Value	Equivlaent `xdmp:commit` and `xdmp:update` Option Settings
`"auto"`	`declare option xdmp:commit "auto";` `declare option xdmp:update "auto";`
`"update-auto-commit"`	`declare option xdmp:commit "auto";` `declare option xdmp:update "true";`
`"update"`	`declare option xdmp:commit "explicit";` `declare option xdmp:update "true";`
`"query"`	`declare option xdmp:commit "explicit";` `declare option xdmp:update "false";`

Note that the default values for `xdmp:commit` and `xdmp:update` are both “auto”, so you do not need to set this value explicitly in most cases.

For more details, see xdmp:update and xdmp:commit in the *XQuery and XSLT Reference Guide*.

4.21.3 Deprecation of transaction-mode Option to xdmp:eval

The `transaction-mode` option of the `xdmp:eval` XQuery function and the `xdmp.eval` JavaScript function is deprecated as of MarkLogic 8.0-7. Use the `commit` and `update` options instead. For more details, see the function reference documentation for `xdmp:eval` (XQuery) and `xdmp.eval` (JavaScript).

This option deprecation (and alternative option settings apply to the following functions:

XQuery	JavaScript
`xdmp:eval`	`xdmp.eval`
`xdmp:javascript-eval`	`xdmp.xqueryEval`
`xdmp:invoke`	`xdmp.invoke`
`xdmp:invoke-function`	`xdmp.invokeFunction`
`xdmp:spawn`	`xdmp.spawn`
`xdmp:spawn-function`	

The following table illustrates the correspondence between the old and new option settings:

`transaction-mode` Option Value	Equivalent `commit` and `update` Option Values
`auto`	`commit: "auto"` `update: "auto"`
`update-auto-commit`	`commit: "auto"` `update: "true"`
`update`	`commit: "explicit"` `update: "true"`
`query`	`commit "explicit"` `update "false"`

4.21.4 XCC Session.setTransactionMode is Deprecated

Use of `Session.setTransactionMode` to specify commit semantics and transaction type is deprecated as MarkLogic 8.0-7. This function will be removed in a future version. Use the new `Session.setAutoCommit` and `Session.setUpdate` methods instead.

The following table illustrates how to replace calls to `setTransactionMode` with equivalent calls to `setAutoCommit` and `setUpdate`.

If you call `setTransactionMode` with this value:	Then replace it with the following calls on the same Session object
`AUTO`	`setAutoCommit(true);` `setUpdate(Session.Update.AUTO);`
`MULTI_AUTO`	`setAutoCommit(false);` `setUpdate(Session.Update.AUTO);`
`UPDATE`	`setAutoCommit(false);` `setUpdate(Session.Update.TRUE);`
`QUERY`	`setAutoCommit(false);` `setUpdate(Session.Update.FALSE);`
`UPDATE_AUTO_COMMIT`	`setAutoCommit(true);` `setUpdate(Session.Update.TRUE);`
`QUERY_SINGLE_STATEMENT`	`setAutoCommit(true);` `setUpdate(Session.Update.FALSE);`

4.22 MarkLogic 7 Incompatibilites

MarkLogic 8 allows you to upgrade either from MarkLogic 5, MarkLogic 6, or MarkLogic 7. If you are upgrading from 4.2, you must first upgrade to at least MarkLogic 5, and there are some known incompatibilities between 4.2 and 5.0 that are documented in the 5.0 *Release Notes*. If you are upgrading from MarkLogic 7, you can skip this section. For convenience, the incompatibilities between MarkLogic 6 and MarkLogic 7 are repeated here, and are as follows:

- Incompatibilities Between MarkLogic 7.0-3 and 7.0-2
- Incompatibilities Between MarkLogic 7.0-2 and 7.0-1
- Incompatibilities Between MarkLogic 7.0-1 and MarkLogic 6

4.22.1 Incompatibilities Between MarkLogic 7.0-3 and 7.0-2

This section describes the incompatibilities between MarkLogic 7.0-3 and 7.0-2.

- HDP No Longer a Supported Hadoop Platform
- Java API: ContentVersionRequest Property Deprecated
- REST API: content-versions Property Deprecated
- REST API: JSON documents Cannot be Retrieved as XML

- Float Precision Greater in 7.0-3

4.22.1.1 HDP No Longer a Supported Hadoop Platform

Hortonworks Data Platform (HDP) is no longer a supported Hadoop distribution for use with the MarkLogic Connector for Hadoop or the distributed mode of MarkLogic Content Pump (mlcp).

4.22.1.2 Java API: ContentVersionRequest Property Deprecated

The REST server configuration property ContentVersionRequest is now deprecated. If you currently use `com.marklogic.admin.ServerConfigurationManager.setContentVersionRequest()` and the related `ServerConfiguration.Policy` type, you should modify your application to use `com.marklogic.admin.ServerConfigurationManager.setUpdatePolicy()` and `ServerConfigurationManager.UpdatePolicy` instead. You should also change calls to `getContentVersionRequest()` into call to `getUpdatePolicy()`.

The table below shows the correspondence between `Policy` values and `UpdatePolicy` values. The behavior is unchanged with respect to these values.

If you used `Policy` Value...	Then use `UpdatePolicy` Value...
`NONE`	`MERGE_METADATA`
`REQUIRED`	`VERSION_REQUIRED`
`OPTIONAL`	`VERSION_OPTIONAL`

4.22.1.3 REST API: content-versions Property Deprecated

The REST instance configuration property `content-versions` is now deprecated, in favor of the new `update-policy` property.

If you currently set `content-versions`, you should modify your application to use the `update-policy` configuration property instead. For example, if you set this property using `PUT /v1/config/properties` or `PUT /v1/config/properties/content-versions`, then you should now use `PUT /v1/config/properties` or `PUT /v1/config/properties/update-policy` to set `update-policy` instead.

Similarly, you should modify any code that reads the configuration properties to expect `update-policy` instead of `content-versions`.

The table below shows the correspondence between `content-versions` values and `update-policy` values. The behavior is unchanged with respect to these values.

If you used `Policy` Value...	Then use `UpdatePolicy` Value...
`none`	`merge-metadata`
`required`	`version-required`
`optional`	`version-optional`

4.22.1.4 REST API: JSON documents Cannot be Retrieved as XML

In previous versions, you could use the REST Client API to retrieve the internal XML representation of a JSON document using `GET /v1/documents` and specifying XML in the `format` request parameter or `application/xml` in the Accept header.

As of MarkLogic 7.0-3, JSON documents are always returned as JSON. You can still examine the internal representation of a JSON document using XQuery or the Query Console database explorer.

4.22.2 Float Precision Greater in 7.0-3

In MarkLogic 7.0-3, the numeric precision of a float has increased. For example:

```
xs:float(3435.99884)
(: Returns 3436 in 7.0-2,
   returns 3435.999 in 7.0-3 :)
```

In most cases, this will not cause an incompatibility, as it is just returning a more precise number. But if you have application logic that relies on the old behavior, you will have to make changes to account for the greater precision.

4.22.3 Incompatibilities Between MarkLogic 7.0-2 and 7.0-1

This section describes the incompatibilities between MarkLogic 7.0-1 and 7.0-2.

- Change to JSON Output from the REST API
- Changes to the MarkLogic Connector for Hadoop API

4.22.3.1 Change to JSON Output from the REST API

The JSON output from the manage REST resource addresses with the `metrics` view has been changed and you may need to change any custom clients that consume this JSON data.

4.22.3.2 Changes to the MarkLogic Connector for Hadoop API

`com.marklogic.mapreduce.MarkLogicDocument` is now an interface instead of a class. The previous functionality of `MarkLogicDocument` is provided by the new class `com.marklogic.mapreduce.DatabaseDocument`.

Modify your code and job configuration to use DatabaseDocument instead of MarkLogicDocument.

4.22.4 Incompatibilities Between MarkLogic 7.0-1 and MarkLogic 6

- XQuery HTTP Client Built-In Functions Now Require a Privilege
- HTTP Client Functions Are Now HTTP 1.1 Compliant
- xdmp:get-request-username and xdmp:get-request-user Changes
- Specifying a Forest Now Only Works With Strict Locking
- Custom Dictionaries for Japanese and Chinese Languages Need to be Re-saved
- Default Attributes on XML Copy Changes
- Serialization of Alerting, Reverse, and Path Range Queries Change
- Java and REST Client API Incompatibilities
- Namespace Change for Properties Persisted Using JSON
- mlcp Incompatibilities
- REST Management API Version Incremented to v2
- Changes to the Configuration Manager
- xdmp:plan Now Requires a Privilege
- fn:analyze-string Now Returns Output in a Different Namespace

4.22.4.1 XQuery HTTP Client Built-In Functions Now Require a Privilege

The HTTP client XQuery APIs (`xdmp:http-delete`, `xdmp:http-get`, `xdmp:http-post`, `xdmp:http-put`, and so on) now require a privilege. Previously, these functions did not require a privilege. To support these privileges, the following privileges are added to MarkLogic 7:

- `http://marklogic.com/xdmp/privileges/xdmp-http-get`
- `http://marklogic.com/xdmp/privileges/xdmp-http-head`
- `http://marklogic.com/xdmp/privileges/xdmp-http-options`
- `http://marklogic.com/xdmp/privileges/xdmp-http-delete`
- `http://marklogic.com/xdmp/privileges/xdmp-http-post`
- `http://marklogic.com/xdmp/privileges/xdmp-http-put`

Additionally, the `network-access` role has been added, which contains all of these privileges.

If you have code that accesses these functions with a user that does not have the `admin` role, you will have to add these privileges (or the `network-access` role) to the set of privileges inherited by your users. For details on adding privileges, see Security Administration in the *Administrator's Guide*. For details on security, see *Understanding and Using Security Guide*.

4.22.4.2 HTTP Client Functions Are Now HTTP 1.1 Compliant

The XQuery HTTP client functions (`xdmp:http-delete`, `xdmp:http-get`, `xdmp:http-post`, `xdmp:http-put`, and so on) now fully implement HTTP 1.1, including the use of connection keep-alives and built-in decoding of chunked transfer encoded responses. In most cases, this will not cause any incompatible behavior, but if your code tried to perform some work to do HTTP 1.1 features such as taking a chunked response as a large binary and decoding the chunks in XQuery, then that code might behave differently in MarkLogic 7. If you have such code, you might need to rework it in MarkLogic 7.

4.22.4.3 xdmp:get-request-username and xdmp:get-request-user Changes

The `xdmp:get-request-username` and `xdmp:get-request-user` functions have changed slightly in MarkLogic 7.

In previous releases, `xdmp:get-request-username` always returned the user in the `Authorization` HTTP header; in MarkLogic 7, if you are using application-level authentication, then it returns the user from the last successful call to `xdmp:login` (when using any other authentication scheme, it is unchanged from previous releases and returns the user in the `Authorization` HTTP header).

In previous releases, `xdmp:get-request-user` returned the ID of the current user; in MarkLogic 7, if you are using application-level authentication, then it returns the user ID from the last successful call to `xdmp:login` (when using any other authentication scheme, it now returns the user ID corresponding to the user in the `Authorization` HTTP header). If you want the old behavior of returning the ID of the current user, use the new function `xdmp:get-current-userid`.

If you have any code that uses the `xdmp:get-request-username` or `xdmp:get-request-user` functions, you might need to change it to work with the current behavior.

4.22.4.4 Specifying a Forest Now Only Works With Strict Locking

The various loading APIs (for example, `xdmp:document-load` and `xdmp:document-insert`) have an option to specify the forest in which a document is loaded. In MarkLogic 7, those forest options are only available with the `locking` parameter on the database set to `strict` (the default is `fast`). If you try to specify forest placement with `fast` locking, it will throw an XDMP-PLACEKEYSLOCKING exception. Previously, this operation would be allowed.

In most cases, it is not recommended to specify a placekey, as MarkLogic does a good job of distributing data. If you want to continue to use the forest placekeys when loading or updating, you must change your `locking` parameter to `strict` or change your code to no longer use forest placekeys. You can also consider using Tiered Storage to control what data goes into which forests. For details on Tiered Storage, see Tiered Storage in the *Administrator's Guide*.

4.22.4.5 Custom Dictionaries for Japanese and Chinese Languages Need to be Re-saved

In MarkLogic 7, there is a change to the way custom dictionaries are stored for languages that use a unified cd (Japanese and both Simplified and Traditional Chinese). The custom dictionaries for those languages now have separate files for tokenization and for stemming. If you are using these languages and you have created a custom dictionary, you must re-save the custom dictionary in MarkLogic 7, which will save it in the new format.

For example, if you have a Japanese language custom dictionary, run the following XQuery program to re-save the custom dictionary:

```
import module namespace
   cdict="http://marklogic.com/xdmp/custom-dictionary"
   at "/MarkLogic/custom-dictionary.xqy";

cdict:dictionary-save("ja", cdict:dictionary-read("ja"))
```

This will re-save the dictionary to the new format.

4.22.4.6 Default Attributes on XML Copy Changes

In MarkLogic 7, the behavior of default attributes has changed when you are copying XML nodes that have an in-scope schema with default attributes. In MarkLogic 6, the default attributes were applied in the copied node. In MarkLogic 7, the default attributes are still applied in the data model of the copied node, but they will only be serialized if `default-attributes=yes` is set in the serialization options (for example, the `xdmp:output` option). This is true for copied nodes in both XQuery (for example, see below) and XSLT (for example, `xsl:copy`).

The following shows an example using the XHTML schema, which is always in-scope.

```
xquery version "1.0-ml";
declare option xdmp:output "default-attributes=no";
(: the above xdmp:output declaration is the default :)

<x>{xdmp:unquote('<html xmlns="http://www.w3.org/1999/xhtml">
    <body>
       <p>hello</p>
    </body>
</html>
')}</x>
=>
In MarkLogic 7 returns: (no default attributes on html element)
```

```
<x>
  <html xmlns="http://www.w3.org/1999/xhtml">
    <body>
       <p>hello</p>
    </body>
  </html>
</x>

In MarkLogic 6 returns: (added default attribute version
                         on html element)
<x>
  <html version="-//W3C//DTD XHTML 1.1//EN"
        xmlns="http://www.w3.org/1999/xhtml">
    <body>
       <p>hello</p>
    </body>
  </html>
</x>
```

If the `xdmp:output` option is set to `default-attributes=yes`, then the attributes will continue to be serialized in MarkLogic 7. Furthermore, if the copied node is used in another schema that has different default values for the default attributes, then the resulting default value for those attributes comes from the copied node, not the new schema. This change is not likely to impact very many applications, and the new behavior is what most people would expect, but if you have code that relies on the old behavior, you will need to explicitly set the `xdmp:output` option is set to `default-attributes=yes`.

4.22.4.7 Serialization of Alerting, Reverse, and Path Range Queries Change

The serialization of path range index queries has changed to use `cts:path-expression` rather than `cts:path` as the element name. This affects not only stored path range index queries, but also stored Alerting queries and reverse queries. All such queries must be transformed to the new serialization and reloaded.

The following XSLT stylesheet makes the required change:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:cts="http://marklogic.com/cts"
                version="2.0">
  <xsl:template match="cts:path">
    <xsl:element name="cts:path-expression">
      <xsl:copy-of select="namespace::*|@*|node()"/>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

You can apply the stylesheet to affected documents using a query similar to the following:

```
xquery version "1.0-ml";
for $query in fn:collection("my-stored-queries")
return xdmp:node-replace($query, xdmp:xslt-eval($stylesheet, $query)
```

4.22.4.8 Java and REST Client API Incompatibilities

Unless otherwise noted, the following changes can affect applications using either the REST or Java Client API:

- Content Transformations on JSON Documents Operate on XML
- Resource Service Extensions Moved to Modules Database on Upgrade
- JSON Key Name Change for System Properties
- Java Batch Example Package Name Change

Content Transformations on JSON Documents Operate on XML

This topic applies to REST and Java Client API applications that include custom content transformations for JSON documents.

JSON documents are stored in MarkLogic Server as XML. In MarkLogic 6, content transformation functions were invoked on JSON documents after conversion from JSON to XML. In MarkLogic 7, content transformations are applied before this conversion.

If you created a content transformation for MarkLogic 6 that expects JSON input, you must modify your implementation to expect XML in the `http://marklogic.com/xdmp/json/basic` namespace instead of JSON text. For details on the XML representation, see Working With JSON in *Application Developer's Guide*.

Transformations invoked during ingestion must similarly be modified to produce XML that conforms to the internal representation for JSON documents. Transformations invoked during document retrieval can either produce conforming XML or JSON text as when generating JSON output.

For details on the new expectations, see Expected Input and Output in *REST Application Developer's Guide*.

Resource Service Extensions Moved to Modules Database on Upgrade

This topic applies to REST and Java Client API applications running on a REST API instance created with MarkLogic 6.0-1 or later that use resource service extensions.

REST API instances created with MarkLogic Server version 6.0-1 stored resource service extensions in the Extensions database. REST API instances created with MarkLogic Server version 6.0-2 or later install resource service extensions in the modules database associated with the instance.

When you upgrade from MarkLogic 6 to MarkLogic 7 or later, any resource service extensions stored in the Extensions database on behalf of a 6.0-1 REST API instance are automatically migrated from the Extensions database to the modules database for you.

If your resource extension has dependent libraries or other assets that you installed in the Extensions database, you should migrate them by re-installing them using the new `/ext` service. For details, see Managing Dependent Libraries and Other Assets in *REST Application Developer's Guide*.

JSON Key Name Change for System Properties

This topic applies to REST applications that reference document properties using the JSON representation.

When you retrieve document properties in JSON format, protected system properties such as last-modified are now enclosed in an object with the key `$ml.prop`. In MarkLogic 6, such properties were immediate children of the `properties` container. User-defined property naming is unchanged.

The example output below shows the change in how the last-modified property value is returned by a request of the form `GET /v1/documents?uri=`*`your-uri`*`&category=properties&format=json`:

MarkLogic 6	MarkLogic 7

{"properties":{
  "last-modified":"*value*"
}}
	{"properties":{
	  "$ml.prop":{
	  "last-modified":"*value*"
	  }
	}}

Java Batch Example Package Name Change

The Java example application in the package `com.marklogic.client.example.batch` is now in the package `com.marklogic.client.example.extension`. If your application references any classes or interfaces from this package, you must change your package imports and recompile.

4.22.4.9 Namespace Change for Properties Persisted Using JSON

This topic applies to REST and Java Client API applications that insert or update document properties using JSON and either of the following are true:

- You define indexes based on JSON property keys.
- Your application includes a content transformation, resource extension, or other code that that manipulates JSON property metadata in its XML representation.

In MarkLogic 6, inserting or updating user-defined document properties using JSON stored the properties as XML elements in the namespace `http://marklogic.com/json`. In MarkLogic 7, such properties are stored as elements in the namespace `http://marklogic.com/xdmp/json/basic`. If you insert or update a property using JSON using MarkLogic 7, the new namespace is used.

If you convert properties in the old namespace to the new namespace, then all pre-existing and future properties will use the new namespace. If you do not perform such a conversion, then you should adapt your indexes and queries to accommodate both namespaces because pre-existing properties will be in the MarkLogic 6 namespace while new or updated properties will be in the MarkLogic 7 namespace.

After upgrading to MarkLogic 7, use one of the following solutions to adapt your content and application to this change:

- Modify Pre-Existing Propertie
- Modify Queries to Use Both Namespaces
- Create a Field That Spans Both Namespaces

Modify Pre-Existing Propertie

Use a query similar to the one in this section to change the namespace of all properties in the old namespace. The example query uses `xdmp:spawn` to perform the property update in batches, thereby avoiding overly large transactions.

The procedure below assumes you have an App Server, such as a REST API instance, associated with your content database. If you are not familiar with Query Console, see *Query Console User Guide*.

The following procedure walks you through installing a transform query in the modules root of an App Server attached to an affected database, and then running the query to modify a specific property.

1. Save the following transform query to a file, such as `update-props.xqy`. You will changed the bolded text in the next step.

```
xquery version "1.0-ml";

declare namespace prop   = "http://marklogic.com/xdmp/property";
declare namespace old-ns = "http://marklogic.com/json";

declare default function namespace
   "http://www.w3.org/2005/xpath-functions";
declare option xdmp:mapping "false";

declare function local:transform(
    $elem as element()
) as element()
{
```

```
    let $elem-name :=
        if (exists($elem/self::old-ns:*))
        then QName("http://marklogic.com/xdmp/json/basic",
                   local-name($elem))
        else node-name($elem)
    return element {$elem-name} {
        $elem/@*,
        for $child in $elem/node()
        return
            typeswitch ($child)
            case element() return local:transform($child)
            default return $child
        }
};

let $max   := 100
let $batch :=
    subsequence(
        cts:search(collection(),
            cts:properties-query(
                cts:element-query(xs:QName("old-ns:my-prop"),
                                  cts:and-query(()))
                )
            ),
        1,
        $max
        )/document-uri(.)
return
    if (empty($batch)) then ()
    else (
        for $doc-uri in $batch
        let $current-props  := xdmp:document-properties($doc-uri) /
            prop:properties/(* except prop:last-modified)
        let $modified-props := $current-props/local:transform(.)
        return xdmp:document-set-properties($doc-uri, $modified-props),
        if (count($batch) lt $max) then ()
        else xdmp:spawn("/some/path/update-props.xqy")
        )
```

2. Modify the saved query to match your environment by making the following changes:

   a. Change occurrences of “my-prop” to the name of an affected property in your content.

   b. Change the module path in the `xdmp:spawn` call to the path where you will install the query in your modules root.

3. Install the saved file in the modules database or modules root of your App Server. Install the module under the path you chose in Step 2b.

   a. To install the modules database of a REST API instance using the REST API, see Managing Dependent Libraries and Other Assets in *REST Application Developer’s Guide*.

b. To install in the modules database of a REST API instance using Java API, see Managing Dependent Libraries and Other Assets in *Java Application Developer's Guide*.

c. To install in the modules database using XQuery, run a query similar to the following in Query Console, after modifying the filesystem path and database URI. The file must be accessible to MarkLogic Server. Run the query with the modules database as the content source.

```
xquery version "1.0-ml";
xdmp:document-load(
  "/filesystem/path/update-props.xqy",
  <options xmlns="xdmp:document-load">
    <uri>/some/path/update-props.xqy</uri>
  </options>
  )
```

4. Run the transform query using Query Console.

a. Create a new query in Query Console with the following contents:

```
xquery version "1.0-ml";
xdmp:invoke("/my.domain/update-props.xqy")
```

b. Modify the module URI in the `xdmp:spawn` call to the URI under which you installed the module in Step 3.

c. In the Query Console Content Source dropdown, select the source that corresponds to your content database and the modules database in which you installed the transform query in Step 3.

d. Click the Run button to perform the transformation.

5. If your database is large, you might exceed the maximum number of spawned queries. If this happens, wait for the previous spawns to complete, and then run the query again.

You must also modify any range indexes, queries, or query options for this property that depend on the old namespace. Change occurrences of `http://marklogic.com/json` to `http://marklogic.com/xdmp/json/basic`.

For example, if in MarkLogic 6 you defined an element range index over the property `my-property` in the namespace `http://marklogic.com/json`, modify the your index configuration to use the namespace URI `http://marklogic.com/xdmp/json/basic`.

Similarly, if you have queries or constraint definitions using the old namespace, change them to use the new namespace. The table below contains an example of how to modify a properties constraint to use the new namespace:

Version	Example

MarkLogic 6	`<options xmlns="http://marklogic.com/appservices/search">`
	`  <constraint name="my-prop">`
	`  <range type="xs:string">`
	`  <element name="my-property"`
	`  **ns="http://marklogic.com/json"** />`
	`  <fragment-scope>properties</fragment-scope>`
	`  </range>`
	`  </constraint>`
	`</options>`

MarkLogic 7	`<options xmlns="http://marklogic.com/appservices/search">`
	`  <constraint name="my-prop">`
	`  <range type="xs:string">`
	`  <element name="my-property"`
	`  **ns="http://marklogic.com/xdmp/json/basic"** />`
	`  <fragment-scope>properties</fragment-scope>`
	`  </range>`
	`  </constraint>`
	`</options>`

Modify Queries to Use Both Namespaces

You can modify your queries and query options to accommodate both namespaces. For example, if your query options already include a constraint on the old namespace, add a constraint for the new namespace, and then use an OR query to apply both constraints.

The following example uses a JSON structured or-query to demonstrate this technique. It assumes the existence of an element range index on the property in the new namespace.

```
{"search": {
  "options": {
    "constraint": [
      {
        "name": "old-prop",
        "range": {
          "type": "xs:string",
          "element": {
            "name": "my-property",
            "ns": "http://marklogic.com/json"
          },
          "fragment-scope": "properties"
        }
      },
      {
        "name": "new-prop",
```

```
            "range": {
              "type": "xs:string",
              "element": {
                "name": "my-property",
                "ns": "http://marklogic.com/xdmp/json/basic"
              },
              "fragment-scope": "properties"
            }
          }
        ]
      },
      "query" : {
        "or-query" : {
          "queries" : [
            {"range-constraint-query": {
              "constraint-name": "old-prop",
              "value": "the value"
            }},
            {"range-constraint-query": {
              "constraint-name": "new-prop",
              "value": "the value"
            }}
        ]}
      }
    }}
```

Create a Field That Spans Both Namespaces

Use this procedure to replace an existing element range index over the property in the old namespace with a field range index that covers both namespaces, and then modify your code and queries to the use new index. To learn more about fields, see Fields Database Settings in *Administrator's Guide*.

1. Use the Admin Interface to define namespace prefixes for `http://marklogic.com/json` and `http://marklogic.com/xdmp/json/basic`. For details, see Steps 1-8 of Defining Path Range Indexes in *Administrator's Guide*.

2. Create a field that includes two paths, one for the property in each namespace. Use the namespace prefixes defined in Step 1. For example, add the paths `old-ns:my-property` and `new-ns:my-property`. For details, see Configuring a New Field in *Administrator's Guide*.

3. Create a field range index over the new field. For details, see Creating a Range Index on a Field in *Administrator's Guide*.

4. Modify any extensions, transformations, queries or query options that rely on the old element range index to use the new field range index. The example below shows a combined query modified to use the new field range index instead of the old element range index.

Old	New

{ "search": {
"options": {
"constraint": {
"name": "my-prop",
"range": {
"type": "xs:string",
"element": {
"name": "my-property",
"ns": "http://marklogic.com/json"
},
"fragment-scope": "properties"
}}},
"query" : {
"range-constraint-query": {
"constraint-name": "my-prop",
"value": "the value"
}}
}}
	{"search": {
	"options": {
	"constraint": {
	"name": "my-prop",
	"range": {
	"type": "xs:string",
	"field": {
	"name": "my-new-property"
	},
	"fragment-scope": "properties"
	}}},
	"query" : {
	"range-constraint-query": {
	"constraint-name": "my-prop",
	"value": "the value"
	}}
	}}

4.22.4.10mlcp Incompatibilities

MarkLogic Content Pump (mlcp) version 1.1 includes the following changes that potentially affect compatibility for users of mlcp version 1.0-*.

- Compressed Input Default URI Includes Input Filename
- Default Character Encoding Changed to UTF-8

Compressed Input Default URI Includes Input Filename

The following change might affect you if you use mlcp to load documents from compressed files (`-input_compressed`). Documents created with mlcp v1.0 used the following default URI template:

```
/path/inside/zip/filename
```

Starting with version 1.1, the default URI template is:

```
/compressed-file-path/path/inside/zip/filename
```

This change can affect the `-output_uri_replace` patterns needed to create documents with the same URIs as those created with mlcp 1.0.

For details, see Default Document URI Construction in the *mlcp User Guide*.

Default Character Encoding Changed to UTF-8

The default content character encoding when importing and exporting documents with mlcp has changed to UTF-8. Previously, mlcp used the platform default encoding for the host which mlcp was running.

You can get the previous behavior by using the following option setting:

```
-content-encoding system
```

4.22.4.11REST Management API Version Incremented to v2

The REST Management API version has been incremented to v2. The v1 services are no longer available. If you send a request using a v1 URL, MarkLogic Server responds with status code 410 (Gone) and a `MANAGE-UNSUPPORTEDVERSION` error.

Requests that use `LATEST` as the version when constructing requests will continue to work. However, you may need to make other changes due to the behavior changes between v1 and v2.

The incompatibilities between v1 and v2 are detailed in the remainder of this section

- View Parameter Required Instead of Path Steps
- JSON Output Includes Units
- Element/Key Name Changes in Status Views
- Changes to the Management API Plugins
- Changes to the Packaging API

View Parameter Required Instead of Path Steps

In previous releases, there were two ways to access some views: by path step or using the `view` request parameter. In MarkLogic 7, the path step form of URL has been removed. You must now use the `view` parameter. For example, MarkLogic 6 supported the following two ways of requesting database status, where *`version`* is `v1` or `LATEST`:

```
GET /manage/version/databases/{id|name}/status
GET /manage/version/databases/{id|name}?view=status
```

In MarkLogic 7, only the second form is supported, as shown in the example below, where *`version`* is `v2` or `LATEST`.

```
GET /manage/version/databases/{id|name}?view=status
```

This change applies to the `config`, `counts`, `edit`, and `status` views for clusters, databases, forests, groups, hosts, and servers. The table below lists the affected GET methods and the equivalent MarkLogic 7 URL.

Previous Form	MarkLogic 7 Form
/manage/v1/clusters/{id|name}/config	/manage/v2/clusters/{id|name}?view=config
/manage/v1/clusters/{id|name}/status	/manage/v2/clusters/{id|name}?view=status
/manage/v1/databases/{id|name}/config	/manage/v2/databases/{id|name}?view=config
/manage/v1/databases/{id|name}/counts	/manage/v2/databases/{id|name}?view=counts
/manage/v1/databases/{id|name}/edit	/manage/v2/databases/{id|name}?view=edit
/manage/v1/databases/{id|name}/status	/manage/v2/databases/{id|name}?view=status
/manage/v1/forests/{id|name}/config	/manage/v2/forests/{id|name}?view=config
/manage/v1/forests/{id|name}/counts	/manage/v2/forests/{id|name}?view=counts
/manage/v1/forests/{id|name}/edit	/manage/v2/forests/{id|name}?view=edit
/manage/v1/forests/{id|name}/status	/manage/v2/forests/{id|name}?view=status
/manage/v1/groups/{id|name}/config	/manage/v2/groups/{id|name}?view=config
/manage/v1/groups/{id|name}/counts	/manage/v2/groups/{id|name}?view=counts
/manage/v1/groups/{id|name}/edit	/manage/v2/groups/{id|name}?view=edit
/manage/v1/groups/{id|name}/status	/manage/v2/groups/{id|name}?view=status
/manage/v1/hosts/{id|name}/config	/manage/v2/hosts/{id|name}?view=config
/manage/v1/hosts/{id|name}/counts	/manage/v2/hosts/{id|name}?view=counts
/manage/v1/hosts/{id|name}/edit	/manage/v2/hosts/{id|name}?view=edit
/manage/v1/hosts/{id|name}/status	/manage/v2/hosts/{id|name}?view=status
/manage/v1/servers/{id|name}/config	/manage/v2/servers/{id|name}?view=config
/manage/v1/servers/{id|name}/edit	/manage/v2/servers/{id|name}?view=edit
/manage/v1/servers/{id|name}/status	/manage/v2/servers/{id|name}?view=status

JSON Output Includes Units

JSON output for the various GET methods now includes units for metrics that were previously flat key-value pairs. The following template summarizes the difference:

- MarkLogic 6: `"key" : the-value`
- MarkLogic 7: `"key" : { "units" : "the-units", "value" : the-value }`

For example, in MarkLogic 6, `elapsed-time` was reported as follows:

```
"elapsed-time" : "0.023453"
```

In MarkLogic 7, this field include a unit, as shown in this example:

```
"elapsed-time" : { "units" : "sec", "value" : "0.023453" }
```

Element/Key Name Changes in Status Views

The following XML element/JSON key name changes have been made:

- Many element/key names with a “total-” prefix in status views no longer use this prefix. This change affects the `load-detail` and `rate-detail` sections of the status views for clusters, databases, forests, groups, and hosts.
- The `on-disk-size` element/key of the databases status view has been renamed to `data-size`. This change affects `GET /manage/version/databases/{id|name}?view=status` output.

For example, in earlier releases, the `load-detail` section of the report returned by `GET /manage/LATEST/hosts/{id|name}?view=status` includes a `total-query-read-load` element/key. The equivalent data is now named `query-read-load`.

Changes to the Management API Plugins

There have been some changes to the Management API plugins for MarkLogic 7, so if you have a plugin written against previous versions, depending upon the plugin, you might need to update it.

In previous releases (MarkLogic 5 and MarkLogic 6), plugins are installed in the following folder:

```
Assets/plugins/marklogic/manage/v1
```

Plugins installed to this directory will continue to work with the following exceptions:

- Previous releases support 3 types of plugins: "format", "resource", and "view extender". In MarkLogic 7:
  - No changes are necessary for "format" plugins, these will continue to work as before with no changes to the plugin module or on the client side.
  - New plugins installed in MarkLogic 7 or later should be created in:

```
Assets/plugins/marklogic/manage/extensions
```

- For "resource" plugins, clients must change the version step to `LATEST` (from `v1`). For example, `/manage/LATEST/myplugin`. No changes are required to the plugin code.
- Support is no longer available for the third type of plugin ("view extender"). If you have a plugin of this type, you can rewrite it as a resource plugin in MarkLogic 7 to provide the same functionality.

For details on extending the Management API with plugins, see Extending Management API with Plugins in the *Monitoring MarkLogic Guide*.

Changes to the Packaging API

The Packaging REST API has changed for MarkLogic 7. Applications written using the MarkLogic 6 Packaging REST API (v1) must be rewritten to work with the MarkLogic 7 Packaging REST API (v2).

The differences between the v1 and v2 versions of the Packaging REST API are summarized in the table below.

MarkLogic 6	MarkLogic 7
`/v1/list/package/appserver={name} (GET)`	`/v2/servers/{name}?view=package (GET)` `/v2/packages/{pkgname} (POST)`
`/v1/list/package/database={name} (GET)`	`/v2/databases/{name}?view=package (GET)` `/v2/packages/{pkgname} (POST)`
`/v1/package/compare (GET)`	`/v2/packages/{pkgname}?view=differences (GET)`
`/v1/package/install (POST)`	`/v2/packages/{pkgname}/install (POST)`
`/v1/package-tickets/revert (POST)`	`/v2/tickets/{ticketnumber}/revert (PUT)`

4.22.4.12Changes to the Configuration Manager

The Configuration Manager Packaging feature has been more tightly integrated with the Configuration Manager. Rather than exporting configurations to an XML file, as in MarkLogic 6, configurations in MarkLogic 7 are now exported, as XML, to a ZIP file.

You can import a configuration saved by MarkLogic 6 into MarkLogic 7. However, you cannot import a configuration saved in MarkLogic 7 into MarkLogic 6.

4.22.4.13xdmp:plan Now Requires a Privilege

In MarkLogic 7, the `xdmp:plan` function requires a privilege (`http://marklogic.com/xdmp/privileges/xdmp-plan`). Previously, it did not require a privilege. If you have any code that calls `xdmp:plan` that is run by non-privileged users, you will need to add the privilege to a role that the users are granted (or to a role that they already have).

4.22.4.14fn:analyze-string Now Returns Output in a Different Namespace

In MarkLogic 7, the `fn:analyze-string` function returns an XML node in the `http://www.w3.org/2005/xpath-functions` namespace, which is the namespace specified by the working draft of the latest XQuery specification. Previously, this function was not fully specified and it returned XML in the `http://www.w3.org/2009/xpath-functions/analyze-string` namespace. If you have code that is expecting output in the old namespace, you will need to change that code to expect the new namepsace.

4.23 Incompatibilities Between MarkLogic 6 and 5.0

MarkLogic 8 allows you to upgrade either from MarkLogic 5, MarkLogic 6, or MarkLogic 7. If you are upgrading from 4.2, you must first upgrade to at least MarkLogic 5, and there are some known incompatibilities between 4.2 and 5.0 that are documented in the 5.0 *Release Notes*. If you are upgrading from MarkLogic 6 or later, you can skip this section. For convenience, the incompatibilities between MarkLogic 6 and 5.0 are repeated here, and are as follows:

- 32-bit and Solaris SPARC Platforms are No Longer Supported
- Default Database Settings Changes
- Serialization Changes for xdmp:from-json and xdmp:to-json
- fn:format-dateTime Functions Now Round to 7 Places Instead of 3
- xdmp:sha1, xdmp:sha256, xdmp:sha512 Not Supported in FIPS Mode
- Values of Type xs:date Sometimes Serialize With Timezone
- DocumentInputFormat Default Input Value Class is MarkLogicDocument
- Application Builder Generated App Differences
- app-user Role Now Has the rest-reader Role
- fn:resolve-uri in XSLT Now Requires a Scheme When Using the Second Parameter
- MarkLogic Content Pump 1.0-2 Requires MarkLogic Server 6.0-2
- Starting With 6.0-2, XCC Has a New insertContentCollectErrors() Method on the Java Session Interface
- New Aggregate Input Default URI in mlcp 1.0-3
- As of 6.0-5, HDP is no longer a supported Hadoop platform

4.23.1 32-bit and Solaris SPARC Platforms are No Longer Supported

MarkLogic 6.0 discontinues support for all 32-bit platforms (32-bit Windows and 32-bit Linux) and also discontinues support for Sun Solaris SPARC. 64-bit operating systems are required. Sun Solaris is supported on x64 platforms only. If you are using one of the discontinued platforms, you will need to migrate your environment to a supported platform. For details on supported platforms, see “Supported Platforms” on page 9.

4.23.2 Default Database Settings Changes

There are several changes to the default settings when you create a new database. The changes are to turn off some features that you might not need by default and to enable the URI lexicon by default. If you need the feature, you can enable it when or after you create the database configuration. The following are the new default values for the settings that changed:

- `maintain last modified` is set to `false`
- `directory creation` is set to `manual`

- `inherit permissions` is set to `false`
- `inherit collections` is set to `false`
- `inherit quality` is set to `false`
- `URI lexicon` is set to `true`

If you plan to use WebDAV on a database, then set `directory creation` to `automatic`. If you want any of the other options to behave as they did with the previous defaults, then modify them when you create the database or after you create the database. If you have scripts that relied on the old default values, you will need to update the scripts to modify the values.

4.23.3 Serialization Changes for xdmp:from-json and xdmp:to-json

The `xdmp:from-json` built-in function serializes as xml that is a serialization of a `json:array` in MarkLogic 6, with key/value pairs expressed as `json:objects`. Previously, they were serialized as a `map:map` and as sequences. Also, null values behave differently and are now included in the JSON output from `xdmp:to-json`. JSON array objects are now returned as `json:arrays` instead of sequences. Because of these changes, any code you have that uses `xdmp:to-json` and `xdmp:from-json` will not work the same way as it did in MarkLogic 5, and will need rewriting.

If you have code that uses these functions, consider rewriting it to use the new json:transform-from-json and json:transform-to-json XQuery library modules. For details, see Working With JSON in the *Application Developer's Guide*.

4.23.4 fn:format-dateTime Functions Now Round to 7 Places Instead of 3

In MarkLogic 6, the `fn:format-dateTime` function, as well as all of the other format functions that show fractional seconds, rounds the seconds to 7 fractional digits. Previously, it rounded to 3 fractional digits, so it is now more precise than it used to be. If you have code that relies on the old rounding behavior, you might see different (and more accurate) answers in MarkLogic 6.

4.23.5 xdmp:sha1, xdmp:sha256, xdmp:sha512 Not Supported in FIPS Mode

MarkLogic 6 runs by default in FIPS mode, which require ciphers and other security requirements that are secure according to FIPS 140-2. The `xdmp:sha1`, `xdmp:sha256`, and `xdmp:sha512` functions are not FIPS 140-2 compliant, and are therefore not allowed in the default FIPS mode; in FIPS mode, these functions throw an exception. If you want to use these functions, you will need to turn FIPS mode off. Alternately, you can change your code to use the more secure `xdmp:hmac-sha1`, `xdmp:hmac-sha256`, or `xdmp:hmac-sha512` functions, which are supported in FIPS mode.

4.23.6 Values of Type xs:date Sometimes Serialize With Timezone

In MarkLogic 6, some range index values that have timezones associated with them include the timezone when retrieved through lexicon functions (`cts:element-values`, `cts:element-attribute-values`, and so on). This difference only occurs for date values that have a timezone different from the current timezone, and only when stored in a range index. Previously, those date values did not include the timezone. For example,

```
xquery version "1.0-ml";

(:
   assumes an element range index is
   configured for the element "hello"
:)
xdmp:document-insert("/date.xml",
<hello>{fn:adjust-date-to-timezone(xs:date("2012-08-10"),
        xs:dayTimeDuration("-PT5H"))}</hello>);

cts:element-values(xs:QName("hello"), (), "type=date")

(:
   returns 2012-08-10-05:00 in MarkLogic 6
   returns 2012-08-10 in MarkLogic 5
:)
```

If you have date range index values stored in range indexes and if your code expects those values to not have timezones, then you will need to modify the code to handle timezones when they occur.

4.23.7 DocumentInputFormat Default Input Value Class is MarkLogicDocument

In MarkLogic Connector for Hadoop 1.1 the default input value class for `com.marklogic.mapreduce.MarkLogicDocument` is `com.marklogic.mapreduce.DocumentInputFormat`. In 1.0, the default input value class was `com.marklogic.mapreduce.MarkLogicNode`. To restore the 1.0 behavior, add the following to your MapReduce configuration file:

```
<property>
  <name>mapreduce.marklogic.input.valueclass</name>
  <value>com.marklogic.mapreduce.MarkLogicNode</value>
</property>
```

4.23.8 Application Builder Generated App Differences

The applications generated from Application Builder are considerably different in MarkLogic 6 than they were in MarkLogic 5. In MarkLogic 6, the applications use the REST API and do not have any XQuery code. The following are some of the differences:

- The generated applications use the REST API and are hosted on a REST API instance.
- The generated applications are now 100% html and javascript; there is no XQuery code directly in the application.
- The default quality weight is now 0. Previously, it was 1. The new default value of 0 is the default for the REST API (which applications generated with Application Builder use) makes search performance faster.
- Any customizations built in MarkLogic 5 will be lost if you re-generate the application in MarkLogic 6.

If you have customizations of your generated applications or if you rely on any of the old application features, then you will need to rework those in MarkLogic 6. For details on Application Builder and the applications it generates, see the *Application Builder Developer's Guide*.

4.23.9 app-user Role Now Has the rest-reader Role

In MarkLogic 6, the `app-user` role now inherits the `rest-reader` role. Previously it did not have this role. The REST API uses the `rest-reader` role by default to put permissions on documents, so when you create documents without explicitly specifying permissions using the REST API or the Java API, users with the `rest-reader` role (and therefore uses with the `app-user` role because it inherits `rest-reader`) can read the documents. This should not cause an incompatibility for existing applications, as the `rest-reader` role did not exist in previous versions of MarkLogic, but those users might be able to read new documents created in MarkLogic 6 using the REST API or the Java API. If you have users who have the `app-user` role, be aware that they will have this additional role.

4.23.10 fn:resolve-uri in XSLT Now Requires a Scheme When Using the Second Parameter

If you use `fn:resolve-uri` in an XSLT stylesheet, it now runs in strict mode and therefore, if you specify the second parameter, the second parameter must be a valid base URI and include a scheme; that is, it cannot start with a slash. The following query demonstrates this change:

```
xquery version "1.0-ml";

xdmp:xslt-eval(
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <xsl:value-of select="resolve-uri('../foo','/bar/boo/bat.xml')"/>
</xsl:template>
</xsl:stylesheet>, document{<_/>})
(:
  returns /bar/foo in 5.0, throws
  XDMP-RELURIBASEARG  in 6.0
:)
```

If you have XSLT code that relies on the old behavior, you will have to modify your code so it does not throw `XDMP-RELURIBASEARG`. You can put a scheme (such as `http://`) in the second parameter, call the function in an XQuery `1.0-ml` module, or use another means besides `fn:resolve-uri`.

4.23.11 MarkLogic Content Pump 1.0-2 Requires MarkLogic Server 6.0-2

MarkLogic Content Pump (mlcp) 1.0 should not be used with MarkLogic Server 6.0-2 or later. Use mlcp 1.0-2 or later with MarkLogic Server 6.0-2 or later.

4.23.12 Starting With 6.0-2, XCC Has a New insertContentCollectErrors() Method on the Java Session Interface

Starting with 6.0-2, XCC has a new method on the `Session` interface. If you recompile a program with the 6.0-2 or later jar file, and if that program implements the `Session` interface, then you will need to add the `insertContentCollectErrors()` method to your source code so your interface implements this function. For example, the open source XQSync program implements the `Session` interface, so if you have old source code for it, when you recompile it you will need to implement the new method in its `Session` interface. You should only need to do this if you are recompiling; if you are not recompiling the program, then it should continue to work with the new XCC jar.

4.23.13 New Aggregate Input Default URI in mlcp 1.0-3

The following change might affect you if you use mlcp to load documents from aggregate input (`-input_file_type aggregates`) and rely on the default URI generation. If you use `-aggregate_uri_id` when loading from aggregate files, you should not be affected by this change.

Documents created with earlier versions of mlcp used the following default URI template:

```
hash_or_taskid-seqnum
```

Starting with mlcp version 1.0-3, the default URI template is:

```
/path/filename-split_start-seqnum
```

For details, see Default Document URI Construction in the *mlcp User Guide*.

4.23.14 As of 6.0-5, HDP is no longer a supported Hadoop platform

Starting with 6.0-5, Hortonworks Data Platform (HDP) is no longer a supported Hadoop distribution for use with the MarkLogic Connector for Hadoop or the distributed mode of MarkLogic Content Pump (mlcp).

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

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.

MarkLogic recommends the following two guidelines for server sizing:

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

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

MarkLogic Server requires 1.5 times the disk space of the total forest size. Specifically, each forest on a filesystem requires its filesystem to have at least 1.5 times the forest size in disk space (or, for each forest less than 32GB, 3 times the forest size) when the `merge max size` database merge setting is set to the default of 32GB. This translates to approximately 1.5 times the disk space of the source content after it is loaded. For example, if you plan on loading content that will result in a 200 GB database, reserve at least 300 GB of disk space. The disk space reserve is required for merges. *

It is critical for swap space to be properly configured on your system according to the recommendations for your platform, as described in Memory, Disk Space, and Swap Space Requirements in the *Installation Guide*.

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

* You need at least 2 times the `merge max size` of free space per forest, regardless of the forest size. Therefore, with the default `merge max size` of 32GB, you need at least 64GB of free space. Additionally, if your journals are not yet created, you need 2 times the journal size of free disk space (if the journal space is not yet allocated). Therefore, to be safe, you need (with the default `merge max size` and a 2G journal size) at least 68GB of free space for each forest, no matter what size the forest is.

5.2 Compatibility with XQuery Specifications

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

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

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

5.3 XQuery Extensions

Working within the W3C XQuery 1.0 Recommendation, MarkLogic 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 developer web site at http://developer.marklogic.com/:

Documentation	Description
MarkLogic Server-Side JavaScript Function Reference	API documentation for the Server-Side MarkLogic built-in extensions to the JavaScript standard functions.
MarkLogic XQuery and XSLT Function Reference	API documentation for the MarkLogic built-in and module extensions to the XQuery standard functions, as well as API documentation for the W3C functions implemented in MarkLogic Server.
Getting Started with MarkLogic Server	A quick, step-by-step overview of how to get up and running with MarkLogic Server.
Installation Guide	Provides procedures for installing MarkLogic Server.
Release Notes	Contains a summary of new features, upgrade compatible information, and known issues.
Concepts Guide	Provides an overview of MarkLogic and conceptual information about the server architecture.
Application Developer's Guide	Provides procedures, methodologies, and conceptual information about general MarkLogic Server application development tasks.
Search Developer's Guide	Provides procedures, methodologies, and conceptual information about search-based application development tasks.
Node.js Application Developer's Guide	Provides procedures, methodologies, and conceptual information about developing MarkLogic Server applications using the Node.js Client API.
Java Application Developer's Guide	Provides procedures, methodologies, and conceptual information about developing MarkLogic Server applications using the Java API.
XCC Developer's Guide	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.
MarkLogic Connector for Hadoop Developer's Guide	Provides information on the MarkLogic Connector for Hadoop, a Java library to help you build applications that combine MarkLogic Server and Hadoop map-reduce jobs.

Documentation	Description
REST Application Developer's Guide	Provides information on MarkLogic Server administration and application development using the MarkLogic REST API.
Semantics Developer's Guide	Provides procedures, methodologies, and conceptual information about semantic application development tasks and MarkLogic SPARQL and triple support.
Temporal Developer's Guide	Provides information on developing applications using MarkLogic bi-temporal features.
Reference Application Architecture Guide	Provides an overview of reference application architectures for multi-tiered applications built using MarkLogic as the database.
Administrator's Guide	Provides procedures for administrative tasks such as creating servers, creating databases, backing up databases, creating users, setting up your security policy, and so on.
Scripting Administrative Tasks Guide	Provides information on writing code to script various administrative tasks such as creating and modifying databases, App Servers, and so on.
Database Replication Guide	Provides information on database replication, useful for disaster recovery scenarios.
Flexible Replication Guide	Provides information on Flexible Replication, useful for information sharing from one database to another.
Monitoring MarkLogic Guide	Provides information on monitoring MarkLogic Server, including using the built-in monitoring tools and integrating with external tools such as Nagios and HP Operations Manager.
MarkLogic Connector for SharePoint® Administrator's Guide	Documentation for the MarkLogic Connector for SharePoint®, which allows you to mirror documents from a Microsoft SharePoint repository in MarkLogic Server.
JavaScript Reference Guide	A language reference for the MarkLogic Server-Side JavaScript language. This book includes MarkLogic-specific Object reference, but is not a comprehensive language reference.

Documentation	Description
XQuery and XSLT Reference Guide	A condensed overview of the XQuery language, including information on the three XQuery dialects in MarkLogic Server. This book does include some syntax information, although it is primarily intended as in introduction and quick-reference to the language, not as a comprehensive reference.
mlcp User Guide	A procedural guide that explains how to use MarkLogic Content Pump (mlcp) command line tool to load content into MarkLogic, extract content from MarkLogic, or copy content between databases.
Content Processing Framework Guide	Provides an introduction to the Content Processing Framework and procedures for installing the default content processing framework.
Query Performance and Tuning Guide	Provides performance-related information that is useful to application developers and administrators.
Scalability, Availability, and Failover Guide	Provides information on large-scale system architecture, clustering, availability, and details on setting up shared-disk and local-disk failover.
Understanding and Using Security Guide	Provides information on the role-based security model in MarkLogic Server.
MarkLogic Server on Amazon EC2 Guide	Information about running MarkLogic Server in an EC2 environment.
Query Console User Guide	Provides step-by step information on using Query Console, a tool to create and run arbitrary XQuery code.
Loading Content Into MarkLogic Server Guide	Provides procedures, methodologies, and conceptual information about loading content into MarkLogic Server. Includes an overview of ingestion techniques available using XQuery, Java, REST, .NET, and the MarkLogic Content Pump (mlcp).
SQL Data Modeling Guide	Provides information on how to use MarkLogic's SQL interface, including the creation of relational schemas and views.
Messages and Codes Reference Guide	A reference guide to MarkLogic Server and MarkLogic Application Services error and log messages.

Documentation	Description
Glossary, Copyright, and Support	Includes a glossary of terms as well as copyright and support information.
MarkLogic REST API Reference	API documentation for the REST API.
Java Client API Documentation	API documentation for the MarkLogic Java Client API.
Node.js Client API Reference	API documentation for the MarkLogic Node.js Client API.
XCC Javadoc API Documentation	API documentation for the MarkLogic XML Contentbase Connector for Java API (XCC/J).
XCC .NET C# API Documentation	API documentation for the MarkLogic XML Contentbase Connector for .NET XCC C# API.
MarkLogic Hadoop MapReduce Connector API	API documentation for the MarkLogic Hadoop MapReduce Connector.
C++ UDF API Reference	API documentation for the C++ User Defined Function (UDF) API.
Application Builder Developer's Guide	Provides step-by-step information on using Application Builder to build a search application, and also provides information on customizing applications built with Application Builder.
Information Studio Developer's Guide	Provides step-by-step information on using Information Studio to load and transform content.

XQuery language documentation is provided through the W3C working group drafts specified in "Compatibility with XQuery Specifications" on page 89. 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 MarkLogic developer site (http://developer.marklogic.com).

XQuery language extensions specific to MarkLogic Server are documented online in the *MarkLogic XQuery and XSLT Function 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 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 Admin Interface, Application Builder, the applications generated with Application Builder, and the other GUI tools (Query Console, Information Studio, and so on) are supported on Internet Explorer 11 on Windows 7, 8, and 10 (all Windows platforms), Firefox 43 on Windows and Mac OS, and Chrome 48 on Windows and Mac OS. Other browser/platform combinations may work but are not as thoroughly tested.

6.0 Technical Support

MarkLogic provides technical support according to the terms detailed in your Software License Agreement or End User License Agreement.

We invite you to visit our support website at http://help.marklogic.com to access information on known and fixed issues, knowledge base articles, and more. For licensed customers with an active maintenance contract, see the Support Handbook for instructions on registering support contacts and on working with the MarkLogic Technical Support team.

Complete product documentation, the latest product release downloads, and other useful information is available for all developers at http://developer.marklogic.com. For general questions, join the general discussion mailing list, open to all MarkLogic developers.

7.0 Copyright

MarkLogic Server 8.0 and supporting products.

NOTICE

Copyright © 2018 MarkLogic Corporation.

This technology is protected by one or more U.S. Patents 7,127,469, 7,171,404, 7,756,858, 7,962,474, 8,935,267, 8,892,599 and 9,092,507.

All MarkLogic software products are protected by United States and international copyright, patent and other intellectual property laws, and incorporate certain third party libraries and components which are subject to the attributions, terms, conditions and disclaimers found at http://docs.marklogic.com/guide/copyright/legal.

MarkLogic and the MarkLogic logo are trademarks or registered trademarks of MarkLogic Corporation in the United States and other countries. All other trademarks are property of their respective owners.

For all copyright notices, including third-party copyright notices, see the Combined Product Notices for your version of MarkLogic.