MarkLogic Server

Release Notes

MarkLogic 9
May, 2017

Last Revised: 9.0-1, May, 2017

Table of Contents

Release Notes

1.0 Introduction

MarkLogic 9 is a major release of MarkLogic Server that includes many new features. The new features are described in “New Features in MarkLogic 9” on page 12. The following lists some of the major features with links to where they are described:

- Template Driven Extraction (TDE)
- SQL Enhancements
- Optic APIs
- Enhanced Tiered Storage
- Encryption at Rest
- Element Level Security
- Redaction
- Geospatial Enhancements
- Entity Services API
- Bulk Asynchronous Data Movement

For a description of these and many more new features, see “New Features in MarkLogic 9” on page 12.

If you are upgrading from MarkLogic 8, some applications will require minor changes to run correctly on MarkLogic 9. For details, see “Known Incompatibilities with Previous Releases” on page 24.

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:

2.1 Supported Platforms

MarkLogic Server is supported on the following platforms:

- Microsoft Windows Server 2012 (x64), Windows 7 and Windows 10 64-bit (x64)*
- Red Hat Enterprise Linux 7 (x64)** *** ****
- CentOS 7 (x64)** *** ****
- Amazon Linux (x64)** ***
- Mac OS X 10.11*****

* Microsoft Windows 7 and Windows 10 are supported for development only.
If MarkLogic Server fails to start up on Windows with the error "the application failed to initialize properly (0xc0150002)", then a dependency is missing from your environment and you need to download and install the following DLL for 64-bit versions of Windows: http://www.microsoft.com/downloads/details.aspx?FamilyID=eb4ebe2d-33c0-4a47-9dd4-b9a6d7bd44da&DisplayLang=en. Additionally, if you get an error on startup saying you need MSVCR100.dll, then install the Microsoft Visual C++ 2010 SP1 Redistributable Package (x64) http://www.microsoft.com/en-us/download/details.aspx?id=13523.

** Either `deadline` or `noop` I/O scheduler is required to ensure efficient disk I/O for MarkLogic Server on Linux. You should not use `noop` unless your MarkLogic host has intelligent I/O controllers or is only connected to SSDs. For more details, see http://help.marklogic.com/Knowledgebase/Article/View/8/0/notes-on-io-schedulers.

***The `redhat-lsb-core`, `glibc`, `gdb`, and `cyrus-sasl-lib` packages are required on Red Hat Enterprise Linux. Additionally, on 64-bit Red Hat Enterprise Linux, both the 32-bit and the 64-bit `glibc` packages are required.

****Red Hat Enterprise Linux 7 (x64) is also supported in a VMWare ESXi 6.0 (installed on bare metal) environment.

*****Mac OS X is supported for development only. Conversion (Office and PDF) and entity enrichment are not available on Mac OS X. Mac OS X 10.8 or 10.11 (Mountain Lion through El Capitan) on a 64-bit capable processor is reqired (http://support.apple.com/kb/HT3696).

> **Note:** Red Hat Enterprise Linux 6 is not supported for MarkLogic 9.

2.2 Supported Filesystems

MarkLogic relies on the operating system for filesystem operations. While any filesystem that works properly (including under heavy load) should work, the following table lists the operating systems along with the filesystems under which they are supported. Other filesystems may work but have not been thoroughly tested by MarkLogic.

Operating System	Supported Filesystems
Linux (all varieties)	XFS (recommended), EXT3, and EXT4 as well as the clustered filesystems for shared-disk failover mentioned in Requirements for Shared-Disk Failover in the *Scalability, Availability, and Failover Guide*. **Warning** Do not use `data=writeback` with EXT3 and EXT4 filesystems. NAS is supported on Red Hat Enterprise Linux 7 and NetAPP.
Windows	NTFS
Mac OS	HFS+
All	Hadoop HDFS, Amazon S3 (no journaling with S3)

Additionally, HDFS storage is supported with MarkLogic on the HDFS platforms described in HDFS Storage in the *Query Performance and Tuning Guide*.

Note: The Solaris OS is not certified for MarkLogic 9.

2.3 Upgrade Support

This section describes upgrade support to MarkLogic 9. For details on installing MarkLogic Server and for the upgrade procedure, see the *Installation Guide*.

Warning MarkLogic Early Access does not support upgrade. This section describes upgrade for 9.0-1 and later.

Upgrading is supported from 7.0-6 or later. If you are running a release prior to 7.0, you must first upgrade to MarkLogic 7 or MarkLogic 8 before upgrading to MarkLogic 9. 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.

Note: MarkLogic Corporation strongly recommends performing a backup of your databases before upgrading to MarkLogic 9. Additionally, MarkLogic Corporation recommends that you first upgrade to the latest maintenance release of MarkLogic 7 or MarkLogic 8 before upgrading to MarkLogic 9.

An upgrade from MarkLogic 7 or MarkLogic 8 does not require a reindex. If you are upgrading from a previous release that does require a reindex and you choose not to reindex your databases, the database will run in 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 9 features (as well as earlier features depending upon which compatibility mode it runs) 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 9. You might need to make some minor code changes to your MarkLogic 8 applications before they can run correctly in MarkLogic 9. For details on the incompatibilities, see "Known Incompatibilities with Previous Releases" on page 24. For instructions on upgrading to MarkLogic 9, including information about database compatibility between MarkLogic 8 and MarkLogic 9, see the *Installation Guide*.

3.0 New Features in MarkLogic 9

This chapter describes the new features in MarkLogic 9.

3.1 Template Driven Extraction (TDE)

MarkLogic 9 enables you to define a relational lens over your document data, so you can query parts of your data using SQL or the new Optic API. Templates let you specify which parts of documents make up rows in a view. You can also use templates to define a semantic lens, specifying which values from a document make up triples in the triple index.

For more details, see Template Driven Extraction (TDE) in the *Application Developer's Guide*.

3.2 SQL Enhancements

MarkLogic is a NoSQL database, where the unit of storage and indexing is a document. The document model makes it possible to express rich, related, varying structures — anything from a scientific journal article to a complex derivative trade. Many users want to view parts of these rich structures as though they were simple tables — to see the data in those documents through a relational lens. While it is possible to create and query SQL views with MarkLogic 8, MarkLogic 9 further enhances the SQL capabilities with a number of new features, including Templates, the new Optic API, and an updated ODBC driver.

In MarkLogic 9, you can define a template that specifies which parts of the document make up a row in a view, and then query that view from a server-side program with `xdmp:sql()` or `xdmp.sql()` or via ODBC. You can also query that view server-side from the new MarkLogic Optic API — a fluent JavaScript interface with the ability to perform joins and aggregates on views over documents.

3.3 Optic APIs

The Optic API blends the relational world with rich NoSQL document features by providing the capability to perform joins and aggregates over documents. One of the enabling features, Template Driven Extraction, makes it possible to create a relational lens over documents stored in MarkLogic by using templates to specify the parts of a document that make up a row in a view. You can access that data using the Optic API in XQuery, JavaScript, or Java.

For more details, see Optic API for Relational Operations in the *Application Developer's Guide*.

3.4 Enhanced Tiered Storage

MarkLogic 9 adds more flexibility to the Tiered Storage feature. First, it allows documents to be matched to a tier based on a query, in addition to the current solution that is based on a range index pair of boundary values. That means that any complex query can be used to create a policy to match data or metadata about documents in order to decide in which tier to save or move the document. Second, it allows tiered storage queries to match dates using age, for instance, you can write a query for high-end tier that matches documents created in the last 30 days, and another for mid-end tier that matches documents created between 1 and 3 years, rather than the fixed pair of boundary values in the current solution. Third, it removes the need of using super databases to create tiers, simplifying the life of developers, which were required to understand the tiering policy in order to write high performance queries or any update.

MarkLogic 9 adds more performance to the Tiered Storage feature. Now queries that use elements which are part of the tiering policy are optimized to run against the nodes that are more likely to have that data. That means that if you have a query that uses the same element as the tiering policy, for instance an element range query on create date > 30 days, the query engine will direct the search only to the data nodes that store that data, in this example high-end tier nodes.

3.5 Encryption at Rest

MarkLogic 9 introduces the ability to encrypt “data at rest” - data that is on media (on disk or in the cloud), as opposed to data that is being used in a process. Encryption can be applied to newly created files, configuration files, or log files. Existing data files can be encrypted by triggering a merge or re-index of the data.

For more information about using Encryption at Rest, see Encryption at Rest in the MarkLogic *Security Guide*.

3.6 Element Level Security

Element Level Security is an addition to the MarkLogic security model that allows you to specify more complex security rules for specific elements within documents. Element Level Security can be applied to either JSON or XML documents. Users without the appropriate permissions cannot view the secured element or JSON property.

Element Level Security can be used in addition to the existing document level security and compartment security. For details about using Element Level Security, see Element Level Security in the *Security Guide*.

3.7 Redaction

MarkLogic 9 introduces a new library module and mlcp command line option that enable you to redact sensitive data when extracting documents from the database. Redaction enables you to obscure or hide portions of a document using a rule-based read transformation.

Redaction is available through the following interfaces:

- The mlcp `-redaction` command line option.
- The `rdt:redact` XQuery function.
- The `rdt.redact` Server-Side JavaScript function.

For more details, see Redacting Document Content in the *Application Developer’s Guide* and Redacting Content During Export or Copy Operations in the *mlcp User Guide*.

3.8 Geospatial Enhancements

MarkLogic 9 includes the following enhancements to geospatial operations:

- Geospatial region search, including a new geospatial region index type
- ETRS-89 coordinate system support
- Double precision coordinate support
- Support for kilometers, meters, and feet as units of measure

You can take advantage of these enhancements using all of the MarkLogic geospatial and search related interfaces, including `cts:search`, `search:search`, JSearch, and the REST, Java, and Node.js Client APIs.

> **Note:** The performance of geospatial region queries for geographic coordinate systems (WGS84 and ETRS89) has not yet been fully optimized. Performance will be improved in a future release.

For details, see Geospatial Search Applications in the *Search Developer's Guide*.

3.9 Entity Services API

The Entity Services API enables you to quickly and easily model your business entities and relationships between them and then generate code and configuration artifacts that provide a framework for an entity based application.

The artifacts you generate with the Entity Services API make it easier to create entities from raw source data, query entities and the relationships between them, and manage your entities and models.

For details, see the *Entity Services Developer's Guide*.

3.10 New Stemming and Tokenization

In MarkLogic 9 the default tokenization and stemming code has been changed for all languages (except English tokenization). Some tokenization and stemming behavior will change between MarkLogic 8 and MarkLogic 9. We expect that in most cases results will be better in MarkLogic 9.

You can now customize stemming and tokenization for a language by selecting one of several built-in stemmer and lexer plug-ins, or by creating your own stemmer and lexer plug-ins in C++.

You can also configure a custom stemming and/or tokenization dictionary for any language. Previously, you could not distinguish between stemming and tokenization dictionaries, and you could not install a dictionary for an unsupported language.

For more details, see the following topics:

- Tokenization and Stemming in the *Search Developer's Guide*
- User-Defined Lexer Plugins in the *Search Developer's Guide*
- Using a User-Defined Stemmer Plugin in the *Search Developer's Guide*
- Custom Dictionaries for Tokenizing and Stemming in the *Search Developer's Guide*
- "Default Stemming and Tokenization Libraries Changed for Most Languages" on page 27

3.11 Accurate Wildcard Expansion Options

MarkLogic 9 adds three new cts:query constructor options for controlling wildcard expansion.

When evaluating a wildcarded search term, MarkLogic must sometimes make a tradeoff between speed and accuracy. MarkLogic attempts to expand a wildcard term from the lexicon. To prevent this expansion from taking too long, there is a limit on how many words MarkLogic will extract from the lexicon. When the limit is reached, MarkLogic falls back on alternative strategies that accurate for all possible terms containing wildcard characters.

For many searches, these inaccuracies are an acceptable tradeoff for the fast reponse of an unfiltered search. Where the inaccuracies are not acceptable, you can use the following new options to change the default behavior: `-lexicon-expansion-limit=`*`N`*, `-limit-check`, `-no-limit-check`.

For more details, see the function reference documentation for the following functions:

XQuery	Server-Side JavaScript
`cts:word-query`	`cts.wordQuery`
`cts:element-word-query`	`cts.elementWordQuery`
`cts:element-attribute-word-query`	`cts.elementAttributeWordQuery`
`cts:field-word-query`	`cts.fieldWordQuery`
`cts:json-property-word-query`	`cts.jsonPropertyWordQuery`
`cts:element-value-query`	`cts.elementValueQuery`
`cts:element-attribute-value-query`	`cts.elementAttributeValueQuery`
`cts:field-value-query`	`cts.fieldValueQuery`
`cts:json-property-value-query`	`cts.jsonPropertyValueQuery`

3.12 mlcp Enhancements

The following capabilities have been added to the mlcp command line tool:

- Support for SSL Connections
- Greater Control over Host Connections
- Redaction
- Batch Support for Server-Side Import Transformations
- Ability to Access and Modify Metadata in a Transformation

In addition the mlcp source code is now available for download through the GitHub marklogic-contentpump project. For details, see Accessing the mlcp Source Code in the *mlcp User Guide*.

3.12.1 Support for SSL Connections

The mlcp command line tool can now connect to MarkLogic via an SSL (Secure Socket Layer) connections. For details, see Connecting to MarkLogic Using SSL in the *mlcp User Guide*.

3.12.2 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*.

3.12.3 Redaction

MarkLogic 9 introduces the ability to redact sensitive data when extracting documents from the database with the export or copy commands by specifying redaction rule collections in the -redact option.

For more details, see Redacting Content During Export or Copy Operations in the *mlcp User Guide* and Redacting Document Content in the *Application Developer's Guide*.

3.12.4 Batch Support for Server-Side Import Transformations

Previously, the batch size was always one when applying a server-side transformation during an mlcp `import` or `copy` job. This restriction has been lifted in MarkLogic 9. All documents in a batch are now transformed and inserted into the database as a single statement, greatly improving performance when using a transformation.

3.12.5 Ability to Access and Modify Metadata in a Transformation

Collections, permissions, document quality, and temporal collection 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, temporal collection, and values metadata for its output document(s).

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

3.13 Java Client API Enhancements

The following capabilities have been added to the Java Client API:

- Bulk Asynchronous Data Movement
- Enhanced Temporal Document Support
- Security and Authentication Improvements
- Values Metadata Support
- Row-Based Search
- Geospatial Search Enhancements

3.13.1 Bulk Asynchronous Data Movement

The new Data Movement SDK feature of the Java Client API enables you to move large amounts of data into, out of, or within a MarkLogic cluster asynchronously. These interfaces leverage your entire cluster for scale-out performance. The feature supports streaming, asynchronous, long running data movement tasks.

For details, see the `com.marklogic.client.datamovement` package in the *Java Client API Documentation* and Asynchronous Multi-Document Operations in the *Java Application Developer's Guide*.

3.13.2 Enhanced Temporal Document Support

You can now pass a temporal document URI when creating, updating, and patching temporal documents. This is logical document URI in a temporal collection. For more details, see the methods of `com.marklogic.client.bitemporal.TemporalDocumentManager` that accept a `temporalDocumentURI` parameter and Working with Temporal Documents in the *Developing Applications With the Java Client API*.

If you have sufficient privileges, you can now wipe (completely remove) a temporal document using `TemporalDocumentManager.wipe`.

You can use the Java Client API to protect a temporal document against update, deletion, or wipe for a specified time. See `TemporalDocumentManage.protect`.

3.13.3 Security and Authentication Improvements

You can now use Kerberos or certificate-based authentication to authenticate with MarkLogic. You can also connect to MarkLogic via a SSL (Secure Socket Layer) connection.

For more details, see Authentication and Connection Security in the *Developing Applications With the Java Client API*.

3.13.4 Values Metadata Support

MarkLogic 9 adds the ability to associate key-value metadata with a document. You can use the Java Client API to add, update, delete, and search values metadata. For more details, see Values Metadata in the *Developing Applications With the Java Client API*.

3.13.5 Row-Based Search

You can use the Java Client API to execute a plan produced by the Optic API and receive row-based results in your Java application. For details, see Optic Java API for Relational Operations in the *Developing Applications With the Java Client API*.

3.13.6 Geospatial Search Enhancements

MarkLogic 9 includes enhancements to geospatial search such as region searches, double precision coordinates, additional coordinate systems. These features are exposed through the Java Client API. For more details, see “Geospatial Enhancements” on page 14, Creating Region Queries Using the Client APIs in the *Search Developer’s Guide*, and `StructuredQueryBuilder.geospatial` in the *Java Client API Documentation*.

3.14 Node.js Client API Enhancements

The following capabilities have been added to the Node.js Client API:

- Authentication and Connection Security
- Enhanced Temporal Document Support
- Values Metadata Support
- Geospatial Search Enhancements
- Minimum Distiance on Near Queries

3.14.1 Authentication and Connection Security

You can now use Kerberos or certificate-based authentication to authenticate with MarkLogic. You can also connect to MarkLogic via a SSL (Secure Socket Layer) connection.

For more details, see Authentication and Connection Security in the *Node.js Application Developer’s Guide*.

3.14.2 Enhanced Temporal Document Support

You can now pass a temporal document URI when creating, updating, and patching temporal documents. This is the logical document URI in a temporal collection. For more details, see the methods in the `documents` namespace that accept a `temporalDocument` parameter and Working with Temporal Documents in the *Node.js Application Developer’s Guide*.

If you have sufficient privileges, you can now wipe (completely remove) a temporal document using `TemporalDocumentManager.wipe`.

3.14.3 Values Metadata Support

MarkLogic 9 adds the ability to associate key-value metadata with a document. You can use the Node.js Client API to add, update, delete, and search values metadata. For more details, see the `metadataValues` metadata category in Working with Metadata in the *Node.js Application Developer's Guide*.

3.14.4 Geospatial Search Enhancements

MarkLogic 9 includes enhancements to geospatial search such as region searches, double precision coordinates, additional coordinate systems. These features are exposed through the Node.js Client API. For more details, see "Geospatial Enhancements" on page 14, Creating Region Queries Using the Client APIs in the *Search Developer's Guide*, and `queryBuilder.geospatialRegion` in the *Node.js Client API Reference*.

3.14.5 Minimum Distiance on Near Queries

The queryBuilder.near method now accepts a minimum distance for near queries. Previously, you could only specify a maximum distance. For details, see the *Node.js Client API Reference* and `cts:near-query`.

3.15 Rest Client API Enhancements

The following capabilities have been added to the REST Client API:

- Enhanced Temporal Document Support
- Row-Based Data Evaluation
- Point-in-Time Operations
- Values Metadata Support
- cts:query Support

3.15.1 Enhanced Temporal Document Support

You can now pass a temporal document URI when creating, updating, and patching temporal documents. This is logical document URI in a temporal collection. For more details see Working with Temporal Documents in the *REST Application Developer's Guide* and the `/v1/documents` methods that accept a `temporal-document` parameter.

If you have sufficient privileges, you can now wipe (completely remove) a temporal document using `DELETE /v1/documents?result=wiped`. For more details, see `DELETE:/v1/documents` in the *MarkLogic REST API Reference*.

The new `POST:/v1/documents/protection` method enables you to protect a temporal document against update, deletion, or wipe for a specified time period. For more details, see the *MarkLogic REST API Reference*.

3.15.2 Row-Based Data Evaluation

The new `/rows` service enables you to execute a plan produced by the Optic API and receive results in a variety of row-based formats. For more details, see:

- `GET:/v1/rows` in the *MarkLogic REST API Reference*
- `POST:/v1/rows` in the *MarkLogic REST API Reference*
- Optic API for Relational Operations in the *Application Developer's Guide*

3.15.3 Point-in-Time Operations

Most read and search operations now accept a timestamp request parameter that enables you to make successive requests that will evaluated against the state of the database at a point in time. The timestamp can be obtained from the ML-Effective-Timestamp header returned by the same methods.

For more details, see Performing Point-in-Time Operations in the *REST Application Developer's Guide*.

3.15.4 cts:query Support

You can use a serialized cts:query in place of a structured query in `GET:/v1/search` and `POST:/v1/search`. For more details, see Searching With cts:query in the *REST Application Developer's Guide* and the *MarkLogic REST API Reference*.

3.15.5 Values Metadata Support

MarkLogic 9 adds the ability to associate key-value metadata with a document. You can use the REST Client API to add, update, delete, and search values metadata. You can use the new metadata category `metadata-values` when working with this type of metadata.

For more details, see Working with Metadata in the *REST Application Developer's Guide*.

3.16 Telemetry

Telemetry is part of our continuous effort to provide better and faster support by automating the data collection process required on most support tickets. When Telemetry is enabled, it collects, encrypts, and sends diagnostic and anonymized system-level information about a MarkLogic cluster to a secure MarkLogic destination.

The Telemetry feature collects only system level information, and sends it to a protected and secure location where it can only be accessed by the MarkLogic technical teams, to be used to facilitate troubleshooting and monitor performance. Telemetry does not collect any personally identifiable information, user data or application logs. See Telemetry in the *Monitoring MarkLogic Guide* for more information.

3.17 XQuery 3.x Features

MarkLogic now supports selected features from the XQuery 3.0 and XQuery 3.1 specifications. These features are only available when using the "1.0-ml" XQuery dialect.

For more details, see XQuery 3.x Features in the *XQuery and XSLT Reference Guide*.

3.18 Query Console Enhancements

MarkLogic 9 includes the following new features and enhancements in the Query Console application. For more details, see the *Query Console User Guide*.

- Support for profiling Server-Side JavaScript. The profiling output differs from the XQuery output in being sampling based.
- Auto-complete suggestions. As you type into a query, Query Console displays a list of functions, keywords, and in-scope variables suggests. For functions, you also see the function reference documentation.
- The editor now provides auto-closure of parentheses, quotes, braces, and other grouping symbols.
- Each query has now has its own tab for displaying results. The results of one query are not lost if you switch to another query and run it.
- Query results description differentiates between a result that is a single item versus a sequence of one itme so that you can tell whether or not the query reutrns a sequence.
- Query execution time is displayed along with the results. For example, Query Console will report "Returned sequence of 2 items in 230ms". The display also includes the time delta between the current and previous run.
- You can now select a content source and App Server independent of one another.
- MarkLogic 9 enhances the database Explorer to enable you to view metadata such as collections, permissions, key-value metadata, and document quality, along with the contents of a document. This feature is available when you select a document from the Explorer document list.
- You can reorder queries in a workspace and reorder tabs in the editor using drag-and-drop.
- Query results larger than 5M will be truncated.

3.19 Application Display Environment Customization

Administrators can now customize the display environment of MarkLogic applications such as Query Console and the Monitoring Dashboard in the following ways:

- Display a notification dialog when a user navigates to one of the MarkLogic application pages.

- Display a customized banner across the top of each page of the MarkLogic applications. This enables you to easily differentiate between environments such as production and staging.,

For more details, see Application Display Environment Customization in the *Administrator's Guide*.

3.20 Rolling Upgrades

A rolling upgrade is one way to address the need for highly available clusters under heavy transaction loads to upgrade to a newer version of MarkLogic in a seamless manner. Hosts in a cluster are upgraded one by one, without incurring any downtime in availability or interruption of transactions.

For more details, see Rolling Upgrades in the *Administrator's Guide*.

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 9. 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:

4.1 JavaScript: ValueIterator Replaced By Sequence

The `ValueIterator` interface used to represent sequences of value in MarkLogic 8 has been replaced by the new `Sequence` interface. A `Sequence` is a JavaScript Iterable object. All functions which previously operated on or returned a `ValueIterator` now use a `Sequence` instead.

In many cases, this change is transparent to your code. However, code that depends on the following `ValueIterator` properties and methods must be changed:

- `ValueIterator.next` - Use a `for..of` loop to iterate over a `Sequence`. Use `fn.head` if you just want to pick off the first or only value in a Sequence.
- `ValueIterator.count` - Use `fn.count` instead.
- `ValueIterator.clone` - No longer needed. You can iterate over the same `Sequence` multiple times.

For more details, see Sequence in the *JavaScript Reference Guide* and Sequence in the *MarkLogic Server-Side JavaScript Function Reference*.

4.2 Database Stemming is Off, Word Searches On By Default

In MarkLogic 9, when you create a new database, the `stemmed searches` property is `off` by default. In MarkLogic 8 and earlier, the default is `basic`.

In MarkLogic 9, when you create a new database, `word searches` are enabled by default. In MarkLogic 8 and earlier releases, `word searches` were disabled by default.

To achieve the pre-MarkLogic 9 default behavior, configure your database to turn off `stemmed searches` and set `word searches` to `basic`.

These changes only affect databases you create after upgrading to MarkLogic 9. Databases that exist when you upgrade will retain their previous settings.

4.3 Collection Lexicon and Triple Index Enabled by Default

In MarkLogic 9, a fresh install of MarkLogic 9 will enable the triple index and collection lexicon for all databases. Databases that exist when you upgrade to MarkLogic 9 will retain their previous settings. Any new databases created after upgrading to MarkLogic 9 will have the triple index and collection lexicon enabled.

4.4 XCC .NET API No Longer Available

The XCC .NET interfaces have been removed from MarkLogic. Current users of XCC .NET are encouraged to use the REST Client API to create equivalent interfaces. For details, see the *REST Application Developer's Guide*.

The XCC Java Library continues to be available.

4.5 Changes in Semantic Query Behavior

MarkLogic 9 introduces the following changes to the behavior of semantic queries:

4.5.1 Triple Index and SPARQL Engine Changes

The triple index and SPARQL engine have been changed in MarkLogic 9. Now as part of a query, two triples are considered equal if their subjects, predicates, and objects compare as equal with the SPARQL "=" operator.

Previously, these triples would be treated as two different, non-identical triples:

```
sem:triple(xs:anyURI("http://a"), xs:anyURI("http://b"),
xs:anyURI("http://c")),
sem:triple("http://a", "http://b", "http://c").
```

In MarkLogic 9, values of type `xs:string` and `xs:anyURI` can compare equal if they have the same lexical form according to the SPARQL "=" operator. This functionality is called D-entailment (D as in datatype). The two triples in the example are now considered to be the same, and the de-duplication process removes the duplicate. If you return frequencies, you would see the second triple show up as one extra in the frequencies.

Also note that the query may return either `sem:triple(xs:anyURI("http://a"), xs:anyURI("http://b"), xs:anyURI("http://c"))`, or `sem:triple("http://a", "http://b", "http://c")` as the result since they are considered equal.

4.5.2 Forest IDs Removed From sem:sparql Function

The forest ID options for the `sem:sparql` function, which were part of MarkLogic 7, were removed in MarkLogic 8, but their functionality was kept for backwards compatibility. In MarkLogic 9, the forest ID option will no longer work with `sem:sparql`.

4.6 Triple Count Increased After Inserting Same Data Twice

During a rolling upgrade, while the cluster is in a mixed mode (not all hosts committed to the new version yet) the triple count of semantice triples may be increased after inserting the same data twice. During a rolling upgrade, the MarkLogic 9 triple index is not able to return triples in the correct order for MarkLogic 8 semantics. For this situation to occur, a user would need to have multiple triples that are identical except for the types of the values.

4.7 Database max merge size Now Defaults to 48 GB

The database parameter `max merge size` defaults to 48 GB (49152 MB) for new databases created in MarkLogic 9. Previously, the default was 32 GB. Existing databases will keep whatever `max merge size` setting is configured after upgrading to MarkLogic 9. The new setting reflects advances in storage systems and should be appropriate for most databases.

4.8 Changes to Range Index Reference Resolution

In MarkLogic 9, index reference resolution in range index construction, query constructors, and lexicon operations will now succeed in some cases that would have thrown an `XDMP-RIDXNOTFOUND` error in previous releases. If the index reference contains enough information to unambiguously match an existing index, MarkLogic will now do so. Previously, MarkLogic assumed default values for some “missing” index attributes, resulting in an error.

Code that previously succeeded will continue to do so. Some code that would previously have gotten an error will no longer do so.

For example, suppose you define a geospatial element range index on element `xs:QName("coords")` with type `long-lat-point` and coordinate system `wgs84`, and then refer to the index by just the element QName (`cts:element-geospatial-values(xs:QName("coord"))`). Previously, the reference would have been an error because the default point type (`point`) would have been assumed. Now, the reference resolves correctly as long as there is not a second geospatial element range index on the same QName with different coordinate system or point type.

When an ambiguous range index reference is detected, MarkLogic 9 throws one of the new exceptions `XDMP-RIDXAMBIGUOUS` (range index) or `XDMP-GIDXAMBIGUOUS` (geospatial index).

4.9 Default Stemming and Tokenization Libraries Changed for Most Languages

In MarkLogic 9 the default tokenization and stemming libraries have been changed for all languages (except English tokenization). Consequently, some tokenization and stemming behavior will change between MarkLogic 8 and MarkLogic 9. We expect that, in most cases, stemming and tokenization will be more precise in MarkLogic 9.

If you upgrade to MarkLogic 9 from an earlier version of MarkLogic, your installation will continue to use the legacy stemming and tokenization libraries as the language baseline. Any fresh installation of MarkLogic will use the new libraries. You can change the baseline configuration using `admin:cluster-set-language-baseline`.

> **Note:** Changing the baseline requires a cluster-wide restart and a reindex to avoid stemming and tokenization anomalies.

> **Note:** Use of the legacy libraries is deprecated. These libraries will be removed from MarkLogic in a future release.

Unless you use the legacy language baseline, reindexing is required for content in the following languages:

- Chinese
- Danish
- Dutch, if you want to query with decompounding
- Finish
- German
- Hungarian
- Japanese
- Korean, unless you use decompounding
- Norwegian (Bokmal and Nynorsk) if you want to query with decompounding
- Norwegian (generic ‘no’ lang code), though use of generic ‘no’ is not recommended
- Romanian
- Russian
- Swedish, if you want to query with decompounding
- Tamil
- Turkish

For other languages (except English), you might be able to avoid incompatibilities depending on the nature of your queries, but reindexing is still strongly recommended.

Tokenization and stemming are significantly different for Japanese. Tokenization is significantly different for Chinese (both simplified and traditional). The impact on other languages is more nuanced, but should lead to better results, overall. You might observe some relevance score changes on stemmed searches due to the higher degree of precision. If you require low-level details about the impact on a specific language, please contact MarkLogic Technical Support.

For more information about the new tokenization and stemming support, see “New Stemming and Tokenization” on page 15.

4.10 SQL DESCRIBE No Longer Supported by xdmp:sql

In MarkLogic 9, the SQL `DESCRIBE` function is not supported. MarkLogic does support `DESCRIBE` queries over ODBC, but not from `xdmp:sql()`.

4.11 Application-Specific Logging

MarkLogic 9 now provides application-specific access to logging. Instead of logging all errors to `ErrorLog.txt`, the `xdmp:log` function now writes to an app-server-specific log file - for example `8000_ErrorLog.txt` or `8020_ErrorLog.txt`. Anything event running on the task server is logged to `TaskServer_ErrorLog.txt`. The `ErrorLog.txt` file is now for MarkLogic “system” logging only.

Splitting the log files in this manner enables custormer log files (which could potentionally contain confidential information) to be separated from system log files. This helps ensure the privacy of customer data when they interact with Support and use the new Telemetry feature.

In a Unix environment you can watch the multiple log files using something like this:

```
tail -f /path/file1 /path/file2 etc.
```

4.12 Change to Classification of Some Special Symbol Tokens

The classification of the following symbols has changed for purposes of tokenization. These changes can affect search results.

Symbols	Old Classification	New Classification
spacing accents (5E, 60, A8, AF, B4, B8)	punctuation	diacritic
copyright (A9) registered (AE) degree (B0)	punctuation	symbol
Spanish masculine/feminine ordinals (AA, BA)	punctuation	diacritic
superscript numbers (B2, B3, B9)	punctuation	diacritic
micro (B5)	punctuation	greek
fractions (BC, BD, BE)	punctuation	symbol

4.13 Change to xdmp:user-last-login

The `userid` parameter has been removed from `xdmp:user-last-login`. In MarkLogic 8 the `xdmp:user-last-login` took one parameter (`userid`). If the `userid` was different from the current user, the function returned an empty sequence. If the `userid` was the same as that of the current user, it only returned the last login information.

In MarkLogic 9 the function does not take a parameter. The `xdmp:user-last-login` function only returns the last login information for the current user.

4.14 Changed Interfaces for xdmp:document-insert and xdmp:document-load

These two functions, `xdmp:document-insert` and `xdmp:document-load`, have significantly changed interfaces in MarkLogic 9. The interfaces will still work for purposes of backward compatibility, but are no longer documented.

4.15 search:parse Returns a Different Type for cts:query Output Format

In MarkLogic 8, `search:parse` returned the XML serialization of a cts:query by default. You could also explicitly select this return type by specifying “cts:query” as the value of the `$output` parameter.

In MarkLogic 9, if you explicitly specify “cts:query” as the value of the `$output` parameter, you will now get a cts:query object instead of a serialized query. The default return type from `search:parse` is unchanged.

If your code explicitly sets the output type to “cts:query” and passes the output of `search:parse` straight through to functions such as `search:resolve` or `cts:search`, no code changes are required.

If your code explicitly sets the output type to “cts:query” and then transforms or traverses the output query XML, then you must change your code to use the new “schema-element(cts:query)” for the `$output` parameter of `search:parse`.

For example, the following call generates the XML serialization of a cts:query in MarkLogic 9:

```
search:parse("cat AND dog", (), "schema-element(cts:query)")
```

4.16 Default Client API Search Behavior Change on Port 8000

The defined default search behavior for the Java, Node.js, and REST Client APIs is unfiltered search unless you override the default with your own options. Prior to MarkLogic 9, this default behavior was not honored when you used the Client APIs to interact with MarkLogic through the pre-defined App Server on port 8000.

As of MarkLogic 9, searches via the Client APIs on port 8000 will default to unfiltered search. To get the old behavior, use query options that explicitly specify filtered search.

4.17 JSON Property Scope and Container Queries Match Array Items Differently

In MarkLogic 8, the query constructed by the XQuery function `cts:json-property-scope-query` and the Server-Side JavaScript function `cts.jsonPropertyScopeQuery` matched if it's criteria query matched anywhere within the configured scope. In MarkLogic 9, the behavior has changed for certain JSON property scope queries where the scope property value is an array of objects.

The behavior has changed for searches that have the following characteristics:

- The `query` parameter of the property scope query is an and-query. Such a scope query effectively finds co-occurrences of matches to the and-query criteria within the specified scope.
- In the document to be matched, the value of the scope property is an array of objects. The item type is determined by examining only the first item in the array.

Given this setup, in MarkLogic 9, the query only matches if all the sub-queries of the and-query occur in the same array item. In MarkLogic 8, the query matches even when the and-query matches occur in different array items. All other forms of JSON property scope query are unchanged.

This change makes it possible to construct a JSON property scope query that constrains matches to occurrences with a single array item. In MarkLogic 8, this was not possible.

This change also affects the `container-query` element of a structured query and QBE container queries.

The following example query is of the form affected by this change. It finds co-occurences of "prop1" with value "value1" and "prop2" with "value2" within the scope of "root". (The and-query criteria can be arbitrarily complex.)

Language	Example Query

XQuery	`cts:json-property-scope-query(`
	`  "root",`
	`  cts:and-query((`
	`  cts:json-property-value-query("prop1", "value1"),`
	`  cts:json-property-value-query("prop2", "value2")`
	`  ))`
	`)`

Server-Side JavaScript	`cts.jsonPropertyScopeQuery(`
	`  'root',`
	`  cts.andQuery([`
	`  cts.jsonPropertyValueQuery('prop1', 'value1'),`
	`  cts.jsonPropertyValueQuery('prop2', 'value2')`
	`  ])`
	`);`

If you search the following documents with the query shown above, you get the results shown. The JSON properties that match the and-query criteria are shown in bold.

Sample Document	MarkLogic 8	MarkLogic 9

`// (1) criteria met in different array items`
`{"root": [`
`  { **"prop1": "value1"**, "prop2": "v" },`
`  { "prop1": "v", **"prop2": "value2" }`
`]}`
	Match	No match

`// (2) criteria met in the same array item`
`{"root": [`
`  { **"prop1": "value1"**, **"prop2": "value2"** },`
`  { "prop1": "v", "prop2": "v" }`
`]}`
	Match	Match

`// (3) criteria met in a child property`
`{"root": {`
`  "child": [`
`  { **"prop1": "value1"**, "prop2": "v" },`
`  { "prop1": "v", **"prop2": "value2" }`
`  ]`
`}}`
	Match	Match

Document (1) does not match in MarkLogic 9 because both and-query criteria are not satisfied in the same array item. Document (2) matches in MarkLogic 9 because both and-query criteria are satisfied in the same array item. The Document (3) results are unaffected because the value of the "root" property is not an array of objects.

You can restore the MarkLogic 8 behavior by using an and-query of multiple JSON property scope queries. For example, the following query matches Document (1) in MarkLogic 9:

Language	Example Query

XQuery	`cts:and-query((`
	`  cts:json-property-scope-query(`
	`  "root", cts:json-property-value-query("prop1", "value1")),`
	`  cts:json-property-scope-query(`
	`  "root", cts:json-property-value-query("prop2", "value2"))`
	`))`

Server-Side JavaScript	`cts.andQuery([`
	`  cts.jsonPropertyScopeQuery(`
	`  'root', cts.jsonPropertyValueQuery('prop1', 'value1')),`
	`  cts.jsonPropertyScopeQuery(`
	`  'root', cts.jsonPropertyValueQuery('prop2', 'value2'))`
	`]);`

4.18 REST Client API Incompatibilities

MarkLogic 9 introduces the following backward incompatible changes to the RES Client API.

- keyvalue Service Removed
- Collections in Request Parameters are OR Related

4.18.1 keyvalue Service Removed

The previously deprecated `/keyvalue` is no longer available. That is, the method GET `/v1/keyvalue` is no longer available.

To perform similar operations, use the `/search` service with a structured query or combined query, or the `/qbe` service.

4.18.2 Collections in Request Parameters are OR Related

When you specify multiple collections through the collection request parameter of the following methods, they are now OR related:

- GET and POST /v1/search
- GET and POST /v1/qbe

- GET and POST /v1/values/{name}

The new behavior is consistent with the semantics of `cts:collection-query` and the structured query `collection-query`.

To get AND semantics, construct your own `and-query` of collection URIs as part of a structured or combined query, or by defining a constraint in your query options.

The new behavior differs from previous behavior in the following way:

- In MarkLogic 8.0-6, the GET methods applied AND semantics to multiple collection parameters, while the POST methods applied OR semantics.
- In MarkLogic 8.0-5 and earlier, both GET and POST methods applied AND semantics to multiple collection parameters.

4.19 Java Client API Incompatibilities

MarkLogic 9 introduces the following backwards incompatible changes to the Java Client API:

- Java Client API: Removal of Deprecated Interfaces
- Java Client API: JAR File Name and Maven Artifact Id Change
- Logging Turned Off by Default

4.19.1 Java Client API: Removal of Deprecated Interfaces

Most of the previously deprecated packages, interfaces, classes, and methods of the Java Client API have been removed. The following table lists what has been removed and provides guidance for modifying your code.

In the table below, “c.m.c.” is an abbreviation for “com.marklogic.client.”.

Removed Package, Class, or Method	Alternative
`c.m.c.admin.ServerConfigurationManager`: • `getContentVersionRequests` • `setContentVersionRequests`	Use the following equivalent methods instead. • `getUpdatePolicy` • `setUpdatePolicy`
`c.m.c.admin.TransformExtensionsManager`: • `writeXqueryTransform` • `writeXSLTransform`	Overloads of the listed methods that accept a `paramTypes` map have been removed. Use one of the overloads that does not accept parameter metadata. This metadata is not required to install or use a transform.

Removed Package, Class, or Method	Alternative
`c.m.c.admin.config`	This package, its sub-packages, and all contained classes, interfaces, and types have been removed. This includes the `QueryOptions` and `QueryOptionsBuilder` classes. Create query options using your preferred JSON or XML libraries, read options from a file, or build options as a string. Read and write options using appropriate handles, such as `DOMHandle`, `JacksonHandle`, `FileHandle`, or `StringHandle`.
`c.m.c.document.JSONDocumentManager`: • `getLanguage` • `setLanguage`	Remove your usage. Language specifications are not supported for JSON. Calls to these methods were ignored in MarkLogic 8.
`c.m.c.io.QueryOptionsHandle`	Create query options using your preferred JSON or XML libraries, read options from a file, or build options as a string. Read and write options using appropriate handles, such as `DOMHandle`, `JacksonHandle`, `FileHandle`, or `StringHandle`. This class is no longer needed with the removal of `QueryOptions` and `QueryOptionsBuilder`.
`c.m.c.query`: • `KeyValueDefinition` • `KeyLocator` • `ElementLocator`	The key-value search capability has been removed. Build equivalent queries using a `StructuredQueryDefinition` (JSON property query or element query) or QBE.
`c.m.c.query.QueryManager`: • `newKeyValueDefinition` • `newElementLocator` • `newKeyLocator`	The key-value search capability has been removed. Build equivalent queries using a `StructuredQueryDefinition` (JSON property query or element query) or QBE.
`c.m.c.io.SearchHandle.forceDOM`	Remove your usage. This setting was ignored in MarkLogic 8.
`c.m.c.query.StructuredQueryBuilder`: • `FragmentScope.DOCUMENT`	Use `FragmentScope.DOCUMENTS`.

Removed Package, Class, or Method	Alternative
`StructuredQueryBuilder` (nested classes): • `AndNotQuery` • `AndQuery` • `BoostQuery` • `CollectionConstraintQuery` • `CollectionQuery` • `CustomConstraintQuery` • `DirectoryQuery` • `DocumentFragmentQuery` • `DocumentQuery` • `ElementConstraintQuery` • `GeospatialConstraintQuery` • `LocksQuery` • `NearQuery` • `NotQuery` • `OrQuery` • `OrQuery` • `PropertiesConstraintQuery` • `PropertiesQuery` • `RangeConstraintQuery` • `TermQuery` • `ValueConstraintQuery` • `WordConstraintQuery`	Continue to construct structured queries using `StructuredQueryDefinition` methods, as before. The return type from the query builder methods is always `StructuredQueryDefinition` now. For example, `StructuredQueryDefinition.and()` used to return an `AndQuery`, but now returns a `StructuredQueryDefinition`.
`StructuredQueryBuilder.elementConstraint`	`StructuredQueryBuilder.containerConstraint`

4.19.2 Java Client API: JAR File Name and Maven Artifact Id Change

The JAR file for the Java Client API is now named `marklogic-client-api-`*`version`*`.jar`. Previously, the name was `java-client-api-`*`version`*`.jar`. For example, the JAR file name for version 3.0.6 is **`java`**`-client-api-3.0.6.jar`, but the JAR file name for version 4.0.1 is **`marklogic`**`-client-api-4.0.1.jar`.

The Maven artifact id for the Java Client API is now `marklogic-client-api` instead of `java-client-api`. Update your build dependency configuration files, such as `pom.xml` or `build.gradle`, accordingly.

4.19.3 Logging Turned Off by Default

The Logback library is no longer included in the Java Client API distribution. Logging is now off by default. To re-enable logging, include Logback or another slf4j JAR in your classpath.

4.20 Node.js Client API Incompatibilities

The following incompatible changes have been made to the Node.js Client API:

4.20.1 Changes to Return Value of documents.remove

The method `DatabaseClient.documents.remove` previously returned an object with both a "uri" property and a "uris" property that served the same purpose. The "uri" property has been removed. The return value now has the following form:

```
{ "uris":[uri1, uri2, ...], "removed":true }
```

The value of the "uris" property is now always an array. Previously, if you passed in just a single string as input, `remove` returned just the URI string, rather than an array of one item.

4.20.2 Transaction Creation Returns an Object by Default

Previously, `DatabaseClient.transactions.open` returned a transaction id string by default, and you could use the `withState` parameter to request a transaction object instead. The use of the string form was deprecated as of MarkLogic 8.

As of MarkLogic 9 and Node.js Client API v2.0.0, `DatabaseClient.transactions.open` defaults to returning a transaction object.

To force the previous behavior, set `withState` to `false` when creating a transaction. This setting is deprecated and will be removed in a future release.

4.20.3 Default Search Result Slice is Zero-Based

Previously, the slice clause on search (`queryBuilder.slice`) accepted a one-based starting position and a page length:

```
slice(oneBasedStart,PageLength)
```

As of MarkLogic 9 and Node.js Client API v2.0.0, `queryBuilder.slice` clause behaves like `Array.prototype.slice`. That is, it takes a zero-based starting position and the (zero-based) position after the last result to be retrieved. For example, the following slice call returns the first 5 results:

```
... .slice(0,5) ...
```

Also, you could previously use `slice(0)` to suppress the return of result documents and just retrieve an abbreviated summary. Now, you must include both the start and end positions for the same effect. For example: `slice(0,0)`.

To restore the legacy behavior, use `marklogic.setSliceMode`. Note, however, that this form is deprecated and will be removed in a later release.

> **Note:** The semantics of valuesBuilder.slice are unchanged. This function still accepts as 1-based starting position and a page length.

4.21 Geospatial Region Accessors Can Now Return Double Values

The introduction of support for double precision coordinates in MarkLogic 9 means that some geospatial operations may return different results than in the past.

Previously, geospatial region accessor functions such as the XQuery function `cts:point-latitude` or the Server-Side JavaScript function `cts.pointLatitude` always returned float values. As of MarkLogic 9, these functions can return either single or double precision values, depending on the governing coordinate system. If you do not use a double precision coordinate system, you should not notice a difference.

The following functions are affected.

XQuery Function	JavaScript Function
`cts:point-latitude`	`cts.pointLatitude`
`cts:point-longitude`	`cts.pointLongitude`
`cts:box-west`	`cts.boxWest`
`cts:box-east`	`cts.boxEast`
`cts:box-south`	`cts.boxSouth`
`cts:box-north`	`cts.boxNorth`

For more details, see How Precision Affects Geospatial Operations in the *Search Developer's Guide*.

4.22 User-Defined Function Plugins Must Be Recompiled

The version of MarkLogic's C++ User-Defined Function (UDF) interface has been incremented to accomodate the following changes:

- The `marklogic::Point` class now accepts and returns longitude and latitude values as doubles instead of floats.
- Support for new types of UDF plugins in support of custom stemming and tokenization plugins.

You must recompile your UDF plugins for use with MarkLogic 9. UDFs from an earlier version of MarkLogic will not work in a mixed cluster prior to commiting the new version. During a rolling upgrade, you must finish the upgrade and then recompile your UDFs for use with MarkLogic 9.

4.23 SLES 12 No Longer Supported

In MarkLogic 9, the SUSE Linux Enterprise Server operating system is not supported. If you are using this discontinued platform, you will need to migrate your environment to a supported platform. For details on supported platforms, see "Supported Platforms" on page 9.

4.24 Solaris No Longer Supported

In MarkLogic 9, the Solaris operating system is not supported. If you are using this discontinued platform, you will need to migrate your environment to a supported platform. For details on supported platforms, see "Supported Platforms" on page 9.

4.25 Nagios Plugin No Longer Supported

Support for the Nagios plugin has been discontinued in MarkLogic 9. If you are using this discontinued platform, you will need to migrate your environment to a supported platform. For details on supported platforms, see "Supported Platforms" on page 9.

4.26 Application Builder and Information Studio No Longer Available

The Application Builder and Information Studio applications have been removed from MarkLogic. The info and infodev APIs remain, but they are deprecated; for details, see "info and infodev APIs Deprecated" on page 88.

An Application Builder application deployed on MarkLogic 8 should continue to work after the upgrade to MarkLogic 9. We do not support any mechanism to support redeploying such an app against a MarkLogic 9 node. The methods and modules needed for this process have been removed.

Porting an Application Builder application running in MarkLogic 7 to MarkLogic 9 is not supported by MarkLogic Server. If you have an active maintenance contract, you can contact MarkLogic Technical Support for guidance in porting this application.

4.27 Admin Interface No Longer Selects a Default Schemas Database

As of MarkLogic 9, you can create a database without an associated schemas database. Earlier versions of MarkLogic required you to specify a schemas database when creating a new database. This change has no impact on users creating databases through the Admin API. However, this change affects the database creation page of the Admin Interface as follows:

The Admin Interface no longer automatically selects the pre-defined Schemas database as the schemas database when you create a database through the UI. Instead, you must explicit select a schemas database or else you will create a database with no associated schemas database.

Depending on the uses to which you put the database, some operations might fail if there is no associated schemas database. For example, features such as temporal document management and template driven extraction require a schemas database to be associated with the content database.

4.28 MarkLogic 8 Incompatibilities

This section describes the incompatibilites between MarkLogic 7 and MarkLogic 8. This is here just for convenience; for the MarkLogic 8 *Release Notes*, see docs.marklogic.com/8.0/guide/relnotes. The following are the incompatibilies:

- 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 8.0-5 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

4.28.1 JSON Related Incompatibilities

MarkLogic 9 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

4.28.1.1 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 43.
- 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 45.
- 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 45.
- 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.

4.28.1.2 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*.

4.28.1.3 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.

4.28.1.4 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`.

4.28.1.5 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.

4.28.1.6 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);
```

4.28.1.7 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.

4.28.1.8 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*.

4.28.1.9 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.

4.28.1.10 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*.

4.28.2 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

4.28.2.1 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.

4.28.2.2 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.

4.28.2.3 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.

4.28.2.4 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*.

4.28.2.5 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.

4.28.3 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 41.

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

4.28.3.1 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.

4.28.3.2 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*.

4.28.3.3 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*.

4.28.3.4 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*.

4.28.3.5 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

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*.

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`.

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*.

4.28.3.6 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*.

4.28.3.7 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.

4.28.3.8 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.

4.28.3.9 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();
```

4.28.4 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.

4.28.5 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.

4.28.6 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 Configuring the ODBC Driver on Linux in the *SQL Data Modeling Guide*.

4.28.7 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.

4.28.8 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.

4. If you have extended your application with any customizations, re-deploy those customizations.

5. Test your re-deployed application.

4.28.9 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).

4.28.10 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

4.28.10.1search: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 9, 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 9 using the `$output` parameter as shown below:

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

4.28.10.2Deprecated 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.

4.28.10.3Deprecated 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")
```

4.28.10.4Structured 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*.

4.28.10.5sort-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*.

4.28.11 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`

4.28.12 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`.

4.28.13 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).

4.28.14 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

4.28.14.1Terms 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.

4.28.15 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

4.28.15.1xdmp.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`).

4.28.15.2In 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.

4.28.15.3xdmp.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.

4.28.15.4Serialization 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.

4.28.15.5Change 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)

4.28.15.6Deprecated 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.

4.28.15.7REST 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`).

4.28.15.8Geospatial 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

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>)`

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`
`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`

XQuery	Server-Side JavaScript
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

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")
```

4.28.16 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

4.28.16.1spell.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*.

4.28.16.2xdmp.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`.

4.28.16.3xdmp.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.

4.28.16.4Use 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*.

4.28.16.5CDH 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 supported versions of Hadoop.

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

4.28.16.6 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 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*.

4.28.17 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

4.28.17.1 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.

4.28.17.2The 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.

4.28.17.3extract-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*.

4.28.17.4XCC 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`.

4.28.17.5 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]))
```

4.28.17.6 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.

4.29 MarkLogic 7 Incompatibilites

MarkLogic 9 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:

4.29.1 Incompatibilities Between MarkLogic 7.0-3 and 7.0-2

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

4.29.1.1 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).

4.29.1.2 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`

4.29.1.3 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`

4.29.1.4 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.

4.29.2 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.

4.29.3 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

4.29.3.1 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.

4.29.3.2 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.

4.29.4 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

4.29.4.1 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 9:

- `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 *Security Guide*.

4.29.4.2 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.

4.29.4.3 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.

4.29.4.4 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*.

4.29.4.5 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.

4.29.4.6 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`.

4.29.4.7 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)
```

4.29.4.8 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.

4.29.4.9 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 Path or Root 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"
	}}
	}}

4.29.4.10mlcp 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
```

4.29.4.11REST 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 9, 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)`

4.29.4.12Changes 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.

4.29.4.13xdmp: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).

4.29.4.14fn: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.

5.0 Planning for Future Upgrades

This chapter provides guidelines and warnings for preparing for changes expected in a future release, such as announcements of deprecated interfaces. You are not required to make changes related to the topics in this section at this time, but you should plan to do so in the future.

5.1 Packaging API Deprecated

In MarkLogic 9, the Packaging API has been deprecated. It will be removed from the product in MarkLogic 10.

5.2 info and infodev APIs Deprecated

The XQuery library modules in the `info` and `infodev` namespaces are deprecated as of MarkLogic 9 and will be removed from the product in a future release. For example, the functions `info:ticket` and `infodev:ticket-create` are deprecated.

5.3 Annotated Query Output from search:parse Deprecated

The `search:parse` XQuery function and `search.parse` Server-Side JavaScript functions can return an annotated `cts:query` if you pass in `"cts:annotated-query"` as the output format parameter value. As of MarkLogic 9, use of `"cts:annotated-query"` is deprecated. Support for this format will be removed in a future release.

If you currently use the annotated query output as an intermediate step in a transformation, you should use the structured query (`"search:query"`) output format instead. Runtime modification of queries is a primary use case for structured query. For more details, see Searching Using Structured Queries in the *Search Developer's Guide*.

If you currently use the annotated query output format to recover the original query text using `search:unparse`, you should cache the original query text yourself.

5.4 Search API Grammar Customization Deprecated

Customizing the string query grammar through the Search API `grammar` query option is now deprecated. Support for this feature will be removed in a future release.

If your application currently relies on a Search API grammar customization, you should consider alternatives such as the following:

- xqysp (http://github.com/mblakele/xqysp) for XQuery.

- PEG.js (http://pegjs.org/) or Jison (http://github.com/zaach/jison) for Server-Side JavaScript.

6.0 Other Notes

This section provides the following information about MarkLogic Server:

6.1 Memory and Disk Space Requirements

MarkLogic Server requires at least 2 GB 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 48 GB, 3 times the forest size) when the `merge max size` database merge setting is set to the default of 48 GB. 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 48 GB, you need at least 96 GB 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 100 GB of free space for each forest, no matter what size the forest is.

6.2 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*.

6.3 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.

6.4 SQL Queries

This section lists the SQL queries that are known not to work in 9.0-1 and those that are not yet fully optimized.

- Sub-select / sub-query (with any operator - `IN`, `NOT IN`, `=`, `>`, etc) is not optimized.
- Queries that return tens of thousands of rows are not optimized. Use `LIMIT` to restrict the number of rows returned, or focus your query with more filters in the `WHERE` clause or a `cts:query` to restrict by collection, date range, etc. If you are using a BI Tool, configure it to avoid an unrestricted `SELECT * FROM` table.
- Special use of the `MATCH` keyword is retained in MarkLogic 9.0-1, but `MATCH` queries run filtered (`MATCH` does not use the Universal Index).
- `FULL OUTER JOIN` is not supported in 9.0-1

- SQL is read-only in 9.0-1. The only way to create a persistent view is by means of a template view or a range view. You can't create a view over views and save it. And you can't create a `TEMP` table. Applications, such as BI Tools, that rely on creating `TEMP` tables to cache results need to be configured to use some other method.

6.5 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 and upgrade compatible information.
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.

Documentation	Description
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.
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.
Entity Services Developer's Guide	Provides procedures, methodologies, and conceptual information about developing MarkLogic Applications using entity relationship modeling and the Entity Services API.
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 HP Operations Manager.

Documentation	Description
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.
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.
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).

Documentation	Description
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.
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.

XQuery language documentation is provided through the W3C working group drafts specified in "Compatibility with XQuery Specifications" on page 91. 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.

6.6 Browser Requirements

The Admin Interface and the other GUI tools (Query Console, Monitoring Dashboard, and so on) are supported on Internet Explorer 11 on Windows 7 and Windows 10, Firefox 45 on Windows and Mac OS, and Chrome 53 on Windows and Mac OS. Other browser/platform combinations may work but are not as thoroughly tested.

7.0 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.

8.0 Copyright

MarkLogic Server 9.0 and supporting products.
Last updated: April 28, 2017

COPYRIGHT

This technology is protected by U.S. Patent No. 7,127,469B2, U.S. Patent No. 7,171,404B2, U.S. Patent No. 7,756,858 B2, and U.S. Patent No 7,962,474 B2, US 8,892,599, and US 8,935,267.

The MarkLogic software is protected by United States and international copyright laws, and incorporates certain third party libraries and components which are subject to the attributions, terms, conditions and disclaimers set forth below.

For all copyright notices, including third-party copyright notices, see the Combined Product Notices.