MarkLogic Server

Release Notes

MarkLogic 10
May, 2017

Last Revised: 10.0-5, September, 2020

Table of Contents

Release Notes

1.0 Introduction

MarkLogic 10 is a major release of MarkLogic Server that includes many new features. The new features are described in “New Features in MarkLogic 10” on page 12. The following lists some of the major features with links to where they are described:

- JavaScript Engine Upgrade
- Element Level Security (ELS) in the Triple Index
- Machine Learning
- Security Library Upgrades

For a description of these and many more new features, see “New Features in MarkLogic 10” on page 12.

If you are upgrading from MarkLogic 9 or earlier, some applications will require minor changes to run correctly on MarkLogic 10. For details, see “Known Incompatibilities with Previous Releases” on page 17.

1.1 Bug Fixes

To review the list of bug fixes included between any two version of MarkLogic Server, click on the “Fixed Bugs” link (https://help.marklogic.com/Bugtrack/List) that is available on the MarkLogic Suppport Portal (https://help.marklogic.com/), and enter the version range you are interested in.

2.0 Installation and Upgrade

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

- Supported Platforms
- Supported Filesystems
- Upgrade Support

2.1 Supported Platforms

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

2.2 Supported Filesystems

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

2.3 Upgrade Support

This section describes upgrade support to MarkLogic 10. 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 10. 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 10. Additionally, MarkLogic Corporation recommends that you first upgrade to the latest maintenance release of MarkLogic 7 or MarkLogic 8 before upgrading to MarkLogic 10.

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 10 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*.

Note: If you are upgrading clusters with DB replication configured, see Upgrading Clusters Configured with Database Replication in the *Database Replication 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 10. You might need to make some minor code changes to your MarkLogic 8 applications before they can run correctly in MarkLogic 10. For details on the incompatibilities, see "Known Incompatibilities with Previous Releases" on page 17. For instructions on upgrading to MarkLogic 10, including information about database compatibility between MarkLogic 8 and MarkLogic 10, see the *Installation Guide*.

If you are upgrading to MarkLogic 9.0-4 or later, you may have to install MarkLogic Converters package separately. For more details, see MarkLogic Converters Installation Changes Starting at Release 9.0-4 in the *Installation Guide*.

If you upgrade to MarkLogic 9.0-5 or later from an earlier version of MarkLogic 9 and you use a geospatial region index, you need to reindex. For more details, see Understanding Tolerance in the *Search Developer's Guide*.

3.0 New Features in MarkLogic 10

This chapter describes the new features in MarkLogic 10.

3.1 JavaScript Engine Upgrade

In MarkLogic 10, the JavaScript engine has been upgraded to V8 version 6.7. For more details on the new language features, please see Google V8 JavaScript Engine and Converting JavaScript Scripts to Modules.

3.2 Element Level Security (ELS) in the Triple Index

In MarkLogic 10, we have extended support for element-level security (ELS) to include the triple index, meaning it can now be leveraged by semantic graphs and SQL. In semantics, individual triples can be protected. In SQL, this allows you to enable column-level security by protecting specific columns in a Template (TDE).

3.3 Machine Learning

The Cognitive Toolkit (CNTK) library has the concept of a default device. This sets the default computation device (CPU or GPU) for the API. Some functions have a device parameter that allows you to override the default, but not all. The default device has been set based on the version:

- The GPU-enabled version of MarkLogic Server has the default device set to GPU (0).
- The CPU-enabled version of MarkLogic Server has the default device set to CPU.

The default device is enabled during node startup. On GPU enabled instances, it is an exclusive lock. CNTK uses cooperative locking for the device access, whereby only a single process can acquire a device lock. This locking mechanism allows CNTK processes to avoid device oversubscription only if they collectively choose to do so. In other words, the device locked by one CNTK process can still be accessed by another CNTK process without acquiring any locks (the existing device lock can be ignored by other CNTK processes). This cooperative locking mechanism does not guarantee any kind of exclusive access to the device. The proper way to ensure exclusivity is to use the NVIDIA System Management Interface (`nvidia-smi`) provided by NVIDIA.

Beginning with version 10.0-2 of MarkLogic Server, the CNTK machine learning libraries are loaded dynamically based on the hardware detected at server start time. The GPU-enabled version of MarkLogic Server has the default device set to GPU (0). The CPU-enabled version of MarkLogic Server has the default device set to CPU.

> **Note:** Starting with version 10.0-2 of MarkLogic Server, on Linux, we no longer have separate *GPU-enabled* and *CPU-enabled* versions. There is only a single installation RPM file. On Windows, however, we still use separate MSI installation files.

3.4 Security Library Upgrades

The following security-related libraries have been upgraded:

- OpenSSL has been upgraded to version 1.0.2s. For more information, please see the list of changes here.
- Kerberos has been upgraded to version 1.17.
- SoftHSM has been upgraded to version 2.5.0.
- OpenLDAP has been upgraded to version 2.4.46.
- SASL has been upgraded to version 2.1.27.
- SoftHSM library has been upgraded from version 2.2.0 to version 2.5.0.

3.5 Triggers and Amps Creation

Starting in 9.0-7 for triggers and 10.0-2 for amps, Database names can be used in the trigger and amp creation apis, thus making it easy to support the same functionality on replica clusters for databases with the same names.

3.6 Default Assignment Policy for New Databases

Starting in MarkLogic Server version 10.0-2, the default setting for assignment policy for new databases is “Segment.” Databases created with previous versions of MarkLogic will retain their original assignment policy following an upgrade. After the upgrade to 10.0-2, all new databases will have “Segment” as the assignment policy.

3.6.1 MarkLogic Supports ECDH Key Exchange for SSL/TLS

In MarkLogic 10.0-1, ECDH is a supported cipher for SSL/TLS communication. SSL/TLS works if an ECDH cipher is specified.

3.7 New Features in MarkLogic 10.0-2

- Security Improvements
- Usability Improvements
- Performance Improvements

3.7.1 Security Improvements

Added support for for Azure Key Vault External KMS. For details, see Using MarkLogic Encryption with Microsoft Azure Key Vault in our *Security Guide*.

Upgraded to version 1.0.2s of the OpenSLL library.

We now use Argon2 for passphrase Key Derivation Function (KDF).

3.7.2 Usability Improvements

Machine Learning using the CNTK API now has support for a single CPU amd GPU on Linux, as well as granular CNTK built-in privileges.

Request Monitoring has been enhanced with: support for triggers; support for a default application server on ports 8000 and 8002. For more details, see Monitoring Requests in our *Query Performance and Tuning Guide*.

Support for Azure Identity to access storage blob.

Support for Database names for amps.

3.7.3 Performance Improvements

The internal SQL Optimizer has been improved in the following areas:

- OR operators are now more efficient
- Support has been added for Power BI “inverse filters.”
- SPARQL Query performance has been improved.

3.8 New Features in MarkLogic 10.0-3

- Usability Improvements
- Performance Improvements

- Other Changes

3.8.1 Usability Improvements

Support for ONNX Runtime API has been added in both JavaScript and XQuery See the Machine Learning with the ONNX API chapter in our *Application Developer's Guide*.

Language codes are now supported in JSON content. MarkLogic now allows natural language in JSON to be tagged with a language other than the default database language.

The MarkLogic SPARQL engine now supports negated property paths as defined in the W3C 1.1 recommendations, allowing users to query graphs with more flexibility.

The granular privilege `create-user-privilege` has been added to enable giving users limited privileges. For more information, see Enabling Non-Privileged Users To Assign Roles in the *Security Guide*.

3.8.2 Performance Improvements

The performance has been improved in both our SQL and the SPARQL internal engines.

3.8.3 Other Changes

Swap space is automatically configured when running MarkLogic Server on Amazon Web Services (AWS). Swap space is configured during the system startup process with the MARKLOGIC_AWS_SWAP_SIZE configuration variable. For more details, see AWS Configuration Variables and Deployment and Startup in the *MarkLogic Server on Amazon Web Services (AWS) Guide*.

The CNTK API is now deprecated and may be removed in a future release. For any new Machine Learning application projects, developers should use the ONNX Runtime API embedded in our server. For more details, please see the Why Using ONNX Runtime in MarkLogic Makes Sense section in our *Application Developer's Guide*.

The Managed Cluster feature supports SSL-enabled clusters. For details, see The Managed Cluster Feature in the *MarkLogic Server on Amazon Web Services (AWS) Guide*.

3.9 New Features in MarkLogic 10.0-4

- Rolling Upgrade Status Added
- Permissions Change for Updating Temporal Collections LSQT Properties
- ODBC Cursor Support

3.9.1 Rolling Upgrade Status Added

MarkLogic 10.0-4 now has an Upgrade tab in the Admin Interface. During an upgrade, click the Upgrade tab to view the upgrade status of each host in the cluster. For more details, see Rolling Upgrade Status in Admin UI in the *Administrator's Guide*.

3.9.2 Permissions Change for Updating Temporal Collections LSQT Properties

The permissions for changing the temporal collection LSQT properties now only requires "admin/temporal" rights. The scope of this change is within RMA. Previously full admin rights to the database were required.

3.9.3 ODBC Cursor Support

ODBC now supports cursors making it more memory efficient on the client by default. Customers should update to the latest ODBC driver.

3.10 New Features in MarkLogic 10.0-5

- Packages by Linux Platform Updated
- IAM Permissions Updated

3.10.1 Packages by Linux Platform Updated

Updated the list of packages required for each supported Linux platform. For more details, see Supported Platforms and Appendix: Packages by Linux Platform in the *Installation Guide for All Platforms*.

3.10.2 IAM Permissions Updated

Updated the minimum required IAM permissions to create and delete a stack. For more details, see Creating an IAM Role in the *MarkLogic Server on Amazon Web Services (AWS) Guide*.

4.0 Known Incompatibilities with Previous Releases

Most applications implemented on previous versions of MarkLogic will run either without modifications or with very minor modifications in MarkLogic 10. This section summarizes product changes that can cause compatibility issues for applications developed using previous releases and includes the following topics:

4.1 Incompatibilities (External)

4.1.1 No NVidia GPUs drivers available for CentOS 8

Currently there are no NVidia GPU drivers available on CentOS 8. Azure running CentOS 8 will not be able to use GPUs. See https://docs.microsoft.com/en-us/azure/virtual-machines/extensions/hpccompute-gpu-linux/ for more information.

4.1.2 The libnsl.so.1 library is required for CentOS 8

For CentOS 8, MarkLogic has a dependency on `libnsl.so.1()`. You must install this library when using CentOS 8. You can either rely on `yum` to pull in the dependency automatically, or install it manually.

4.1.3 DHF mlUpdateIndexes fails when database index json payload has language tag

Entity Services in MarkLogic 10.0-2 has a change whereby a language tag is added to database properties. In Data Hub, the `mlUpdateIndexes` fails when database index json payload has language tag. The workaround is to delete the language tag, and you should be able to update the indexes using `mlUpdateIndexes` gradle task.

4.2 Incompatibilities between 10.0-5 and 10.0-4

4.2.1 Entity Services Features Deprecated

The XML representation of an entity type has been deprecated in 10.0-5.

The following Entity Services functions have been deprecated in 10.0-5 because DHF provides the equivalent functionality. These functions will not be supported in MarkLogic 11.

JavaScript	XQuery
es.databasePropertiesGenerate	es:database-properties-generate
es.extractionTemplateGenerate	es:extraction-template-generate
	es:optional
es.piiGenerate	es:pii-generate
es.schemaGenerate	es:schema-generate
es.searchOptionsGenerate	es:search-options-generate

4.3 Hadoop Connector Removed

The Hadoop Connector was deprecated starting with MarkLogic 10.0-4 and is removed from the product in MarkLogic 10.0-5.

4.4 Ops Director Deprecated

The Ops Director feature is deprecated in MarkLogic 10.0-5.

4.5 Reindexing After 10.0-5 Upgrade

A fix made in 10.0-5 may cause reindexing after the server is upgraded to 10.0-5. If you have a server configuration where a named field in a database is configured for value searches and stemming is turned *off*, while the database itself has stemming is turned *on*, reindexing will occur after the upgrade to 10.0-5 is completed.

4.6 Incompatibilities between 10.0-4 and 10.0-3

4.6.1 mlcp Distributed Mode Deprecated

mlcp distributed mode has been deprecated for 10.0-4.

4.6.2 HDFS Support Deprecated

HDFS support has been deprecated in MarkLogic 10.0-3.

4.6.3 Priority to Determine MarkLogic License Key

MarkLogic prioritizes the license key value from the following locations in the following order. The version number determines which locations take precedence.

> **Note:** The mlcmd.conf location is only considered when MarkLogic is running on Amazon Web Services (AWS).

10.0-4:

1. Admin Interface (server.xml)
2. /etc/marklogic.conf
3. mlcmd.conf (AWS CloudFormation template user data)

10.0-3:

1. mlcmd.conf (AWS CloudFormation template user data)
2. /etc/marklogic.conf
3. Admin Interface (server.xml)

4.6.4 Entity Services Features Deprecated

The following Entity Services functions were deprecated in 10.0-4 because DHF provides the equivalent functionality. These functions will not be supported in MarkLogic 11.

JavaScript	XQuery
es.instanceConverterGenerate	es:instance-converter-generate
es.instanceFromDocument	es:instance-from-document
es.instanceGetAttachments	es:instance-get-attachments
es.instanceJsonFromDocument	es:instance-json-from-document
es.instanceXmlFromDocument	es:instance-xml-from-document
es.modelFromXml	es:model-from-xml
es.modelGetTestInstances	es:model-get-test-instances
es.modelToXml	es:model-to-xml
	es:add-attachments
	es:copy-attachments
	es:serialize-attachments
	es:init-instance
	es:init-source
	es:init-translation-source
es.versionTranslatorGenerate	es:version-translator-generate
	es:extract-array
	es:with-namespace

4.7 Incompatibilities between 10.0-3 and 10.0-2

- Privilege escalation allowing execution of xdmp:data-directory()
- The API trgr:triggers-change-modules-database() Only Has Two Parameters
- Configuration Manager Removed

4.7.1 Privilege escalation allowing execution of xdmp:data-directory()

Certain privileged built-in functions, for example `xdmp:data-directory` and `xdmp:list-cache-size`, that return static environmental data may be prematurely optimized out with the results in-lined as literals in code passed to `xdmp:invoke-function` or `xdmp:spawn-function`.

The in-lining occurs in the outer environment not the inner environment, so the execution privileges checked are of the outer environment not the inner environment. As a result, even if the inner environment does not have privileges to execute the built-in functions, no exception is thrown when the optimized code is run. This issue is addressed in 10.0-3.

4.7.2 The API trgr:triggers-change-modules-database() Only Has Two Parameters

The API `trgr:triggers-change-modules-database()` only takes two parameters in 10.0-3.

1. the old database as `xs:anyAtomicType`. This can be an ID `unsignedLong` or a name `xs:string`

2. the new database as `xs:anyAtomicType`. This can be a database's ID `unsignedLong` or a database's name `xs:string`.

For more details and sample code, see our API documentation.

4.7.3 Configuration Manager Removed

The Configuration Manager tool is deprecated starting with MarkLogic 9.0-5 and removed from MarkLogic Server in version 10.0-3.

4.8 Incompatibilities between 10.0-2 and 10.0-1

4.8.1 Module and Library Removed

Beginning with version 10.0-2, the `mustache.xqy` module and library existing under the folder `MarkLogic/Modules/MarkLogic/mustache/` is no longer available.

4.8.2 Encryption Change

Data encrypted by the embedded KMS in MarkLogic 10.0-2 cannot be decrypted using versions prior to MarkLogic 10.0-2. Exporting the internal wallet requires a passphrase that is now encrypted by argon2, where previously the files were encrypted with crypt. Additionally, backups encrypted with only a passphrase in MarkLogic 10.0-2 cannot be restored in versions prior to 10.0-2 for the same reason.

4.9 Incompatibilities Between 10.0-1 and MarkLogic 9

4.9.1 Change to Admin Role

In MarkLogic 9, a user with the `admin` role was automatically able to see template driven extraction (TDE) views that were read-protected. In MarkLogic 10, a user with the `admin` role cannot see TDE views unless explicitly granted read access.

4.9.2 XQuery 0.9-ml Deprecated

The XQuery `0.9-ml` dialect has been deprecated and will eventually no longer be supported. MarkLogic recommends that you port any legacy `0.9-ml` XQuery code to either enhanced XQuery dialect (`1.0-ml`) or strict XQuery dialect (`1.0`). For more information, see Porting 0.9-ml XQuery Code to Enhanced 1.0-ml in the *XQuery and XSLT Reference Guide*.

Template Driven Extraction (TDE) functions are also affected by this deprecation. The function `cts:valid-index-path` may throw error messages if you do not upgrade your code to XQuery dialect `1.0-ml` or `1.0`.

4.9.3 JavaScript: Calling the seal() Function on MarkLogic Wrapped Objects Now Raises an Error

The following code will now throw an error because fn.doc("/doc-1.json").toObject()[0].root.d is a MarkLogic wrapped object.

```
var r = fn.doc("/doc-1.json").toObject()[0].root.d
  r.beforeSeal = "hello"
    var before =  r.beforeSeal;
      Object.seal(r)
=>
Error running JavaScript request: TypeError: Cannot seal
```

4.9.4 Certificate Authority Handling Changed During Upgrade

When you upgrade to MarkLogic Server 10, the CAs defined by External Securities are merged into their appropriate App Server's CA list. This consolidates the functionality of where an App Server admin defines CAs to accept as the signer of client certificates.

4.9.4.1 Change in Default rest-reader and rest-writer Permissions

To enable the creation of a document by a user who does not have `rest-reader` or `rest-writer` privileges, the following backward-incompatible changes were made to the REST API to improve security:

Prior to MarkLogic 10.0-1, when inserting documents, the REST API assigned permissions based on the default permissions configured for the user and role but also assigned read permissions to the `rest-reader` role and assigned update permissions to the `rest-writer` role. As a result, any user with the `rest-reader` role had permission to read to read all documents and any user with the `rest-writer` role had permission to update all documents.

In MarkLogic 10.0-1, when inserting documents, the REST API assigns permissions based only on the default permissions configured for the user and role. As a result, it is possible to adopt a security model in the REST API where no role has access to all documents.

In other words MarkLogic 9, when you wrote documents with the REST API using PUT v1/documents, the documents had the union of the following permissions:

- Any permissions specified in the request
- The default permissions for the user and document or, when update policy is set to overwrite-metadata, the existing permissions on the document
- Read permission for the rest-reader role and update permissions for the rest-writer role

In MarkLogic 10, when you write documents with the REST API using PUT v1/documents, the documents have the union of the following permissions:

- Any permissions specified in the request
- The default permissions for the user and document or, when update policy is set to overwrite-metadata, the existing permissions on the document

What has changed is that the `rest-reader` and `rest-writer` convenience roles no longer have any permissions on a document unless one of the following is true:

- The request specifies permissions for the `rest-reader` or `rest-writer` role.
- The definition of the user grants default permissions to the `rest-reader` or `rest-writer`.

Since the `rest-writer` convenience role default permissions grant reader permission to the `rest-reader` role and update permissions to the `rest-writer` role, documents written by a user who has the `rest-writer` convenience role are readable by users with the rest-reader role and writable by users with the `rest-writer` role.

- In MarkLogic 9, a user given the `rest-reader` or `rest-writer` role had access to every document written with the REST API.
- In MarkLogic 10, a security model need not grant any role access to every document written with the REST API. Documents inserted by users with the `rest-writer` role still

have read permissions for the `rest-reader` role and update permissions for the `rest-writer` role.

To override this backward incompatibility, you must modify the user role creating documents to give default permissions to `rest-writer` and `rest-reader`.

4.9.5 Changed Functions

The following functions have been changed as of MarkLogic 10:

- sec:create-external-security

  This function no longer takes the arguments `client-certificate-authorities` and `require-client-certificate`.

For more information, see External Certificate User Authentication in the *Security Guide*.

4.9.6 Deprecated Functions

The following functions have been deprecated as of MarkLogic 10 and will be removed in a future release:

- `external-security-set-ssl-client-certificate-authorities`
- `external-security-set-ssl-require-client-certificate`
- `external-security-get-ssl-client-certificate-authorities`
- `external-security-get-ssl-require-client-certificate`

Please use the following functions instead:

- `appserver-set-ssl-client-certificate-authorities`
- `appserver-set-ssl-require-client-certificate`
- `appserver-get-ssl-client-certificate-authorities`
- `appserver-get-ssl-require-client-certificate`

4.10 MarkLogic 9 Incompatibilities

This section describes the incompatibilities between MarkLogic 8 and MarkLogic 9. This is here just for convenience; for the MarkLogic 9 *Release Notes*, see docs.marklogic.com/9.0/guide/relnotes. The following are the incompatibilities:

- JavaScript: ValueIterator Replaced By Sequence
- Database Stemming is Off, Word Searches On By Default
- Collection Lexicon and Triple Index Enabled by Default
- XCC .NET API No Longer Available

4.10.1 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*.

4.10.2 Database Stemming is Off, Word Searches On By Default

In MarkLogic 9 and later, 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 and later, 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 `true`.

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

4.10.3 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.

4.10.4 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.

4.10.5 Changes in Semantic Query Behavior

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

- Triple Index and SPARQL Engine Changes
- Forest IDs Removed From sem:sparql Function

4.10.5.1 Triple Index and SPARQL Engine Changes

The triple index and SPARQL engine were 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 and later, 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.

4.10.5.2 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`.

4.10.6 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 semantic triples may be increased after inserting the same data twice. During a rolling upgrade, the MarkLogic 9 and later 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.

4.10.7 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 and later. 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.

4.10.8 Changes to Range Index Reference Resolution

In MarkLogic 9 and later, index reference resolution in range index construction, query constructors, and lexicon operations 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 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).

4.10.9 Default Stemming and Tokenization Libraries Changed for Most Languages

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

If you upgrade to MarkLogic 9 or later 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 and you have an active maintenance contract, you can contact MarkLogic Technical Support.

For more details on incompatibilities related to the changes to stemming and tokenization, see the Knowledge Base article entitled "MarkLogic Server v9 Tokenization and Stemming" from MarkLogic Technical Support, available at the following URL:

https://help.marklogic.com/knowledgebase/article/View/484

For more information about the new tokenization and stemming support, see "Default Stemming and Tokenization Libraries Changed for Most Languages" on page 30.

4.10.10 SQL DESCRIBE No Longer Supported by xdmp:sql

In MarkLogic 9, the SQL `DESCRIBE` function was no longer supported. MarkLogic does support `DESCRIBE` queries over ODBC, but not from `xdmp:sql()`.

4.10.11 Application-Specific Logging

MarkLogic 9 and later provide 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 customer log files (which could potentially 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.
```

4.10.12 Change to Classification of Some Special Symbol Tokens

The classification of some symbols changed for purposes of tokenization as of MarkLogic 9, including the symbols in the following table. 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

4.10.13 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.

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

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

4.10.15 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 and later, 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 and later:

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

4.10.16 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 default to unfiltered search. To get the old behavior, use query options that explicitly specify filtered search.

4.10.17 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 its criteria query matched anywhere within the configured scope. In MarkLogic 9 and later, the behavior 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 and later, 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-occurrences 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 previous query shown, 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 and later because both and-query criteria are not satisfied in the same array item. Document (2) matches in MarkLogic 9 and later 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 and later:

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'))`
	`]);`

To ensure the modified queries match only when the and-query matches occur in the same instance of the scope property (“root” in the above examples), wrap it in JSON property scope query on the parent property. For example:

Language	Example Query

XQuery	`cts:json-property-scope-query("parent-of-root",`
	`  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.jsonPropertyScopeQuery('parentOfRoot',`
	`  cts.andQuery([`
	`  cts.jsonPropertyScopeQuery(`
	`  'root', cts.jsonPropertyValueQuery('prop1', 'value1')),`
	`  cts.jsonPropertyScopeQuery(`
	`  'root', cts.jsonPropertyValueQuery('prop2', 'value2'))`
	`]));`

4.10.18 REST Client API Incompatibilities

MarkLogic 9 introduced the following backward incompatible changes to the REST Client API.

- keyvalue Service Removed
- Collections in Request Parameters are OR Related
- Default value of Document Management “repair” parameter changed

4.10.18.1keyvalue 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.

4.10.18.2Collections 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.

4.10.18.3 Default value of Document Management “repair” parameter changed

The default value of the `repair` parameter in the Document Management PUT /v1/documents function has changed in the following way:

- In Marklogic 8.0 and earlier, the default value for `repair` is `full`.
- In Marklogic 9.0 and later, the default value for `repair` is `none`.

4.10.19 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

4.10.19.1Java 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.

Note: In the following table, "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.
`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`.

Removed Package, Class, or Method	Alternative
`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`

4.10.19.2Java 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.

4.10.19.3Logging 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.

4.10.20 Node.js Client API Incompatibilities

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

- Changes to Return Value of documents.remove
- Transaction Creation Returns an Object by Default
- Default Search Result Slice is Zero-Based

4.10.20.1Changes 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.

4.10.20.2Transaction 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.

4.10.20.3Default 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.

4.10.21 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*.

4.10.22 User-Defined Function Plugins Must Be Recompiled

The version of MarkLogic's C++ User-Defined Function (UDF) interface has been incremented to accommodate 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 committing the new version. During a rolling upgrade, you must finish the upgrade and then recompile your UDFs for use with MarkLogic 9.

4.10.23 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 10.

4.10.24 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 10.

4.10.25 Nagios Plugin No Longer Supported

Support for the Nagios monitoring plugin has been discontinued in MarkLogic 9.

You can create your own MarkLogic monitoring integration using the MarkLogic REST Management API; for details see the *Monitoring MarkLogic Guide*.

MarkLogic does not endorse or support any particular 3rd party monitoring integrations. However, the MarkLogic open source community includes integrations with 3rd party monitoring platforms such as New Relic and App Dynamics. For more details see:

- New Relic: https://github.com/marklogic-community/newrelic-plugin
- App Dynamics: https://github.com/Appdynamics/marklogic-monitoring-extension

4.10.26 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 101.

An Application Builder application deployed on MarkLogic 8 will 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 or later 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.

4.10.27 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 explicitly 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.

4.10.28 Internal Security ON with External Security Object Behavior Change

This section describes some changes related to external security support. You should be aware of these changes, but they do not introduce incompatibilities or necessitate a change to your application.

In MarkLogic 9 and later, you can log into port 7001 on an LDAP account when “internal security” on appserver port 7001 is set to “true” or “false”. In previous versions of MarkLogic, you can only log into port 7001 on an LDAP account if “internal security” on appserver port 7001 is set to “false”.

In MarkLogic 9 and later, you can assign multiple external security objects to an App Server. When there are multiple external security objects assigned, a MarkLogic user is authenticated and assigned to an external security based on the order in which the external security objects are assigned. In previous version of MarkLogic, you could only assign one external security object to an App Server.

If internal security is enabled for an App Server, then when a user attempts to authenticate with MarkLogic, MarkLogic first checks to see if the user is in the security database. If so, then MarkLogic verifies the credentials against the security database. When this verification fails, the behavior of MarkLogic 8 and earlier versions differs from the behavior of MarkLogic 9 and later as follows:

- In MarkLogic 8 and earlier, if Security database verification fails, then the login attempt fails with an error.
- In MarkLogic 9 and later, if Security database verification fails, then MarkLogic attempts to authenticate the user against any external security objects assigned to the App Server. If there are no external security objects assigned or if the user cannot be authenticated against any assigned external security objects, then the login fails with an error.

Thus, when internal security is enabled, there can be cases where a login will succeed in MarkLogic 9 that would have failed with earlier versions. The behavior is unchanged if there are no external security objects assigned to the App Server or internal security is disabled.

4.10.29 REST Management API Changes in MarkLogic 9

In MarkLogic 9, the following REST Management API methods were deleted and their functionality was moved elsewhere. The following table lists the deleted methods and the new alternative.

Endpoint Deleted in 9.0	Alternative
`DELETE /manage/v2/credentials`	`DELETE:/manage/v2/credentials/properties`
`GET /manage/v2/credentials`	`GET:/manage/v2/credentials/properties`
`PUT /manage/v2/credentials`	`PUT:/manage/v2/credentials/properties`
`GET /manage/v2/databases/{id|name}/sub-databases`	`GET:/manage/v2/databases`
`POST /manage/v2/databases/{id|name}/sub-databases`	`POST:/manage/v2/databases`
`GET /manage/v2/databases/{id|name}/super-databases`	`GET:/manage/v2/databases`
`POST /manage/v2/databases/{id|name}/super-databases`	`POST:/manage/v2/databases`
`DELETE /manage/v2/databases/{sub-id-or-name}/super-databases/{super-id-or-name}`	`DELETE:/manage/v2/databases/{id|name}`
`GET /manage/v2/databases/{sub-id-or-name}/super-databases/{super-id-or-name}`	`GET:/manage/v2/databases/{id|name}/properties`
`DELETE /manage/v2/databases/{super-id-or-name}/sub-databases/{sub-id-or-name}`	`DELETE:/manage/v2/databases/{id|name}`
`GET /manage/v2/databases/{super-id-or-name}/sub-databases/{sub-id-or-name}`	`GET:/manage/v2/databases/{id|name}/properties`

4.10.30 Configuration Packaging Format Incompatibilities

Configuration packages created with the Packaging REST API and/or with the Configuration Packaging XQuery library module are not compatible with the configuration format introduced in MarkLogic release 9.0-5.

These configuration packages cannot be used with the CMA REST API and/or with the Configuration Management API XQuery/JavaScript library modules.

For more details, see the following sections:

- "Configuration Management API (CMA) XQuery and JavaScript Libraries" on page 46;
- "Configuration Management API (CMA) REST Endpoints" on page 46;
- "Packaging API Removed" on page 101;
- "Configuration Packaging XQuery Library Deprecated" on page 106.

4.10.31 Configuration Management API (CMA) XQuery and JavaScript Libraries

MarkLogic 9.0-5 introduced library functions for configuration management.

The configuration management functions can be used to:

- retrieve a configuration of an individual resource, a set of resources, or a full cluster;
- generate a configuration from scenarios, such as High Availability (HA) scenario;
- apply a named configuration, overriding parameters and setting options.

Use the following functions for configuration management:

XQuery	Server-Side JavaScript
`cma:generate-config`	`cma.generateConfig`
`cma:apply-config`	`cma.applyConfig`

For more details, see the *MarkLogic XQuery and XSLT Function Reference* and the *MarkLogic Server-Side JavaScript Function Reference*.

4.10.32 Configuration Management API (CMA) REST Endpoints

MarkLogic 9.0-5 introduced a REST API for configuration management: Configuration Management API (CMA).

The Configuration Management API is a RESTful API that allows retrieving, generating, and applying configurations for MarkLogic clusters, databases, and application servers.

Use the following endpoints for configuration management:

Endpoint	Description
`GET /manage/v3`	This endpoint enables retrieving configuration of an individual resource, a set of resources, or a full cluster. It also enables generating new configurations from scenarios.

Endpoint	Description
`POST /manage/v3`	This endpoint enables applying named configurations to MarkLogic resources, overriding parameters and setting options. The configurations may be applied to an individual resource, a set of resources, or a full cluster.

For more details, see the *MarkLogic REST API Reference*.

4.10.33 Java Client API 4.1.1 Incompatibilities

Version 4.1.1 of the Java Client API introduced the following incompatibilities with earlier Java Client API versions:

- Load Balancer Configuration for DMDSDK Jobs

4.10.33.1Load Balancer Configuration for DMDSDK Jobs

This change does not affect you if you do not connect to MarkLogic through a load balancer or if you do not use the Data Movement SDK (DMSDK) feature of the Java Client API.

You must change the configuration of any `DatabaseClient` objects used to connect to MarkLogic through a load balancer on behalf of a Data Movement SDK job. The following changes apply:

- You must add a connection type parameter value of `GATEWAY` when configuring a DatabaseClient.
- Do not use a `FilteredForestConfiguration` to configure connections through a load balancer.

For more details, see Working with a Load Balancer in the *Java Application Developer's Guide*.

4.10.34 MarkLogic SQL ORDER-BY Keyword

Starting in MarkLogic 9.0-9, `ORDER-BY` ordering on NULL is set to `NULLS LAST`. This changes default behavior for nulls in query results and therefore may cause backwards incompatibility for certain queries (unless the `Optic Nulls Smallest On` trace event is turned on).

4.10.35 Changes to Accepted XML Character Set

As of MarkLogic 9.0-6, parsing of XML documents with an XML declaration that explicitly specifies XML version 1.1 (version="1.1") enforces the XML 1.1 character set. Consequently, you can now create content containing characters not accepted by XML 1.0.

Characters in the XML 1.1 "restricted character" ranges must be given as character entities. This enforcement applies to the following character ranges:

- 0x1-0x8

- 0xB-0xC
- 0xE-0x1F
- 0x7F-0x84
- 0x86-0x9F

The following character ranges that were previously disallowed are now accepted.

- 0x1-0x8
- 0xB-0xC
- 0xE-0x1F

Serializing content that contains characters allowed by XML 1.1 but not XML 1.0 can cause problems for client layers that cannot handle the XML 1.1 character set. You can control whether to serialize XML as version 1.0 or 1.1 by setting the output version in your App Server output options, using methods such as the following:

- Admin Interface: Set the output version to "1.0" under `Groups` > *`yourGroup`* > `App Servers` > *`yourAppServer`* > `Output Options`.
- Admin API: Call `admin:appserver-set-output-version` with a value of "1.0" for the $value parameter.
- REST Management API: Send a request to `PUT:/manage/v2/servers/{id|name}/properties` that sets the `output-version` property to "1.0".

4.10.36 Minimum Required Version of HDP is 2.6

As of MarkLogic 9.0-5, if you are using the Hortonworks Data Platform (HDP) with the following MarkLogic technologies, you must use HDP v2.6.

- mlcp
- HDFS
- The MarkLogic Connector for Hadoop

4.10.37 Reindex Recommended for Geospatial Region Indexes

This change affects only database configuration that define a geospatial region index.

MarkLogic 9.0-5 introduces support for a "tolerance" option on geospatial region queries. As a consequence, if you use geospatial region indexes, your region queries might not be accurate until you reindex. This is applicable whether or not you take advantage of the new tolerance option.

4.10.38 Geospatial Region Query Results Might Differ

As of MarkLogic 9.0-5, MarkLogic uses the coordinate system *default* tolerance when evaluating a geospatial region query. Previously, MarkLogic used the coordinate system *minimum* tolerance.

Consequently, some region queries might not give the same results after upgrading to MarkLogic 9.0-5. Use the "tolerance" option to adjust your region queries, if necessary.

To learn more about tolerance, see Understanding Tolerance in the *Search Developer's Guide*.

4.10.39 return-query Option Output Format Change

This change only affects users of the REST, Java, and Node.js Client APIs who use the `return-query` query option and generate a search response summary in JSON.

In prior versions of MarkLogic, the `return-query` query option returned a serialized JSON structured query in the `query` property of the search response in JSON, but a serialized cts:query in an XML search response. As of MarkLogic 9.0-5, `return-query` always generates a serialized cts:query. The serialized JSON representation can be passed to `cts:query` or `cts.query` to reconstruct an in-memory query object on MarkLogic, just as you can with the XML serialization.

This change will not affect you if you do not use a JSON search response, do not use `return-query`, or do not have code that depends on the serialization produced by `return-query`.

4.10.40 Redaction: Deterministic Masking Values Differ

MarkLogic 9.0-4 introduced support for salting in the generation of masking values by the `mask-deterministic` redaction function. This means the same text redacted with `mask-deterministic` in earlier versions of MarkLogic will not produce the same masking value by default when redacted using MarkLogic 9.0-4.

To preserve the previous behavior, modify your `mask-deterministic` rules to set the `salt` and `extend-salt` options to an empty value.

For more details, see mask-deterministic in the *Application Developer's Guide*.

4.11 Incompatibilities Between 9.0-2 and 9.0-3

The following incompatibilities exist between MarkLogic 9.0-2 and MarkLogic 9.0-3:

- Changes to Authentication Behavior with Client Certificate
- XCC ContentSource.newSession Interface Change
- Document Digest Authorization Behavior Changed in 9.0-3
- 1-click AMIs, new compatible CloudFormation, and additional upgrade procedures
- map:new Retains Keys with Empty Values

4.11.1 Changes to Authentication Behavior with Client Certificate

MarkLogic Server 9.0-3 authenticates using both a client certificate and a username/password. This provides a greater level of security, by requiring that the user provide a client certificate that matches the specified user. In MarkLogic 9.0-2, if the password is incorrect, but the user has the correct client certificate, and the "ssl require client certificate" is false, authentication will succeed. In MarkLogic 9.0-3 if the password is incorrect, but the user has the correct client certificate and "ssl require client certificate is false", authentication fails.

The "ssl require client certificate" is an appserver configuration. With the appserver, there are different authentication options: basic, digest, application-level, certificate, and kerberos-ticket. This change only applies to basic, digest or application-level authentication. This behavior only happens when "ssl require client certificate" is set to false

In previous versions, MarkLogic would accept username/password for any internal user, once MarkLogic verified that certificate was signed by the selected CA (certificate authority). MarkLogic 9.0-1 and 9.0-2 authentication is restricted to an internal user with a matching common name (CN). In MarkLogic 9.0-3, the username does not need to match the common name (CN).

The behavior in MarkLogic 9.0-3 is the same as it is in MarkLogic 8.0. For more information, see the Certificate topic in the *Security Guide*.

4.11.2 XCC ContentSource.newSession Interface Change

The XCC method `ContentSource.newSession` has been extended to include a new overload that accepts a `char[]` value for the password parameter, rather than a `String`, to enable more secure handling of passwords.

As a consequence of this change, passing `null` in the password parameter of `newSession` now results in a compile time error because the compiler cannot determine the correct overload without type information. Use a typecast to disambiguate your call.

4.11.3 Document Digest Authorization Behavior Changed in 9.0-3

Prior to MarkLogic 9.0-3, the server did not check the nonce for digest authentication. In MarkLogic 9.0-3, the server checks the nonce, verifies that it is a valid nonce, and verifies that the URI in the Authorization header is same as the request URI.

4.11.4 1-click AMIs, new compatible CloudFormation, and additional upgrade procedures

To support 1-click deployment in AWS Marketplace, MarkLogic 9 AMIs have data volume pre-configured (device on `/dev/sdf`). To be compatible with 1-click AMIs, new CloudFormation templates are released on https://developer.marklogic.com/products/cloud/aws. To upgrade existing clusters, new 1-click compatible CloudFormation templates are required.

If custom templates or scripts are used, additional steps may be required to handle blank data volume that is part of the new marketplace AMIs. Please find more details about the upgrade procedure on https://developer.marklogic.com/products/cloud/aws.

4.11.5 map:new Retains Keys with Empty Values

As of MarkLogic 9.0-4, the `map:new` function retains any input key-value pair whose value is an empty sequence. Previous versions of MarkLogic 9 and versions of MarkLogic 8 prior to 8.0-7 discarded such key-value pairs.

For example:

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

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

4.11.6 The mlcp Option -tolerate_errors is Ignored

The `-tolerate_errors` option of the mlcp `import` command is deprecated. As of MarkLogic 9.0-2, mlcp ignores this option and always behaves as if `-tolerate_errors` is set to true. The option will be removed in a future release.

4.11.7 Changes to jsearch.facets Output Structure

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

As of MarkLogic 9.0-2, the structure of each facet is an array of arrays instead of a JSON object if and only if you include an explicit `orderBy` clause in your facet definition. If you do not use orderBy, the output is unchanged.

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

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

In MarkLogic 9.0-1, if you use `xdmp.x509CertificateGenerate` to generate a certificate, and the configuration object includes array-valued properties, the array values were encoded as a single string. As of MarkLogic 9.0-2, the array type is preserved. This change applies to any Relative Distinguished Names (RDNs) within a Distinguished Name (DN), such as the `issuer` and `subject` DNs.

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

```
var certObj = {
  version: "2",
  serialNumber: "BA0195369CD6B679",
  issuer: {
```

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

If you round trip the generated certificate through `xdmp.x509CertificateExtract`, you will see the following output for `issuer.organizationName` in MarkLogic 9.0-1 vs. MarkLogic 9.0-2.

```
// Marklogic 9.0-1
organizationName: "[\"MarkLogic\", \"Mark Logic\"]"

// MarkLogic 9.0-2 and later
organizationName: ["MarkLogic", "Mark Logic"]
```

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

4.11.9 Node.js Client API: valuesBuilder.slice is Now Zero-Based

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

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

As Node.js Client API v2.0.3, the `valuesBuilder.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) ...
```

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

4.11.10 Changes to xdmp:update XQuery Prolog Option

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

As of MarkLogic 9.0-2, setting the option to “false” tells MarkLogic to treat the code as a query transaction. This could cause your program to get an error if you explicitly set the option to “false” and your code performs an update operation.

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

For related changes, see the following topics:

- “xdmp:transaction-mode XQuery Prolog Option Deprecated” on page 103
- “Deprecation of transaction-mode Option to xdmp:eval” on page 104

4.11.11 Java Client API 4.0.2 Ignores HttpClientConfigurator

As of version 4.0.2, the Java Client API uses OkHttp as its HTTP client for communicating with MarkLogic over HTTP. This change should be transparent to most applications, but imposes the following backward incompatibility on applications that customize their HTTP configuration:

Attaching a configurator based on `HttpClientConfigurator` to a `DatabaseClientFactory` object no longer has any effect on the HTTP configuration. Use the new `com.marklogic.client.extra.okhttpclient.OkHttpClientConfigurator` interface instead. The `HttpClientConfigurator` interface is deprecated and will be removed in a future release.

4.12 MarkLogic 8 Incompatibilities

This section describes the incompatibilities 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 incompatibles:

- JSON Related Incompatibilities
- Semantics Incompatibilities
- REST and Java Client API Incompatibilities
- 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

4.12.1 JSON Related Incompatibilities

MarkLogic 10 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 will end up being more efficient, but will require you to do some minor code changes. This section lists the incompatibilities 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

4.12.1.1 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 searchable 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 56.
- 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 58.
- 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 58.
- 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 and you have an active maintenance contract, you can contact MarkLogic Technical Support.

4.12.1.2 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*.

4.12.1.3 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.

4.12.1.4 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`.

4.12.1.5 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.

4.12.1.6 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);
  ```

4.12.1.7 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.

4.12.1.8 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*.

4.12.1.9 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 explicitly scoped to an array item or the property value. 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 occurrence 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.

4.12.1.10 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*.

4.12.2 Semantics Incompatibilities

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

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

4.12.2.1 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.

4.12.2.2 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.

4.12.2.3 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.

4.12.2.4 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 *Semantic Graph Developer's Guide*.

4.12.2.5 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.

4.12.3 REST and Java Client API Incompatibilities

This section covers the incompatibilities 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 54.

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

4.12.3.1 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.

4.12.3.2 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*.

4.12.3.3 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*.

4.12.3.4 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*.

4.12.3.5 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*.

4.12.3.6 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*.

4.12.3.7 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.

4.12.3.8 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.

4.12.3.9 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();
```

4.12.4 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 and you have an active maintenance contract, you can contact MarkLogic Technical Support.

4.12.5 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.

4.12.6 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*.

4.12.7 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.

4.12.8 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.

4.12.9 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).

4.12.10 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

4.12.10.1search: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 10, 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 10 using the `$output` parameter as shown below:

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

4.12.10.2Deprecated 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.

4.12.10.3Deprecated 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")
```

4.12.10.4Structured 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*.

4.12.10.5sort-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*.

4.12.11 Locks and Properties Query Built-In Functions Renamed

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

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

4.12.12 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 applications that rely on `text/json` to identify JSON documents; they need the Mimetype changed to `application/json`.

4.12.13 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).

4.12.14 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

4.12.14.1 Terms Matched by additional-query Are Highlighted in Snippets

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

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

4.12.15 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

4.12.15.1 xdmp.multipartDecode Now Returns a JSON Payload for Headers

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

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

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

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

- `thsr.queryLookup`

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

4.12.15.3xdmp.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.

4.12.15.4Serialization 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.

4.12.15.5Change to Required Java Version

The following tools and libraries that depend on a Java Runtime Environment (JRE) now require at least Java 8, rather than Java 7:

- mlcp

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

  Note: The IBM JRE is not supported.

4.12.15.6Deprecated 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.

4.12.15.7REST 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`).

4.12.15.8Geospatial 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`

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

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")
```

4.12.16 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

4.12.16.1spell.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*.

4.12.16.2xdmp.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`.

4.12.16.3xdmp.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.

4.12.16.4Use 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*.

4.12.16.5 CDH 4.3 is No Longer a Supported Hadoop Distribution

The MarkLogic features and tools that rely on Hadoop no longer support CDH 4.3. Instead, use one of the newer supported versions of Hadoop.

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

4.12.16.6 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*.

4.12.17 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

4.12.17.1 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.

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

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

4.12.17.3 extract-document-data Results Now Inline By Default

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

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

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

4.12.17.4XCC 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`.

4.12.17.5Client 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]))
```

4.12.17.6 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.

4.13 MarkLogic 7 Incompatibilities

MarkLogic 10 allows you to upgrade either from MarkLogic 5, MarkLogic 6, or MarkLogic 7. If you are upgrading from 4.2, you must first upgrade to at least MarkLogic 5, and there are some known incompatibilities between 4.2 and 5.0 that are documented in the 5.0 *Release Notes*. If you are upgrading from MarkLogic 7, you can skip this section. For convenience, the incompatibilities between MarkLogic 6 and MarkLogic 7 are repeated here, and are as follows:

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

4.13.1 Incompatibilities Between MarkLogic 7.0-3 and 7.0-2

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

- HDP No Longer a Supported Hadoop Platform
- Java API: ContentVersionRequest Property Deprecated
- REST API: content-versions Property Deprecated
- REST API: JSON documents Cannot be Retrieved as XML
- Float Precision Greater in 7.0-3

4.13.1.1 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).

4.13.1.2 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

4.13.1.3 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`

4.13.1.4 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.

4.13.2 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.

4.13.3 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

4.13.3.1 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.

4.13.3.2 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.

4.13.4 Incompatibilities Between MarkLogic 7.0-1 and MarkLogic 6

4.13.4.1 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 10:

- `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*.

4.13.4.2 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.

4.13.4.3 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.

4.13.4.4 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*.

4.13.4.5 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.

4.13.4.6 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`.

4.13.4.7 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)
```

4.13.4.8 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.

4.13.4.9 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 Properties
- Modify Queries to Use Both Namespaces
- Create a Field That Spans Both Namespaces

Modify Pre-Existing Properties

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 bold 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 file system 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"
	}}
	}}

4.13.4.10mlcp 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
```

4.13.4.11REST 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 Plug-ins
- 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 Plug-ins

There have been some changes to the Management API plug-ins for MarkLogic 10, 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), plug-ins are installed in the following folder:

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

Plug-ins installed to this directory will continue to work with the following exceptions:

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

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

- For "resource" plug-ins, 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 plug-ins, 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)`

4.13.4.12Changes 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.

4.13.4.13xdmp: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).

4.13.4.14fn: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 menopause.

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 XQuery 0.9-ml Deprecated

The XQuery 0.9-ml dialect has been deprecated and will no longer be supported in a future release. MarkLogic recommends that you port any legacy 0.9-ml XQuery code to either enhanced XQuery dialect (1.0-ml) or strict XQuery dialect (1.0). For more information, see Porting 0.9-ml XQuery Code to Enhanced 1.0-ml in the *XQuery and XSLT Reference Guide*.

Template Driven Extraction (TDE) functions are also affected by this deprecation. The function cts:valid-index-path may throw error messages if you do not upgrade your code to XQuery dialect 1.0-ml or 1.0.

5.2 Packaging API Removed

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

For information about the deprecated endpoints and the alternative endpoints to use instead, see

The following table lists the deprecated endpoints and the new alternative.

Deprecated Endpoints	Alternative
Base path `/manage/v2/packages`: • `GET /manage/v2/packages` • `POST /manage/v2/packages` • `GET /manage/v2/packages/{pkgname}` • `POST /manage/v2/packages/{pkgname}` • `HEAD /manage/v2/packages/{pkgname}` • `DELETE /manage/v2/packages/{pkgname}` • `GET /manage/v2/packages/{pkgname}/databases` • `GET /manage/v2/packages/{pkgname}/databases/{name}` • `POST /manage/v2/packages/{pkgname}/databases/{name}` • `HEAD /manage/v2/packages/{pkgname}/databases/{name}` • `DELETE /manage/v2/packages/{pkgname}/databases/{name}` • `POST /manage/v2/packages/{pkgname}/install` • `GET /manage/v2/packages/{pkgname}/servers` • `GET /manage/v2/packages/{pkgname}/servers/{name}` • `POST /manage/v2/packages/{pkgname}/servers/{name}` • `HEAD /manage/v2/packages/{pkgname}/servers/{name}` • `DELETE /manage/v2/packages/{pkgname}/servers/{name}`	Use the following endpoints of the CMA REST API instead: • GET /manage/v3 • POST /manage/v3
Base path `/manage/v2/tickets`: • `POST /manage/v2/tickets/{ticketnumber}/revert`	

5.3 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.

5.4 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.

5.5 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.

5.6 Search API searchable-expression Deprecated

Due to security and performance considerations, beginning in MarkLogic 9.0-10, the `searchable-expression` property/element in query options is deprecated.

In addition, in 9.0-10 and moving forward, the `searchable-expression` requires the `eval-search-string` privilege.

Before MarkLogic 11, Search API users should modify queries that use query options with searchable expressions to remove the `searchable-expression`. The replacement is as follows:

- For the parts of the searchable expression resolvable with indexes, use query clauses instead
- For the parts of the searchable expression resolvable only by scanning (such as regex predicates), the preferred approach is to remove them in favor of using indexes. Where necessary, filter the results in post-processing

5.7 The mlcp Option -tolerate_errors Deprecated

The `-tolerate_errors` option of the mlcp `import` command is deprecated (and ignored) as of MarkLogic 9.0-2. The option will be removed in a future release. mlcp now always tolerates errors.

5.8 xdmp:transaction-mode XQuery Prolog Option Deprecated

The XQuery prolog option `xdmp:transaction-mode` is deprecated as of MarkLogic 9.0-2. Use the `xdmp:commit` and `xdmp:update` prolog options instead.

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

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

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

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

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

5.9 Deprecation of transaction-mode Option to xdmp:eval

The `transaction-mode` option of the `xdmp:eval` XQuery function and the `xdmp.eval` JavaScript function is deprecated as of MarkLogic 9.0-2. Use the `commit` and `update` options instead. For more details, see the function reference documentation for `xdmp:eval` (XQuery) and `xdmp.eval` (JavaScript).

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

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

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

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

5.10 XCC Session.setTransactionMode is Deprecated

Use of `Session.setTransactionMode` to specify commit semantics and transaction type is deprecated as MarkLogic 9.0-2. This function will be removed in a future version. Use the new `Session.setAutoCommit` and `Session.setUpdate` methods instead.

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

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

5.11 Java Client API 4.0.2 Deprecations

Java Client API 4.0.2 introduces the following deprecations.

The `com.marklogic.client.extra.httpclient.HttpClientConfigurator` interface is deprecated and will be removed in a future release. Use the new `com.marklogic.client.extra.okhttpclient.OkHttpClientConfigurator` interface instead. Attaching a configurator based on `HttpClientConfigurator` to a `DatabaseClientFactory` object no longer has any effect on the HTTP configuration.

The single parameter version of `DatabaseClientFactory.SecurityContext.withSSLContext` has been deprecated and will be removed in a future release. Instead, use the new version that requires an `X509TrustManager` as its second parameter. This change affects all classes that implement `DatabaseClientFactory.SecurityContext`, such as `DatabaseClientFactory.CertificateAuthContext`, `DatabaseClientFactory.KerberosAuthContext`, and `DatabaseClientFactory.DigestAuthContext`.

5.12 Java Client API 4.0.4 Deprecations

Java Client API 4.0.4 deprecates the following interfaces:

- NamespacesManager Interface Deprecated
- QueryBatcher.getQuerySuccessListeners Deprecated

5.12.1 NamespacesManager Interface Deprecated

The Java Client API interfaces for configuration namespace bindings are deprecated as of version 4.0.4 and will be removed in a future release.

Instead, you should use the REST Management API to define namespace bindings for your App Server. For details, see the `namespaces` property of the payload for `PUT:/manage/v2/servers/{id|name}/properties`.

This deprecation notice affects the following components of the Java Client API:

- The `com.marklogic.client.admin.NamespacesManager` interface.
- The `com.marklogic.client.admin.ServerConfigurationManager.newNamespacesManager` method.

5.12.2 QueryBatcher.getQuerySuccessListeners Deprecated

The method `QueryBatcher.getQuerySuccessListeners` is deprecated as of Java Client API 4.0.0 and will be removed in a future release. Use `QueryBatcher.getUrisReaderListeners` instead.

Both methods do the same thing in the same way. You do not need to change anything but the method name.

5.13 REST Client API Namespace Configuration Deprecation

The REST Client API methods for configuring namespace bindings are deprecated as of MarkLogic 9.0-5 and will be removed in a future release.

Instead, you should use the REST Management API to define namespace bindings for your App Server. For details, see the `namespaces` property of the payload for `PUT:/manage/v2/servers/{id|name}/properties`.

This deprecation notice affects the following methods of the REST Client API:

- GET, POST, PUT and DELETE on `/v1/config/namespaces`
- GET, PUT, and DELETE on `/v1/config/namespaces/{prefix}`

5.14 Configuration Packaging XQuery Library Deprecated

In MarkLogic 9.0-5, the Configuration Packaging XQuery API library has been deprecated. It will be removed from the product in a future release.

The following table lists the deprecated methods and the new alternative.

Deprecated Methods	Alternative
Configuration Packaging (`pkg:`) library: • `pkg:create` • `pkg:database-configuration` • `pkg:delete` • `pkg:differences` • `pkg:errors` • `pkg:exists` • `pkg:get-database` • `pkg:get-database-list` • `pkg:get-modules` • `pkg:get-package` • `pkg:get-package-list` • `pkg:get-server` • `pkg:get-server-list` • `pkg:install` • `pkg:installable` • `pkg:put-database` • `pkg:put-modules` • `pkg:put-server` • `pkg:remove-database` • `pkg:remove-modules` • `pkg:remove-server` • `pkg:revert` • `pkg:server-configuration` • `pkg:valid`	Use the following methods of the Configuration Management (`cma:`) library instead: • `cma:generate-config` • `cma:apply-config` For more details, see “Configuration Management API (CMA) XQuery and JavaScript Libraries” on page 46.

5.15 Configuration Manager Deprecated

The Configuration Manager tool is deprecated starting with MarkLogic 9.0-5 and removed from MarkLogic Server in version 10.0-3.

5.16 Hadoop Connector Deprecated

The Hadoop Connector is deprecated starting with MarkLogic release 10.0-3 and will be removed from the product in a future release.

5.17 HDFS Support Deprecated

HDFS support has been deprecated in MarkLogic 10.0-3.

5.18 CNTK Machine Learning Runtime Deprecated

The CNTK Machine Learning libraries and APIs are deprecated starting with MarkLogic release 10.0-4 and will be removed from the product in a future release. For more information, please see this Release Note from Microsoft.

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 10.0-5 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 10.0-5, but `MATCH` queries run filtered (`MATCH` does not use the Universal Index).
- `FULL OUTER JOIN` is not supported in 10.0-5

- SQL is read-only in 10.0-5. 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.
Semantic Graph 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.
Ops Director Guide	Provides procedures for setting up and using Ops Director for monitoring MarkLogic clusters.
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 Web Services (AWS) 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, 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).
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 110. 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 many of the other GUI tools (Query Console, Ops Director 1.x, and so on) are certified as follows:

- Internet Explorer 11 on Windows 10.Beginning with MarkLogic 10.0-3, Windows 7 support is discontinued because it will reach End-of-Life in January of 2020. Please see the MicroSoft support page for more details.
- Chrome 63 on Windows and Mac OS.

Ops Director 2.0 is certified as follows:

- Internet Explorer 11 on Windows 10
- Chrome 63 on Windows 10 and Mac OS

Other browser/platform combinations may work but are not as thoroughly tested. Please consult the documentation for your individual GUI tool for exact browser requirements.

6.7 Security: Prevent Abuse of System Entity Expansion

Normal XML processing allows for external entities to be referenced and included in the parsed content of XML files. If you want to disable this processing, set the trace event “Disable XML External Entities”.

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 technical questions, we encourage you to ask your question on Stack Overflow.

8.0 Copyright

MarkLogic Server 10.0 and supporting products.
Last updated: April 7, 2020

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 for your version of MarkLogic.