MarkLogic

# MarkLogic Kafka Connector

**Changes for version 1.2.2**

Biju George
7-8-2020

# Contents

# Summary of this document

This document describes the changes made to the connector version 1.2.2. Below are the changes / capabilities added

New Capabilities

1. Support of AVRO Documents with AVRO Converter
2. Support of JSON Documents with JSON Converter
3. Mutual Authentication with MarkLogic server

Documentation changes

1. Added the error handling details of Connector
2. Quick Recipes for customizing the connector.

# Examples of Sink Configurations

## JsonConverter without SSL

```
name = MarkLogicSink
connector.class = com.marklogic.kafka.connect.sink.MarkLogicSinkConnector
tasks.max = 1
key.converter = org.apache.kafka.connect.json.JsonConverter
value.converter = org.apache.kafka.connect.json.JsonConverter
topics = perf-queue
ml.connection.host = 192.168.0.29
ml.connection.port = 8071
ml.connection.database = CertServer
ml.connection.securityContextType = BASIC
ml.connection.username = srportal
ml.connection.password = abc
ml.connection.type = DIRECT
ml.connection.simpleSsl = false
ml.connection.ssl = false
ml.connection.certFile = \
ml.connection.certPassword = \
ml.connection.externalName = \
ml.connection.hostNameVerifier = \
ml.connection.tlsVersion = \
ml.connection.mutualAuth = false
ml.dmsdk.batchSize = 1
ml.dmsdk.threadCount = 1
ml.dmsdk.transform = \
ml.dmsdk.transformParams = \
ml.document.collections = coll-nossl
ml.document.format = JSON
ml.document.mimeType = \
ml.document.permissions = rest-reader,read,rest-writer,update
ml.document.uriPrefix = /kafka-data/nossl/
ml.document.uriSuffix = .json
key.converter.schemas.enable = false
value.converter.schemas.enable = false
errors.tolerance = all
errors.log.enable = false
errors.deadletterqueue.topic.name = test-dlq
errors.deadletterqueue.context.headers.enable = true
```

## JsonConverter with Simple SSL

```
name = MarkLogicSink
connector.class = com.marklogic.kafka.connect.sink.MarkLogicSinkConnector
tasks.max = 1
key.converter = org.apache.kafka.connect.json.JsonConverter
value.converter = org.apache.kafka.connect.json.JsonConverter
topics = perf-queue
ml.connection.host = 192.168.0.29
ml.connection.port = 8071
ml.connection.database = CertServer
ml.connection.securityContextType = BASIC
ml.connection.username = srportal
ml.connection.password = abc
ml.connection.type = DIRECT
ml.connection.simpleSsl = true
ml.connection.ssl = false
ml.connection.certFile = \
ml.connection.certPassword = \
ml.connection.externalName = \
ml.connection.hostNameVerifier = \
ml.connection.tlsVersion = \
ml.connection.mutualAuth = false
ml.dmsdk.batchSize = 1
ml.dmsdk.threadCount = 1
ml.dmsdk.transform = \
ml.dmsdk.transformParams = \
ml.document.collections = coll-simplessl
ml.document.format = JSON
ml.document.mimeType = \
ml.document.permissions = rest-reader,read,rest-writer,update
ml.document.uriPrefix = /kafka-data/simplessl/
ml.document.uriSuffix = .json
key.converter.schemas.enable = false
value.converter.schemas.enable = false
errors.tolerance = all
errors.log.enable = false
errors.deadletterqueue.topic.name = test-dlq
errors.deadletterqueue.context.headers.enable = true
```

## JsonConverter with 1way Mutual Authentication and STRICT HostName verification

```
name = MarkLogicSink
connector.class = com.marklogic.kafka.connect.sink.MarkLogicSinkConnector
tasks.max = 1
key.converter = org.apache.kafka.connect.json.JsonConverter
value.converter = org.apache.kafka.connect.json.JsonConverter
topics = perf-queue
ml.connection.host = myvm-host1
ml.connection.port = 8071
ml.connection.database = CertServer
ml.connection.securityContextType = BASIC
ml.connection.username = srportal
ml.connection.password = abc
ml.connection.type = DIRECT
ml.connection.simpleSsl = false
ml.connection.ssl = true
ml.connection.certFile = \
ml.connection.certPassword = \
ml.connection.externalName = \
ml.connection.hostNameVerifier = STRICT
ml.connection.tlsVersion = TLSv1.2
ml.connection.mutualAuth = false
ml.dmsdk.batchSize = 1
ml.dmsdk.threadCount = 1
ml.dmsdk.transform = \
ml.dmsdk.transformParams = \
ml.document.collections = coll-1wayauth-strict
ml.document.format = JSON
ml.document.mimeType = \
ml.document.permissions = rest-reader,read,rest-writer,update
ml.document.uriPrefix = /kafka-data/1wayauth/strict/
ml.document.uriSuffix = .json
key.converter.schemas.enable = false
value.converter.schemas.enable = false
errors.tolerance = all
errors.log.enable = false
errors.deadletterqueue.topic.name = test-dlq
errors.deadletterqueue.context.headers.enable = true
```

## JsonConverter with 2way Mutual Authentication and STRICT HostName verification

```
name = MarkLogicSink
connector.class = com.marklogic.kafka.connect.sink.MarkLogicSinkConnector
tasks.max = 1
key.converter = org.apache.kafka.connect.json.JsonConverter
value.converter = org.apache.kafka.connect.json.JsonConverter
topics = perf-queue
ml.connection.host = myvm-host1
ml.connection.port = 8071
ml.connection.database = CertServer
ml.connection.securityContextType = BASIC
ml.connection.username = srportal
ml.connection.password = abc
ml.connection.type = DIRECT
ml.connection.simpleSsl = false
ml.connection.ssl = true
ml.connection.certFile = /tmp/srportal.p12
ml.connection.certPassword = abc
ml.connection.externalName = \
ml.connection.hostNameVerifier = STRICT
ml.connection.tlsVersion = TLSv1.2
ml.connection.mutualAuth = true
ml.dmsdk.batchSize = 1
ml.dmsdk.threadCount = 1
ml.dmsdk.transform = \
ml.dmsdk.transformParams = \
ml.document.collections = coll-mutualauth-strict
ml.document.format = JSON
ml.document.mimeType = \
ml.document.permissions = rest-reader,read,rest-writer,update
ml.document.uriPrefix = /kafka-data/mutualauth/strict/
ml.document.uriSuffix = .json
key.converter.schemas.enable = false
value.converter.schemas.enable = false
errors.tolerance = all
errors.log.enable = false
errors.deadletterqueue.topic.name = test-dlq
errors.deadletterqueue.context.headers.enable = true
```

## JsonConverter with 2way Mutual Authentication and ANY HostName verification

```
name = MarkLogicSink
connector.class = com.marklogic.kafka.connect.sink.MarkLogicSinkConnector
tasks.max = 1
key.converter = org.apache.kafka.connect.json.JsonConverter
value.converter = org.apache.kafka.connect.json.JsonConverter
topics = perf-queue
ml.connection.host = 192.168.0.29
ml.connection.port = 8071
ml.connection.database = CertServer
ml.connection.securityContextType = BASIC
ml.connection.username = srportal
ml.connection.password = abc
ml.connection.type = DIRECT
ml.connection.simpleSsl = false
ml.connection.ssl = true
ml.connection.certFile = /tmp/srportal.p12
ml.connection.certPassword = abc
ml.connection.externalName = \
ml.connection.hostNameVerifier = ANY
ml.connection.tlsVersion = TLSv1.2
ml.connection.mutualAuth = true
ml.dmsdk.batchSize = 1
ml.dmsdk.threadCount = 1
ml.dmsdk.transform = \
ml.dmsdk.transformParams = \
ml.document.collections = coll-mutualauth-any
ml.document.format = JSON
ml.document.mimeType = \
ml.document.permissions = rest-reader,read,rest-writer,update
ml.document.uriPrefix = /kafka-data/mutualauth/any/
ml.document.uriSuffix = .json
key.converter.schemas.enable = false
value.converter.schemas.enable = false
errors.tolerance = all
errors.log.enable = false
errors.deadletterqueue.topic.name = test-dlq
errors.deadletterqueue.context.headers.enable = true
```

## JsonConverter with Certificate Authentication and ANY HostName verification

```
name = MarkLogicSink
connector.class = com.marklogic.kafka.connect.sink.MarkLogicSinkConnector
tasks.max = 1
key.converter = org.apache.kafka.connect.json.JsonConverter
value.converter = org.apache.kafka.connect.json.JsonConverter
topics = perf-queue
ml.connection.host = myvm-host1
ml.connection.port = 8071
ml.connection.database = CertServer
ml.connection.securityContextType = CERTIFICATE
ml.connection.type = DIRECT
ml.connection.simpleSsl = false
ml.connection.ssl = true
ml.connection.certFile = /tmp/srportal.p12
ml.connection.certPassword = abc
ml.connection.externalName = \
ml.connection.hostNameVerifier = ANY
ml.connection.tlsVersion = TLSv1.2
ml.connection.mutualAuth = false
ml.connection.username = \
ml.connection.password = \
ml.dmsdk.batchSize = 1
ml.dmsdk.threadCount = 1
ml.dmsdk.transform = \
ml.dmsdk.transformParams = \
ml.document.collections = coll-certauth
ml.document.format = JSON
ml.document.mimeType = \
ml.document.permissions = rest-reader,read,rest-writer,update
ml.document.uriPrefix = /kafka-data/certauth/
ml.document.uriSuffix = .json
key.converter.schemas.enable = false
value.converter.schemas.enable = false
errors.tolerance = all
errors.log.enable = false
errors.deadletterqueue.topic.name = test-dlq
errors.deadletterqueue.context.headers.enable = true
```

## JsonConverter with Certificate Authentication and STRICT HostName Verification

```
name = MarkLogicSink
connector.class = com.marklogic.kafka.connect.sink.MarkLogicSinkConnector
tasks.max = 1
key.converter = org.apache.kafka.connect.json.JsonConverter
value.converter = org.apache.kafka.connect.json.JsonConverter
topics = perf-queue
ml.connection.host = 192.168.0.29
ml.connection.port = 8071
ml.connection.database = CertServer
ml.connection.securityContextType = CERTIFICATE
ml.connection.type = DIRECT
ml.connection.simpleSsl = false
ml.connection.ssl = true
ml.connection.certFile = /tmp/srportal.p12
ml.connection.certPassword = abc
ml.connection.externalName = \
ml.connection.hostNameVerifier = STRICT
ml.connection.tlsVersion = TLSv1.2
ml.connection.mutualAuth = false
ml.connection.username = \
ml.connection.password = \
ml.dmsdk.batchSize = 1
ml.dmsdk.threadCount = 1
ml.dmsdk.transform = \
ml.dmsdk.transformParams = \
ml.document.collections = coll-certauth
ml.document.format = JSON
ml.document.mimeType = \
ml.document.permissions = rest-reader,read,rest-writer,update
ml.document.uriPrefix = /kafka-data/certauth/
ml.document.uriSuffix = .json
key.converter.schemas.enable = false
value.converter.schemas.enable = false
errors.tolerance = all
errors.log.enable = false
errors.deadletterqueue.topic.name = test-dlq
errors.deadletterqueue.context.headers.enable = true
```

## String Converter with simpleSSL

```
name = MarkLogicSink
connector.class = com.marklogic.kafka.connect.sink.MarkLogicSinkConnector
tasks.max = 1
key.converter = org.apache.kafka.connect.storage.StringConverter
value.converter = org.apache.kafka.connect.storage.StringConverter
topics = perf-queue
ml.connection.host = myvm-host1
ml.connection.port = 8071
ml.connection.database = CertServer
ml.connection.securityContextType = BASIC
ml.connection.username = srportal
ml.connection.password = abc
ml.connection.type = DIRECT
ml.connection.simpleSsl = true
ml.connection.ssl = false
ml.connection.certFile = \
ml.connection.certPassword = \
ml.connection.externalName = \
ml.connection.hostNameVerifier = \
ml.connection.tlsVersion = \
ml.connection.mutualAuth = false
ml.dmsdk.batchSize = 1
ml.dmsdk.threadCount = 1
ml.dmsdk.transform = \
ml.dmsdk.transformParams = \
ml.document.collections = coll-simplessl-string-conv
ml.document.format = JSON
ml.document.mimeType = \
ml.document.permissions = rest-reader,read,rest-writer,update
ml.document.uriPrefix = /kafka-data/simplessl/string/
ml.document.uriSuffix = .json
key.converter.schemas.enable = false
value.converter.schemas.enable = false
errors.tolerance = all
errors.log.enable = false
errors.deadletterqueue.topic.name = test-dlq
errors.deadletterqueue.context.headers.enable = true
```

## AvroConverter with simpleSSL

```
name = MarkLogicSink
connector.class = com.marklogic.kafka.connect.sink.MarkLogicSinkConnector
tasks.max = 1
key.converter = io.confluent.connect.avro.AvroConverter
value.converter = io.confluent.connect.avro.AvroConverter
topics = perf-queue
ml.connection.host = myvm-host1
ml.connection.port = 8071
ml.connection.database = CertServer
ml.connection.securityContextType = BASIC
ml.connection.username = srportal
ml.connection.password = abc
ml.connection.type = DIRECT
ml.connection.simpleSsl = true
ml.connection.ssl = false
ml.connection.certFile = \
ml.connection.certPassword = \
ml.connection.externalName = \
ml.connection.hostNameVerifier = \
ml.connection.tlsVersion = \
ml.connection.mutualAuth = false
ml.dmsdk.batchSize = 1
ml.dmsdk.threadCount = 1
ml.dmsdk.transform = \
ml.dmsdk.transformParams = \
ml.document.collections = coll-simplessl-avro-conv
ml.document.format = JSON
ml.document.mimeType = \
ml.document.permissions = rest-reader,read,rest-writer,update
ml.document.uriPrefix = /kafka-data/simplessl/avro/
ml.document.uriSuffix = .json
key.converter.schemas.enable = true
key.converter.schema.registry.url = http://localhost:8081
value.converter.schemas.enable = true
value.converter.schema.registry.url = http://localhost:8081
errors.tolerance = all
errors.log.enable = false
errors.deadletterqueue.topic.name = test-dlq
errors.deadletterqueue.context.headers.enable = true
```

## Error handling of the Connector

The below properties control the behavior of the connector when error happens

| Property | Description |
|---|---|
| errors.tolerance | *none* – This is the default value. With this value, the connector stops when connector gets an incorrect message.<br>*all* – If dead letter queue is NOT configured, then the connector stays alive and the bad messages are silently ignored. If dead letter queue is configured, then bad messages are moved to dead letter queue.<br><br>Notes:<br>1. The sink connector does not have a capability to process anything from the dead letter queue. The customer should build another process to inspect and correct the bad messages. |
| errors.deadletterqueue.topic.name | The name of the dead letter queue. This topic should be configured prior to starting the connector. The connector will not start if this topic is not available. If this topic is not configured and if errors.tolerance is 'all', the bad messages are silently ignored. |
| errors.deadletterqueue.context.headers.enable | This is an optional parameter.<br>If 'true', the failure reason for a message is recorded in message header.<br>If 'false', the failure reason for a message is NOT recorded in message header. |

Other error situations:

1. The connector attempts to connect to MarkLogic target on starting. The connector will not start if connection cannot be established. The messages sent to Kafka topic will be processed when the connector comes up successfully.
2. If MarkLogic becomes in-accessible when the connector is running, connector fails immediately and will require to re-start after the connection problem is resolved. The outstanding messages will be re-processed after the connector comes up. The connector manages the offsets internally so that re-processing will be only for the messages that were sent to the topic which were not processed when the connector was deleted.

## Recipes for customizing the connector

Below are some scenarios which will require the connector to be customized.

1. Document URI: The default behavior of the connector is to generate a uuid as last part of the document uri. The prefix and suffix can be controlled with Sink properties

   `ml.document.uriPrefix`
   `ml.document.uriSuffix`

   If the customer require to manipulate the URI, then `extractId` method of interface `com.marklogic.client.ext.document.ContentIdExtractor` should be implemented. An example implementation is given below. Here the last of the URI is generated from the value in the JSON document in the path `Account.id`

```
public class DefaultContentIdExtractor implements ContentIdExtractor {
	private ObjectMapper om = new ObjectMapper();
	@Override
	public String extractId(AbstractWriteHandle content) throws IOException {
		/* Sample Implementation of customising the URI*/
		JsonNode node = om.readTree(content.toString());
		String id = node.get("ddaAccount").get("id").asText();
		return id;
		//return UUID.randomUUID().toString();
	}
}
```

2. Additional document metadata: The default behavior of the connector adds only collections to the document. If additional metadata properties to be added, there can be two approaches
   i) Use a transform function and configure the transform in the sink properties
   ii) Update `com.marklogic.kafka.connect.sink.DefaultSinkRecordConverter` to add the additional meta data. A simple implementation of adding two meta data attributes is given below.

```
protected DocumentMetadataHandle addTopicToCollections (String topic, Boolean
addTopicToCollections) {
              DocumentMetadataHandle metadata = new DocumentMetadataHandle();
                     if (addTopicToCollections) {
                            metadata.getCollections().add(topic);
                     }
                     //Custom meta data addition.
                     metadata.getMetadataValues().add("publishedDt", new Date().toString());
                     metadata.getMetadataValues().add("publishedFrom", "Topic::" + topic);
              return metadata ;
       }
```

3. Security options: The sink connector supports BASIC/DIGEST/CERTIFICATE authentication with both 1waySSL and 2waySSL. The Host verification supported are ANY, STRICT and COMMON. If a custom verifier or any other security related changes are required, then `buildDatabaseClientConfig` method of the interface `com.marklogic.kafka.connect.DatabaseClientConfigBuilder` should be changed or newly implemented. The connector can be considered as any other Java client for implementing the security changes. Below are the security related sink properties and how to update those properties. The properties newly introduced in Sink version 1.2.2 is highlighted in **bold**

| Property | Description |
|---|---|
| `ml.connection.securityContextType` | The security context of the MarkLogic target App Server. Possible values are BASIC, DIGEST, CERTIFICATE |
| `ml.connection.username` | The username to connect to MarkLogic. |
| `ml.connection.password` | The password to connect to MarkLogic. This does not have effect with CERTIFICATE authentication |
| `ml.connection.type` | Configure as GATEWAY if the endpoint is a load balancer. Otherwise configure as DIRECT |
| `ml.connection.simpleSsl` | Configure as ‘true’ if a simple SSL connection is used. No Host Verification is done when configured like this. This might be enough for development, but not for production. |
| **`ml.connection.ssl`** | If stricter SSL is what is configured in the App server, configure as ‘true’. SimpleSsl should be ‘false’ if ml.connection.ssl is configured as ‘true. Refer the example sink configurations. |
| `ml.connection.certFile`<br>`ml.connection.certPassword`<br>`ml.connection.externalName` | Applicable for CERTIFICATE authentication only. The path to the user certificate in PKCS12 format, certificate password and external name to be provided. For details of certificate based authentication please see below documentation.<br>https://docs.marklogic.com/guide/security/security-based-auth |

| | |
|---|---|
| **ml.connection.hostNameVerifier** | Possible values are ANY, STRICT or COMMON. This is to configure the strictness of HostName verification. Refer below documentation for additional details. Default behavior is of 'ANY', if not configured.<br>http://docs.marklogic.com/javadoc/client/index.html?com/marklogic/client/DatabaseClientFactory.SSLHostnameVerifier.HostnameVerifierAdapter.html |
| **ml.connection.tlsVersion** | The TLS version. TLSv1.2 would be most appropriate value. However, these values can be configured if you have a lower TLS configuration.<br>TLSv1, TLSv1.1, TLSv1.2 |
| **ml.connection.mutualAuth** | If a 2waySSL is required for BASIC and DIGEST authentication, configure this as true. If true, the client certificate should be supplied. Hence, ml.connection.certFile & ml.connection.certPassword also need to be configured. |

## AVRO Converter support

The connector supports AVRO documents to be sent to the topic. The connector should be configured with AvroConverter and schema registry. The connector will ignore the schema and process only the value. An example of testing an avro document with a schema is below. Refer the examples provided for configuring the AVRO converter.

```
prompt>> ./kafka-avro-console-producer --broker-list localhost:9092 --topic perf-queue  --property
value.schema='{"type":"record","name":"myrecord","fields":[{"name":"f1","type":"string"},{"name":"
f2","type":"int"}]}'
   > {"f1":"string2","f2":1001}
```

The above example will insert will create a document with content `{"f1":"string2","f2":1001}`

## JSON Converter Support

The connector can be configured to use JSONConverter. Refer the examples provided for configuring the JSON converter. An example of testing below.

```
prompt>> ./kafka-console-producer --broker-list localhost:9092 --producer.config
../etc/kafka/producer.properties --topic perf-queue
   > {"f1":"string2","f2":1001}
```

==========