

# Cassandra

## Camel Cassandra Component

Available as of Camel 2.15

[Apache Cassandra](#) is an open source NoSQL database designed to handle large amounts on commodity hardware. Like Amazon's DynamoDB, Cassandra has a peer-to-peer and master-less architecture to avoid single point of failure and garanty high availability. Like Google's BigTable, Cassandra data is structured using column families which can be accessed through the Thrift RPC API or a SQL-like API called CQL.

This component aims at integrating Cassandra 2.0+ using the CQL3 API (not the Thrift API). It's based on [Cassandra Java Driver](#) provided by DataStax.

Maven users will need to add the following dependency to their `pom.xml`:

### pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-cassandraql</artifactId>
  <version>x.y.z</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

## URI format

The endpoint can initiate the Cassandra connection or use an existing one.

URI	Description
cql:localhost/keyspace	Single host, default port, usual for testing
cql:host1,host2/keyspace	Multi host, default port
cql:host1,host2:9042/keyspace	Multi host, custom port
cql:host1,host2	Default port and keyspace
cql:bean:sessionRef	Provided Session reference
cql:bean:clusterRef/keyspace	Provided Cluster reference

To fine tune the Cassandra connection (SSL options, pooling options, load balancing policy, retry policy, reconnection policy...), create your own Cluster instance and give it to the Camel endpoint.

## Endpoint Options

Option	Default	Description
clusterName		Cluster name
username and password		Session authentication
cql		CQL query. Can be overridden with a message header.
consistencyLevel		ANY, ONE, TWO, QUORUM, LOCAL_QUORUM...
prepareStatements	true	Use prepared statement (default) or not
resultSetConversionStrategy	ALL	How is ResultSet converted transformed into message body ALL, ONE, LIMIT_10, LIMIT_100...

## Messages

### Incoming Message

The Camel Cassandra endpoint expects a bunch of simple objects (`Object` or `Object[]` or `Collection<Object>`) which will be bound to the CQL statement as query parameters. If message body is null or empty, then CQL query will be executed without binding parameters.

Headers:

- `CamelCqlQuery` (optional, `String` or `RegularStatement`): CQL query either as a plain `String` or built using the `QueryBuilder`.

## Outgoing Message

The Camel Cassandra endpoint produces one or many a `Cassandra Row` objects depending on the `resultSetConversionStrategy`:

- `List<Row>` if `resultSetConversionStrategy` is `ALL` or `LIMIT_[0-9]+`
- `Single Row` if `resultSetConversionStrategy` is `ONE`
- Anything else, if `resultSetConversionStrategy` is a custom implementation of the `ResultSetConversionStrategy`

## Repositories

Cassandra can be used to store message keys or messages for the idempotent and aggregation EIP.

Cassandra might not be the best tool for queuing use cases yet, read [Cassandra anti-patterns queues and queue like datasets](#). It's advised to use `LeveledCompaction` and a small `GC grace` setting for these tables to allow tombstoned rows to be removed quickly.

### Idempotent repository

The `NamedCassandraIdempotentRepository` stores messages keys in a Cassandra table like this:

#### CAMEL\_IDEMPOTENT.cql

```
CREATE TABLE CAMEL_IDEMPOTENT (
  NAME varchar,    -- Repository name
  KEY varchar,     -- Message key
  PRIMARY KEY (NAME, KEY)
) WITH compaction = {'class':'LeveledCompactionStrategy'}
AND gc_grace_seconds = 86400;
```

This repository implementation uses lightweight transactions (also known as Compare and Set) and requires Cassandra 2.0.7+.

Alternatively, the `CassandraIdempotentRepository` does not have a `NAME` column and can be extended to use a different data model.

Option	Default	Description
<code>table</code>	<code>CAMEL_IDEMPOTENT</code>	Table name
<code>pkColumns</code>	<code>NAME, KEY</code>	Primary key columns
<code>name</code>		Repository name, value used for <code>NAME</code> column
<code>ttl</code>		Key time to live
<code>writeConsistencyLevel</code>		Consistency level used to insert/delete key: ANY, ONE, TWO, QUORUM, LOCAL_QUORUM...
<code>readConsistencyLevel</code>		Consistency level used to read/check key: ONE, TWO, QUORUM, LOCAL_QUORUM...

### Aggregation repository

The `NamedCassandraAggregationRepository` stores exchanges by correlation key in a Cassandra table like this:

### CAMEL\_AGGREGATION.cql

```
CREATE TABLE CAMEL_AGGREGATION (  
  NAME varchar,           -- Repository name  
  KEY varchar,            -- Correlation id  
  EXCHANGE_ID varchar,    -- Exchange id  
  EXCHANGE blob,          -- Serialized exchange  
  PRIMARY KEY (NAME, KEY)  
) WITH compaction = {'class':'LeveledCompactionStrategy'}  
   AND gc_grace_seconds = 86400;
```

Alternatively, the `CassandraAggregationRepository` does not have a `NAME` column and can be extended to use a different data model.

Option	Default	Description
table	CAMEL_AGGREGATION	Table name
pkColumns	NAME,KEY	Primary key columns
exchangeIdColumn	EXCHANGE_ID	Exchange Id column
exchangeColumn	EXCHANGE	Exchange content column
name		Repository name, value used for NAME column
ttl		Exchange time to live
writeConsistencyLevel		Consistency level used to insert/delete exchange: ANY, ONE, TWO, QUORUM, LOCAL_QUORUM...
readConsistencyLevel		Consistency level used to read/check exchange: ONE, TWO, QUORUM, LOCAL_QUORUM...