neo4j

Neo4j 4.0 Migration Guide

This guide describes how to migrate from Neo4j version 3.5 to Neo4j version 4.0.x.

This guide describes the following:

Who should read this?

This migration guide is written for:

- the engineer performing the Neo4j production migration.
- the operations engineer supporting and maintaining the Neo4j production database.
- the enterprise architect researching database migration.
- the infrastructure architect planning the Neo4j production migration.
- the enterprise data security manager responsible for the company's strategy for role-based access control.

1. Important information

This chapter provides important information that you must know before attempting a migration from Neo4j version 3.5 to Neo4j version 4.0.x.

This chapter describes the following:

- Supported upgrade paths
- Limitations

1.1. Supported upgrade paths

The following upgrade path is supported:

3.5.any 4.0.x

The following steps are required if you need to upgrade from a version earlier than 3.5:

1. Upgrade to version 3.5.latest by following the instructions in the Neo4j Operations Manual for 3.5.
2. Upgrade to version 4.0.x as per instructions in this guide.

1.2. Limitations

- Neo4j does not support downgrades.
- A Neo4j migration must be performed as an isolated operation. If you are planning to upgrade from a single-instance installation to a Causal Cluster, this must be performed separately from the migration to 4.0.2.
- In order to further minimize risk, it is recommended that while migrating, you do not switch from Community Edition to Enterprise Edition, change configuration, perform architectural restructuring, or similar tasks.

2. Prepare to upgrade

This chapter provides a checklist of things to prepare before performing a migration from Neo4j version 3.5 to Neo4j version 4.0.x.

Neo4j 4.0 requires Java 11.

Follow this checklist in order to ensure that you are well prepared before you start a production migration from Neo4j version 3.5 to Neo4j version 4.0.x:

Review Release Notes
To view the details of the changes that are included in each version, see the Release Notes.

Apply configuration changes
Prepare the contents of *neo4j.conf* to be used for the migrated database. If you are migrating a Causal Cluster, do this for each of the members in the cluster.

a. Review the section Changes to configuration settings and update all applicable configuration settings.

b. It is also useful to inspect the current configuration file and take note of any non-default settings. When upgrading, it is particularly important to note any custom values of the settings `dbms.directories.data` and `dbms.default_database`. In cluster installations, pay attention to cluster-specific configuration settings, which may be different on different cluster members.

> Some configuration settings that have changed names are are automatically migrated to the new setting names during startup. When this happens, it is logged in *neo4j.log*. The automatic migration is not permanent, so if it is not changed in *neo4j.conf*, will take place each time at startup. When the deprecated setting name are subsequently removed, unexpected problems may occur. It is therefore strongly recommended to update all relevant configuration settings at the time of the migration to 4.0.

Upgrade application code

Review the changes outlined in this guide and apply the necessary changes to your source code. How much development time is required to update the code will depend on the particular application. Make sure to test the application code thoroughly.

Upgrade custom plugins

Check the *plugins* directory to verify whether custom plugins are used in your deployment. Ensure that any plugins are compatible with Neo4j 4.0.2.

Plan disk space requirements

An upgrade requires substantial free disk space, as it makes an entire copy of the database. For the upgrade, make sure to make available an additional (50% * size_of(*database directory*). In a default configuration, the *database directory* is *databases/neo4j*, which is located in the *data* directory. In addition to this, do not forget to reserve the disk space needed for the pre-upgrade backup.

The upgraded database may require slightly larger data files overall.

Perform a test upgrade

Based on the findings in this chapter, allocate a production-like test environment for the upgrade and do a test upgrade. The test upgrade will give you valuable information about the time required for the production upgrade, as well as potential additional action points, such as upgrade of plugins and application code.

Review the logs

The *neo4j.log* file contains valuable information on how many steps the upgrade will involve and how far it has progressed. For large upgrades, it is a good idea to monitor this log continuously. Below is a sample of what the log may look like:

```
2018-09-18 13:24:23.243+0000 INFO  Starting...
2018-09-18 13:24:24.262+0000 INFO  Initiating metrics...
2018-09-18 13:24:24.488+0000 INFO  Starting upgrade of database
2018-09-18 13:24:24.538+0000 INFO  Migrating Indexes (1/5):
2018-09-18 13:24:24.542+0000 INFO    10% completed
2018-09-18 13:24:24.543+0000 INFO    20% completed
2018-09-18 13:24:24.543+0000 INFO    30% completed
...
...
...
2018-09-18 13:24:24.574+0000 INFO  Migrating Counts store (5/5):
2018-09-18 13:24:24.574+0000 INFO    10% completed
2018-09-18 13:24:24.574+0000 INFO    20% completed
2018-09-18 13:24:24.575+0000 INFO    30% completed
...
...
...
2018-09-18 13:24:24.576+0000 INFO    100% completed
2018-09-18 13:24:24.584+0000 INFO  Successfully finished upgrade of database
```

3. Surface changes

This chapter describes breaking changes to the Neo4j surface when migrating from Neo4j version 3.5 to Neo4j version 4.0.

This chapter describes the following:

- Security
- Changes to configuration settings
- Cypher syntax
- Procedures
- Authentication and authorization
- Logs
- Metrics
- Cluster discovery
- REST endpoints
- JMX
- Index migration
- Tools
- Backups
- Embedded layout
- Core Java API

3.1. Security

In 3.x it was possible to blacklist properties using the configuration settings `dbms.security.property_level.enabled` and `dbms.security.property_level.blacklist`. These configuration settings have been discontinued and the blacklisting functionality must be replaced by the Cypher `DENY` command. Note that the `DENY` command must be applied while Neo4j is running. For details, see Cypher Manual Security Graph and sub-graph access control.

Neo4j will fail to start if any of the old blacklist configuration settings are present in *neo4j.conf*.

3.2. Changes to configuration settings

Previous name	Change	New name (if applicable)
`dbms.active_database`	Renamed	`dbms.default_database`
`dbms.connectors.default_listen_address`	Renamed	`dbms.default_listen_address`
`dbms.connectors.default_advertised_address`	Renamed	`dbms.default_advertised_address`
`dbms.ssl.policy.*.allow_key_generation`	This setting is removed. Neo4j will no longer automatically generate a self-signed certificate.	
`dbms.backup.address`	Renamed	`dbms.backup.listen_address`

Previous name	Change	New name (if applicable)
`dbms.logs.query.enabled`	This is no longer a boolean setting. Valid values are: `OFF`, `INFO` or `VERBOSE`.	
`causal_clustering.cluster_routing_ttl`	Renamed	`dbms.routing_ttl`
`causal_clustering.middleware_logging.level`	This setting has been renamed, and valid values are: `DEBUG`,`INFO`, `WARN`, `ERROR` or `NONE`	`causal_clustering.middleware.logging.level`
`causal_clustering.disable_middleware_logging`	This setting is removed. Set `causal_clustering.middleware.logging.level=OFF` to disable middleware logging.	
`metrics.neo4j.logrotation.enabled`	Renamed	`metrics.neo4j.logs.enabled`
`metrics.enabled`	This setting no longer changes the default values of the individual metrics. Instead it turns off the whole metrics module.	
`dbms.security.auth_provider`	This setting is replaced by two new settings.	`dbms.security.authentication_providers` and `dbms.security.authorization_providers`

The configuration for SSL encryption is reworked. See Operations Manual SSL framework

3.3. Cypher syntax

- All changes in the Cypher language syntax are detailed in Cypher Manual Removals, deprecations, additions and extensions. Please review it thoroughly and make necessary changes in your code.
- We would like to draw some extra attention to the fact that the parameter syntax `{parameter}` is completely removed and has been replaced by the syntax `$parameter`.

3.4. Procedures

The following procedures have been refactored:

Old procedure	New procedure	Comment
`db.awaitIndex (indexId :: INTEGER?, timeOutSeconds = 300 :: INTEGER?) :: VOID`	`db.awaitIndex (indexName :: STRING?, timeOutSeconds = 300 :: INTEGER?) :: VOID`	Indexes are now uniquely identified by name, instead of ID.
`dbms.cluster.overview() :: (id :: STRING?, addresses :: LIST? OF STRING?, role :: STRING?, groups :: LIST? OF STRING?, database :: STRING?)`	`dbms.cluster.overview() :: (id :: STRING?, addresses :: LIST? OF STRING?, databases :: MAP?, groups :: LIST? OF STRING?)`	Shows roles for all databases.
`dbms.cluster.role() :: (role :: STRING?)`	`dbms.cluster.role (database :: STRING?) :: (role :: STRING?)`	Takes `database` name as parameter.
`dbms.cluster.routing.getRoutingTable (context :: MAP?) :: (ttl :: INTEGER?, servers :: LIST? OF MAP?)`	`dbms.cluster.routing.getRoutingTable (context :: MAP?, database = null :: STRING?) :: (ttl :: INTEGER?, servers :: LIST? OF MAP?)`	Takes `database` name as parameter.

Old procedure	New procedure	Comment
`db.createIndex (index :: STRING?, providerName :: STRING?) :: (index :: STRING?, providerName :: STRING?, status :: STRING?)`	`db.createIndex (indexName :: STRING?, labels :: LIST? OF STRING?, properties :: LIST? OF STRING?, providerName :: STRING?, config = {} :: MAP?) :: (name :: STRING?, labels :: LIST? OF STRING?, properties :: LIST? OF STRING?, providerName :: STRING?, status :: STRING?)`	Used to take the index pattern `":Label(prop)"` as an argument, and now takes `labels` and `properties` as separate lists. Those are also yielded as result. Now needs to be given an `indexName`. Can now take index settings as a map. This is optional.
`db.createUniquePropertyConstraint (index :: STRING?, providerName :: STRING?) :: (index :: STRING?, providerName :: STRING?, status :: STRING?)`	`db.createUniquePropertyConstraint (constraintName :: STRING?, labels :: LIST? OF STRING?, properties :: LIST? OF STRING?, providerName :: STRING?, config = {} :: MAP?) :: (name :: STRING?, labels :: LIST? OF STRING?, properties :: LIST? OF STRING?, providerName :: STRING?, status :: STRING?)`	Used to take the index pattern `":Label(prop)"` as an argument, and now takes `labels` and `properties` as separate lists. Those are also yielded as result. Now needs to be given a `constraintName`. Can now take index settings as a map. This is optional.
`db.createNodeKey (index :: STRING?, providerName :: STRING?) :: (index :: STRING?, providerName :: STRING?, status :: STRING?)`	`db.createNodeKey (constraintName :: STRING?, labels :: LIST? OF STRING?, properties :: LIST? OF STRING?, providerName :: STRING?, config = {} :: MAP?) :: (name :: STRING?, labels :: LIST? OF STRING?, properties :: LIST? OF STRING?, providerName :: STRING?, status :: STRING?)`	Used to take the index pattern `":Label(prop)"` as an argument, and now takes `labels` and `properties` as separate lists. Those are also yielded as result. Now need to be given a `constraintName`. Can now take index settings as a map. This is optional.

Old procedure	New procedure	Comment
`db.indexes() :: (description :: STRING?, indexName :: STRING?, tokenNames :: LIST? OF STRING?, properties :: LIST? OF STRING?, state :: STRING?, type :: STRING?, progress :: FLOAT?, provider :: MAP?, id :: INTEGER?, failureMessage :: STRING?)`	`db.indexes() :: (id :: INTEGER?, name :: STRING?, state :: STRING?, populationPercent :: FLOAT?, uniqueness :: STRING?, type :: STRING?, entityType :: STRING?, labelsOrTypes :: LIST? OF STRING?, properties :: LIST? OF STRING?, provider :: STRING?)`	Rename `indexName` to `name`. Rename `tokenNames` to `labelsOrTypes`. Rename `progress` to `populationPercent`. Field `type` used to describe entity type (node or relationship), uniqueness, and index type. This splits up into `type`, `uniqueness`, and `entityType`. Field `provider` is now a string instead of a map. Removed `description` in favor of `db.schemaStatements`. Moved `failureMessage` to procedure `db.indexDetails`.
`db.resampleIndex (index :: STRING?) :: VOID`	`db.resampleIndex (indexName :: STRING?) :: VOID`	Indexes are now uniquely identified by name, instead of index pattern `":Label(prop)"`.

The following are new procedures:

New procedure	Comment
`db.indexDetails (indexName :: STRING?) :: (id :: INTEGER?, name :: STRING?, state :: STRING?, populationPercent :: FLOAT?, uniqueness :: STRING?, type :: STRING?, entityType :: STRING?, labelsOrTypes :: LIST? OF STRING?, properties :: LIST? OF STRING?, provider :: STRING?, indexConfig :: MAP?, failureMessage :: STRING?)`	For the specified index all information included by `db.indexes` together with `indexConfig` and `failureMessage`.
`db.schemaStatements () :: (name :: STRING?, type :: STRING?, createStatement :: STRING?, dropStatement :: STRING?)`	Get all create and drop statements needed to exactly replicate the schema rules (indexes and constraints) for this database.

3.5. Authentication and authorization

3.5.1. Deprecated and removed security procedures

In 3.x, authentication and authorization was managed via the built-in `dbms.security` procedures. In 4.x, these procedures still exist but are deprecated. If you still want to use them, they must now be run in a session towards the `system` database, and cannot be followed by `YIELD`. There are two options for rewriting your code and routines for managing authentication and authorization. The first of these is recommended:

1. Rewrite the procedures to the corresponding Cypher administration commands, using the the

conversion guide below.

2. Run the procedures in a session towards the `system` database and replace any `YIELD` parts by post-processing on the application side.

The procedure `dbms.security.changePassword(password, requirePasswordChange)` has been entirely removed since the corresponding Cypher administration command also requires the old password, and thus is more secure.

The following table is a conversion guide between the security procedures and the Cypher administration commands. For more info about the administration commands, see Cypher Manual → User and role management.

Procedure	Administration command
`dbms.security.createUser`	`CREATE USER`
`dbms.security.deleteUser`	`DROP USER`
`dbms.security.changePassword`	`ALTER CURRENT USER SET PASSWORD`
`dbms.security.listUsers`	`SHOW USERS`
`dbms.security.changeUserPassword`	`ALTER USER`
`dbms.security.suspendUser`	`ALTER USER`
`dbms.security.activateUser`	`ALTER USER`
`dbms.security.addRoleToUser`	`GRANT ROLE TO USER`
`dbms.security.removeRoleFromUser`	`REVOKE ROLE FROM USER`
`dbms.security.listRoles`	`SHOW ROLES`
`dbms.security.listRolesForUser`	`SHOW USERS`
`dbms.security.listUsersForRole`	`SHOW ROLES WITH USERS`
`dbms.security.createRole`	`CREATE ROLE`
`dbms.security.deleteRole`	`DROP ROLE`

3.5.2. Removal of flat files for authentication and authorization

In 3.x, authentication and authorization were managed in flat files. Users in the *auth* file and roles and role assignments in the *roles* file will be automatically migrated to the `system` database when upgrading from Neo4j 3.5 to Neo4j 4.0.

The Neo4j admin commands `set-initial-password` and `set-default-admin` continue to work in 4.0 and write to the same files as in 3.x. Any content in these files will be considered on the first start of Neo4j after upgrading from 3.5. You can run these commands before upgrading the Neo4j installation, or after, as long as they are run before completing the upgrade of the database files which is done at first start of the new installation.

The command `set-initial-password` will only be applied if the default user `neo4j` with the default password is the only user present, while `set-default-admin` will only be applied when no roles are present.

The use of *auth* and *role* files in Neo4j 3.x meant that multiple databases could have different user and role configurations. In addition, a single database configured in a cluster could have different *auth* and *role* settings on each instance of the cluster. Neo4j 4.0 allows multiple databases to run within a single instance, or in a cluster. If you are bringing multiple databases together from multiple Neo4j 3.5 installations, or if you are upgrading a cluster with multiple instances, you need to manually merge the *auth* and *role* files before the upgrade.

It is still possible to have different security configurations per database after the upgrade, but this needs to be managed through the granting of privileges and roles specific to databases after the upgrade. The built-in roles from 3.5 still exist, but will apply to all databases after the upgrade, unless explicitly modified using the new security administration commands. The ability to manage database specific roles and privileges is described in more detail in Cypher Manual Administration.

It is no longer possible to have different security privileges on different instances of a cluster. The entire cluster shares the privileges configured in the `system` database using Cypher administration commands. In practice this means that users have the same privileges regardless of which server in a cluster they access.

3.6. Logs

Relevant logs produced by Neo4j will now have a prefix which indicates the database to which the log line pertains. Such log lines will have the database name printed prior to the regular text. For example, `[neo4j]` or `[system]`.

Example 1. Some log lines for the system database

```
2019-12-02 22:27:41.820+0000 INFO [o.n.k.d.Database] [system] No check point found in transaction log
2019-12-02 22:27:41.820+0000 INFO [o.n.k.d.Database] [system] Recovery required from position
LogPosition{logVersion=0, byteOffset=64}
2019-12-02 22:27:41.820+0000 INFO [o.n.k.r.Recovery] [system]   10% completed
2019-12-02 22:27:41.820+0000 INFO [o.n.k.r.Recovery] [system]   20% completed
2019-12-02 22:27:41.820+0000 INFO [o.n.k.r.Recovery] [system]   30% completed
...
```

Other log lines might relate to the DBMS as a whole, or be logged by a component that lives on a higher level but still operates on a particular database. For example:

Example 2. Some log lines from the Core database manager starting the Neo4j database.

```
2019-12-02 22:27:41.964+0000 INFO [c.n.c.c.CoreDatabaseManager] Creating 'neo4j' database.
2019-12-02 22:27:41.967+0000 INFO [c.n.c.c.CoreDatabaseManager] Starting 'neo4j' database.
...
```

3.7. Metrics

In 4.0.2, there are two types of metrics: global metrics and database-local metrics. The metric naming is different in 4.0.2 compared to 3.x. For details about available metrics and the new naming patterns, please refer to Operations Manual Metrics.

3.8. Cluster discovery

Cluster discovery is now implemented on top of Akka, instead of Hazelcast, and a few minor changes have been made as part of this transition.

- The advertised host must exactly match that of `initial_discovery_members`.
- Connections are now opened from Cores to Read Replicas, in addition to vice versa, so therefore the advertised discovery port must be **open** on Read Replicas.

3.9. Cluster REST endpoints

The REST endpoints have moved and now exist per database:

Old endpoint	New endpoint
`/db/manage/server/causalclustering/writable`	`/db/<databasename>/cluster/writable`
`/db/manage/server/causalclustering/read-only`	`/db/<databasename>/cluster/read-only`
`/db/manage/server/causalclustering/available`	`/db/<databasename>/cluster/available`
`/db/manage/server/causalclustering/status`	`/db/<databasename>/cluster/status`

3.10. JMX

In 3.x, Neo4j exposed several JMX beans in order to provide some monitoring information in addition to the metrics exposed by Neo4j. In some instances, the provided data was incomplete or incorrect, and in some cases different beans even provided conflicting information. All of the previous JMX endpoints have been removed and are replaced by a new set of beans that expose exactly the same information as the corresponding Neo4j metrics.

JMX beans are available only in Enterprise Edition.

3.11. Index migration

- Indexes are automatically upgraded to the most recent index provider during migration.

  Depending on what index providers were used previously, the migration of indexes may change the distribution of memory utilization. In a database with many indexes, a significant amount of memory may have been reserved for Lucene. After the migration, it could be necessary to allocate some of that memory to the page cache instead. For a detailed description on how memory is allocated and used, refer to Operations Manual Memory configuration. Use `neo4j-admin memrec --database` to inspect the database before and after migration.

- Support for explicit indexes has been removed and the functionality has been replaced by full-text indexes. For details, see Cypher Manual Indexes to support full-text search.

3.12. Tools

Database specific commands provided by `neo4j-admin` now support `--database`, which can be used to specify a database for a specified operation.

In cases when the `--database` option is not specified, `neo4j` will be used as the default database.

3.13. Backups

Backups must now be taken of all databases.

A default installation has two databases, named `system` and `neo4j` respectively. Use the `--database` option of the `neo4j-admin backup` command to specify the database to backup. For more information, see Operations Manual Perform a backup.

3.14. Embedded layout

To support multiple databases in embedded, the store files, transaction files and log files no longer reside in the base directory. Instead, files are separated per database in separate directories.

3.15. Core Java API

3.15.1. JDK 11

Neo4j 4.0 is the first major release that requires JDK 11. Custom extensions and procedures can also be compiled now for JDK 11 (for example `-target 11`. It is generally recommended to use the latest available JDK 11 in order to access available fixes and leverage performance improvements.

3.15.2. Classes removed or excluded from the public API

Please refer to Classes removed from public API for a complete list of classes removed or excluded from the public API.

3.15.3. Renamed classes

The following classes have been renamed:

Old class name	New class name
`org.neo4j.graphdb.factory.GraphDatabaseSettings.BoltConnector`	`org.neo4j.configuration.connectors.BoltConnector`
`org.neo4j.graphdb.factory.GraphDatabaseSettings.BoltConnector.EncryptionLevel`	`org.neo4j.configuration.connectors.BoltConnector.EncryptionLevel`
`org.neo4j.kernel.configuration.HttpConnector`	`org.neo4j.configuration.connectors.HttpConnector`
`org.neo4j.graphdb.factory.GraphDatabaseSettings`	`org.neo4j.configuration.GraphDatabaseSettings`
`org.neo4j.graphdb.factory.GraphDatabaseSettings.SchemaIndex`	`org.neo4j.configuration.GraphDatabaseSettings.SchemaIndex`
`org.neo4j.backup.OnlineBackup`	`com.neo4j.backup.OnlineBackup`
`org.neo4j.helpers.SocketAddress`	`org.neo4j.configuration.helpers.SocketAddress`
`org.neo4j.graphdb.event.TransactionEventHandler`	`org.neo4j.graphdb.event.TransactionEventListener`
`org.neo4j.graphdb.factory.GraphDatabaseSettings.TransactionStateMemoryAllocation`	`org.neo4j.configuration.GraphDatabaseSettings.TransactionStateMemoryAllocation`
`org.neo4j.graphdb.index.fulltext.AnalyzerProvider`	`org.neo4j.graphdb.schema.AnalyzerProvider`

3.15.4. Changes to the API

org.neo4j.graphdb.schema

Neo4j 4.0 comes with significant changes in schema and indexes. Most of the related classes have additional possibilities. Changes include:

- Starting with 4.0, all of the indexes are named. The name of an index can be retrieved using `getName()` call on `IndexDefinition` and `ConstraintDefinition`.
- The definition of an index can be looked up by name using `Schema`.
- Single label and relationship type accessors `getLabel()` and `getRelationshipType()` have been removed from `IndexDefinition`.

  Affected classes:

- `org.neo4j.graphdb.schema.ConstraintCreator`
- `org.neo4j.graphdb.schema.ConstraintDefinition`
- `org.neo4j.graphdb.schema.IndexCreator`

* `org.neo4j.graphdb.schema.IndexDefinition`
* `org.neo4j.graphdb.schema.Schema`

org.neo4j.graphdb.event

Transaction event listeners have an updated behavior. Changes include:

* As part of the callback, you will always receive the owning `GraphDatabaseService` as one of the parameters.
* The `beforeCommit` listener method has access to an ongoing transaction over the `transaction` call parameter.
* `DatabaseEventListener` is a new type of listener that has been introduced. Since Neo4j now supports multiple databases you might want to be able to listen to database events from several databases. It can be registered and de-registered in `DatabaseManagementService`.

  Affected classes:

* `org.neo4j.graphdb.event.TransactionEventListener`
* `org.neo4j.graphdb.event.DatabaseEventContext`
* `org.neo4j.graphdb.event.DatabaseEventListener`

org.neo4j.helpers

Most of the helpers are no longer part of the public API. The `SocketAddress` helper has minor API changes.

Affected classes:

* `org.neo4j.configuration.helpers.SocketAddress`

com.neo4j.backup

The backup facade has been simplified and adapted to a multi-database environment.

Affected classes:

* `com.neo4j.backup.OnlineBackup`

org.neo4j.configuration

Configuration API has been updated to be typed. It is no longer safe to assume that the configuration is a set of random key-value pairs. All pairs unknown to Neo4j will be rejected. Additionally, some settings have been renamed as well. Please check settings names migration in the corresponding migration manual section.

Affected classes:

* `org.neo4j.configuration.GraphDatabaseSettings`
* `org.neo4j.graphdb.config.Setting`
* `org.neo4j.configuration.connectors.BoltConnector`

org.neo4j.graphdb.Transaction

Transaction API changes are one of the biggest API updates that are part of 4.0. All of the methods that should be executed in transaction have been moved from `GraphDatabaseService` to `Transaction`. This means that if you need to create entities, or access them, you should now be able to find all of the methods in `Transaction`. Additionally, starting with 4.0, transactions are no longer thread-bound. This means that any call to `GraphDatabaseService::beginTx()` will create a new independent transaction, even if it called from one thread.

Affected classes:

- `org.neo4j.graphdb.Transaction`

org.neo4j.graphdb.Entity

As part of 4.0, the `PropertyContainer` interface is removed, and all property-related methods moved to `Entity`. Access to entities should always be transactional. This also means that an entity can only be safely accessed from a transaction where it was created or retrieved.

Affected classes:

- `org.neo4j.graphdb.Entity`
- `org.neo4j.graphdb.Node`
- `org.neo4j.graphdb.Relationship`

org.neo4j.graphdb.GraphDatabaseService

As part of 4.0, all methods that require transactions are moved to `Transaction`. In addition, a set of `executeTransactionally` methods have been added to provide a convenient way of query executions in a separate transaction.

Affected classes:

- `org.neo4j.graphdb.GraphDatabaseService`

org.neo4j.harness and com.neo4j.harness

Support has been added for official testing support classes. Starting with 4.0, Neo4j provides a set of Junit 4 rules and Junit 5 extensions for community and enterprise users.

Affected classes:

- `com.neo4j.harness.junit.extension.EnterpriseNeo4jExtension`
- `com.neo4j.harness.junit.rule.EnterpriseNeo4jRule`
- `org.neo4j.harness.junit.extension.Neo4j`
- `org.neo4j.harness.junit.extension.Neo4jExtension`
- `org.neo4j.harness.junit.extension.Neo4jExtensionBuilder`
- `org.neo4j.harness.junit.rule.Neo4jRule`

org.neo4j.dbms.api

The top-level Neo4j API has been updated. The main access point that should be used to access individual databases, or perform any database management operations, is called `DatabaseManagementService`. It can be constructed by the Community or Enterprise version of `DatabaseManagementServiceBuilder`.

Example 3. Using `DatabaseManagementService`

In this example, we are constructing a new `managementService` and a lookup `GraphDatabaseService` for the database named `neo4j`:

```
var managementService = new DatabaseManagementServiceBuilder( homeDirectory ).build();
var databaseService = managementService.database( "neo4j" );
```

Affected classes:

* `org.neo4j.dbms.api.DatabaseManagementService`
* `org.neo4j.dbms.api.DatabaseManagementServiceBuilder`

4. Upgrade a single instance

This chapter describes the necessary steps to migrate a single instance from Neo4j version 3.5 to Neo4j version 4.0.x.

Pre-upgrade steps

* Refer to Supported upgrade paths regarding supported upgrade paths.
* Read Prepare to upgrade thoroughly and perform all the steps listed there.

Shutdown and backup

1. If the database is running, shut it down cleanly.
2. Perform and verify backups:
   - Back up *neo4j.conf*.
   - Back up all the files used for encryption, i.e. private key, public certificate, and the contents of the *trusted* and *revoked* directories. The locations of these are described in Operations Manual → SSL framework.
   - Verify that you have a full backup that is stored in a safe location, either using the online backup tool or offline backups.

Upgrade

1. Install Neo4j 4.0.2 using one of the following methods, specific to your technology:
   a. If using a tarball or zipfile for installation:
      i. Untar or unzip Neo4j 4.0.2.
      ii. Transfer the new *neo4j.conf* that you prepared in the *Apply configuration changes* step in Prepare to upgrade.
      iii. Set `dbms.allow_upgrade=true` in *neo4j.conf* of the 4.0.2 installation. Neo4j will fail to start without this configuration.
      iv. Copy the files used for encryption from the old installation to the new one.
      v. Copy the *data* directory from the old installation to the new one. This step is not applicable if you have `dbms.directories.data` pointing to a directory outside of *NEO4J_HOME*.
   b. If using a Debian or RPM distribution:
      i. Set `dbms.allow_upgrade=true` in *neo4j.conf*.
      ii. Install Neo4j 4.0.2.
      iii. When prompted, review the differences between the *neo4j.conf* files of the previous version and Neo4j 4.0.2. Transfer any custom settings to the 4.0.2 installation, as noted under the *Apply configuration changes* step in Prepare to upgrade. Make sure to preserve `dbms.allow_upgrade=true` as set in the instruction above. Neo4j will fail to start without this configuration.
2. Start up Neo4j 4.0.2. The database upgrade will take place during startup.

   The *neo4j.log* file contains valuable information on how many steps the upgrade will involve and how far it has progressed. For large upgrades, it is a good idea to monitor this log continuously.

Post-upgrade steps

1. When the upgrade has finished, `dbms.allow_upgrade=true` should be set to `false` or be removed.
2. Restart the database.
3. It is good practice to make a full backup immediately after the upgrade.

5. Upgrade a Causal Cluster

This chapter describes the necessary steps to migrate a Causal Cluster from Neo4j version 3.5 to Neo4j version 4.0.x.

Pre-upgrade steps

Refer to Supported upgrade paths regarding supported upgrade paths.

1. Read Prepare to upgrade thoroughly and perform all the steps listed there.
2. Perform and verify backups:
    - Back up *neo4j.conf*.
    - Back up all the files used for encryption, i.e. private key, public certificate, and the contents of the *trusted* and *revoked* directories. The locations of these are described in Operations Manual SSL framework. You should back up these files on each server in the cluster.
    - Verify that you have a full backup that is stored in a safe location.
3. Prepare a new *neo4j.conf* file for each of the servers in the cluster, following the instructions under the *Apply configuration changes* step in Prepare to upgrade.

Downtime

The upgrade of a Causal Cluster to version 4.0.2 will require downtime. A test upgrade on a production-like equipment provides information on the duration of the downtime.

Upgrade steps

1. Shut down all the servers in the cluster
2. On one of the Core Servers:
    a. Install Neo4j using one of the following methods, specific to your technology:
        - If using a tarball or zipfile for installation:
            i. Untar or unzip the version of Neo4j that you want to upgrade to.
            ii. Transfer the new *neo4j.conf* that you prepared in the *Apply configuration changes* step in Prepare to upgrade.
            iii. Set `dbms.mode=SINGLE` in *neo4j.conf*.
            iv. Set `dbms.allow_upgrade=true` in *neo4j.conf*. Neo4j will fail to start without this configuration.
            v. Copy the files used for encryption from the old installation to the new one.
            vi. Copy the *data* directory from the old installation to the new one. This step is not applicable if you have `dbms.directories.data` pointing to a directory outside of *NEO4J_HOME*.
        - If using a Debian or RPM distribution:
            i. Set `dbms.mode=SINGLE` in *neo4j.conf*.
            ii. Set `dbms.allow_upgrade=true` in *neo4j.conf*.

        iii. Install the version of Neo4j that you want to upgrade to.
        iv. When prompted, review the differences between the *neo4j.conf* files of the previous version and Neo4j 4.0.2. Transfer any custom settings to the 4.0.2 installation, as noted under the *Apply configuration changes* step in Prepare to upgrade. Make sure to preserve `dbms.mode=SINGLE` and `dbms.allow_upgrade=true` as set in the instruction above.

    b. Start up Neo4j. The database upgrade will take place during startup.

       The *neo4j.log* file contains valuable information on how many steps the upgrade will involve and how far it has progressed. For large upgrades, it is a good idea to monitor this log continuously.

    c. Stop your Neo4j database once again.
    d. Set `dbms.allow_upgrade=false`, or remove it.
    e. Set `dbms.mode=CORE` in *neo4j.conf* to re-enable Causal Clustering in the configuration.
    f. Use `neo4j-admin dump` to make a copy of the database.
    g. Do not yet restart the database.

3. On each of the other Core Servers:
    a. Delete the database directory (in a default configuration, this is the directory *databases/neo4j* which is located in the *data* directory).
    b. Install the version of Neo4j that you want to upgrade to.
    c. Transfer any custom settings to the new installation, as prepared in the pre-upgrade step.
    d. If using a tarball or zipfile for installation: Copy the files used for encryption from the old installation to the new one.
    e. Perform `neo4j-admin unbind` on the instance.
    f. Using `neo4j-admin load`, restore the upgraded database onto this server.
4. Startup all the Core Servers and see the cluster form.
5. On each of the Read Replica servers:
    a. Stop Neo4j.
    b. Delete the database directory (in a default configuration, this is the directory *databases/neo4j* which is located in the *data* directory).
    c. Install Neo4j 4.0.2.
    d. Transfer settings to the new installation, as prepared in the pre-upgrade step.
    e. If using a tarball or zipfile for installation: Copy the files used for encryption from the old installation to the new one.
    f. Using `neo4j-admin dump/load`, restore the upgraded database onto this server. Alternatively, you can omit this step and let the Read Replica do a complete store copy.
    g. Start the Read Replica and see it join the cluster.

6. Upgrade Neo4j drivers

This chapter describes the necessary information to migrate Neo4j drivers from 1.7 to 4.0.

The 4.0 drivers have been designed to work with Neo4j 4.0. In 4.0, the drivers are built to provide a user-friendly and unified API across all languages, to take advantage of all new features and services introduced in Neo4j 4.0.

In previous versions of Neo4j, client-server communication used encrypted local connections and generated a self-signed certificate out of the box. In 4.0 however, the default is set to unencrypted. Please see Driver Manual → Security for more information.

Neo4j 4.0 introduces a reactive API, compatible with the Reactive Streams standard. This enables fine-grained control of the data flow for Cypher query results, including the ability to pause or cancel part-way through. Read more in Driver Manual → Queries and results.

When using the 4.0 driver to connect to a 4.0 database, it is possible to work with multiple databases. From a driver API perspective, this means that one database must be selected for use as an execution context for transactions within a session. This can be configured on session construction. If no database is selected, the driver will connect to the server's default database.

The examples in this chapter are mainly written in Java, using the Java driver. However, similar code can be translated to other languages.

6.1. New driver releases

Starting with Neo4j 4.0, the versioning scheme for the database, driver and protocol are all aligned. For supported drivers, this means that the version number will go from 1.7 to 4.0.

The new 4.0 drivers for different languages can be found with the links below:

- .NET 4.0 driver
- Java 4.0 driver
- JavaScript 4.0 driver

The Neo4j 4.0 Python and Go drivers are still under construction.

The current stable 1.7 driver releases for these languages will work in fallback mode with Neo4j 4.0. Therefore, all functionality that exists in Neo4j 3.5 will also be available in Neo4j 4.0, but new functionality introduced in 4.0 will not.

Note that a 1.7 driver communicating with a 4.0 server may need to be have encryption explicitly switched off. This is due to a change in the defaults between Neo4j 3.x and 4.0.

6.2. Compatibility

The compatibility between Neo4j 3.5 and 4.0, and 4.0 Bolt drivers is illustrated in the tables below:

Table 1. Protocols

	Neo4j 4.0	**Neo4j 3.5**
Bolt v4.0	All features fully supported.	Not supported.
Bolt v3	All features fully supported, but the support may be removed in next version.	All features fully supported.
Bolt v2	Not supported.	All features fully supported, but the support may be removed in next version.
Bolt v1	Not supported.	All features fully supported, but the support may be removed in next version.

Table 2. Drivers

	Neo4j 4.0		**Neo4j 3.5**	
	Bolt version	**Support**	**Bolt version**	**Support**

	Neo4j 4.0		Neo4j 3.5	
Java 4.0 driver	Bolt v4.0	All features fully supported.	Bolt v3	All features fully supported, but the support may be removed in next version.
.NET 4.0 driver	Bolt v4.0	All features fully supported.	Bolt v3	All features fully supported, but the support may be removed in next version.
JavaScript 4.0 driver	Bolt v4.0	All features fully supported.	Bolt v3	All features fully supported, but the support may be removed in next version.
Python 1.7 driver[1]	Bolt v3	All features partially supported.	Bolt v3	All features fully supported.
Go 1.7 driver[1]	Bolt v3	All features partially supported.	Bolt v3	All features fully supported.

[1]*Neo4j Python and Go 4.0 drivers are still under construction. Please refer to https://pypi.org/project/neo4j-driver/ and https://github.com/neo4j/neo4j-go-driver for the latest versions that are available.*

6.3. What's new?

- Bolt v4.0 is implemented in both 4.0 drivers and 4.0 servers.
- Reactive API is now available with 4.0 servers. To make use of the reactive API, the starting point is `RxSession` on the driver object.
- With 4.0 servers, session instances should now be acquired against a specific database. Causal chaining is still respected on each database (transactions cannot span across multiple databases). The driver itself connects to Neo4j DBMS.
- A new feature detection method `driver.supportsMultiDb()` is added for querying whether the remote database supports multiple databases.
- A new `driver.verifyConnectivity()` method is introduced for connectivity verification purposes. The driver instances by default will not verify DBMS availability after construction.
- New connection URI schemes with variants that contain extra encryption and trust information - `neo4j+s`, `bolt+s`, `neo4j+ssc` and `bolt+ssc`. The `+s` variants enable encryption with a full certificate check, and the `+ssc` variants enable encryption, but with no certificate check. This latter variant is designed specifically for use with self-signed certificates. For more information, see Additional URI Schemes.

Example 4. Detecting multiple database support

```
import org.neo4j.driver.Driver;
import org.neo4j.driver.Result;
import org.neo4j.driver.Session;
import org.neo4j.driver.SessionConfig;
import org.neo4j.driver.Values;

...
private final Driver driver;
...

public void printGreeting( final String message )
{
   SessionConfig sessionConfig = driver.supportsMultiDb() ? SessionConfig.forDatabase( "neo4j" )
                                                          : SessionConfig.defaultConfig();

   try ( Session session = driver.session( sessionConfig ) )
   {
       String greeting = session.writeTransaction( tx -> {
           Result result = tx.run( "CREATE (a:Greeting) SET a.message = $message RETURN a.message +
', from node ' + id(a)",
                                   Values.parameters( "message", message ) );
           return result.single().get( 0 ).asString();
       } );
       System.out.println( greeting );
   }
}
```

6.4. Breaking changes

- The driver's default configuration for encrypted is now false (meaning that driver will only attempt plain text connections by default). Connections to encrypted services (such as Neo4j Aura) should now explicitly be set to encrypted.
- When encryption is explicitly enabled, the default trust mode is to trust the CAs that are trusted by operating system. This means that encrypted connections to servers holding self-signed certificates will now fail on certificate verification by default.
- Hostname verification is turned on by default when encryption is turned on.
- `v1` is removed from drivers' package name. For example, in the Java driver, all public APIs are in the package `org.neo4j.driver` instead of the old `org.neo4j.driver.v1`.
- The `neo4j://` scheme replaces `bolt+routing://` and can be used for both clustered and single-instance configurations. This is a rename only, and `neo4j://` URIs can still be used to communicate with Neo4j 3.x clusters. Please note though that Neo4j 3.x standalone instances do not expose a routing interface.

  The `bolt://` scheme is used for direct connection to a particular Neo4j server. This scheme is no longer required for standalone machines, however. Neo4j 4.0 now exposes a routing interface for all deployment topologies, allowing `neo4j://` URIs to be used for all deployments. The `bolt://` scheme is now mainly only useful when targeting a specific machine, rather than an entire service. This can be a certain server in a Causal Cluster or the one server in a single-instance environment.
- For drivers where synchronous and asynchronous methods are both implemented, asynchronous methods have been extracted out and put in `AsyncSession`, whereas synchronous methods remain in `Session`. This change ensures that blocking and non-blocking APIs can never be mixed together.
- `Driver#session` method now makes use of a session configuration object or option builder, rather than method arguments.
- Bookmark has changed from a string, and/or a list of strings, to a Bookmark object.
- For synchronous Transaction API, `Transaction#success` and `Transaction#failure` have been removed.

The `success`/`close` pattern for Transaction objects is now obsolete and has been fully superseded by `commit` and `rollback` methods. However, unlike `Transaction#success`, which only marks the transaction to be successful and then waits for `Transaction#close` to actually perform the real commit, `Transaction#commit` commits the transaction immediately.

A transaction in 4.0 can only be committed or rolled back once. If a transaction is not committed explicitly using `Transaction#commit`, `Transaction#close` will roll back the transaction.

- `Statement` has been renamed to `Query`. `StatementResult` has been renamed to `Result`. Similarly, `StatementResultCursor` has been renamed to `ResultCursor`.
- A result can only be consumed once.

  A result is consumed if either the query result has been discarded by invoking `Result#consume`, and/or the outer scope where the result is created, such as a transaction or a session, has been closed. Attempts to access consumed results will be responded with a `ResultConsumedException`.
- `LoadBalancingStrategy` is removed from Config class, and the drivers always default to `LeastConnectedStrategy`.
- The recommended Driver Connection URI scheme is as follows:

Table 3. Recommended Driver Connection URI scheme.

		4.0 drivers	**1.7 drivers**
4.0 Neo4j	Single instance	`neo4j`	`bolt`
	Cluster core members	`neo4j`	`neo4j` (`bolt`+routing)
	Cluster read replicas	`neo4j`	`bolt`
3.5 Neo4j	Single instance	`bolt`	`bolt`
	Cluster core members	`neo4j`	`neo4j` (`bolt`+routing)
	Cluster read replicas	`bolt`	`bolt`

6.4.1. Driver-specific breaking changes

In addition to the breaking changes mentioned above, which apply in general for all drivers, the following drivers have further breaking API changes:

.NET driver

- The `Neo4j.Driver` package contains only the asynchronous API.
  - Synchronous session API (Simple API) has been moved to the namespace Neo4j.Driver.Simple.
  - Reactive API is presented in the namespace Neo4j.Driver.Reactive.
- The `IDriverLogger` has been renamed to `ILogger`.
- `TrustStrategy` is replaced with `TrustManager`.

Example 5. Migrating from the 1.7 .NET driver to the 4.0 .NET driver.

Example code for the 4.0 .NET driver

```
using Neo4j.Driver.Simple;
...
private readonly IDriver _driver;
private readonly string
_previousNeo4jSessionBookmark;
...
public void PrintGreeting(string message)
{
   using (ISession session = _driver.Session(o
=>
      o.WithDatabase("neo4j")
       .WithDefaultAccessMode(AccessMode.Write)
       .WithBookmarks
(_previousNeo4jSessionBookmark)))
   {
       using (ITransaction transaction =
session.BeginTransaction())
       {
           Query query = new Query("CREATE
(a:Greeting) SET a.message = $message RETURN
a.message + ', from node ' + id(a)", new
Dictionary<string, object>{{"message",
message}});
           IResult result = transaction.Run
(query);
               string greeting = result.
Single()[0].As<string>();
               Console.WriteLine(greeting);
               transaction.Commit(); // commit
immediately here
           }
        _previousNeo4jSessionBookmark =
session.LastBookmark;
        }
     }
```

Example code for the 1.7 .NET driver

```
using Neo4j.Driver;
...
private readonly IDriver _driver;
private readonly string
_previousSessionBookmark;
...
public void PrintGreeting(string message)
{
   using (ISession session =_driver.Session(
      AccessMode.Write,
_previousSessionBookmark))
   {


      using (ITransaction transaction =
session.BeginTransaction())
      {
         Statement query = new Statement
("CREATE (a:Greeting) SET a.message = $message
RETURN a.message + ', from node ' + id(a)", new
Dictionary<string, object>{{"message",
message}});
          IStatementResult result =
transaction.Run(query);
          transaction.Success(); // mark
success, actually commit will happen in
transaction.Dispose()
          var greeting = result.Single()[0].As
<string>();
          Console.WriteLine(greeting);
   }
   _previousSessionBookmark = session
.LastBookmark;
   }
}
```

JavaScript driver

- `session#close()` and `driver#close()` both now return `Promises`, and no longer accept callback function arguments.
- `driver.onError` and `driver.onCompleted` callbacks have been completely removed. Errors should be monitored on related code paths (i.e. through `Promise#catch`, etc.).

Example 6. Migrating from the 1.7 JavaScript driver to the 4.0 JavaScript driver.

Example code for the 4.0 JavaScript driver

```
var neo4j = require('neo4j-driver')
...
const driver = neo4j.driver(uri, neo4j.auth
.basic(user, password))
...

const session = driver.session()
try {
  const tx = session.beginTransaction()
  const result = await tx.run('CREATE
(a:Greeting) SET a.message = $message RETURN
a.message + ", from node " + id(a)', { message:
'hello, world' })
  const greeting = result.records[0].get(0)
  console.log(greeting)
  await tx.commit()
} finally {
  await session.close()
}
```

Example code for the 1.7 JavaScript driver

```
var neo4j = require('neo4j-driver').v1
...
const driver = neo4j.driver(uri, neo4j.auth
.basic(user, password))
...

const session = driver.session()
try {
  const tx = session.beginTransaction()
  const result = await tx.run('CREATE
(a:Greeting) SET a.message = $message RETURN
a.message + ", from node " + id(a)', { message:
'hello, world' })
  const greeting = result.records[0].get(0)
  console.log(greeting)
  await tx.commit()
} finally {
  session.close(callback) // another session
can be chained in callback
}
```

Java driver

Example 7. Migrating from the 1.7 Java driver to the 4.0 Java driver.

Example code for the 4.0 Java driver

```
import org.neo4j.driver.Bookmark;
import org.neo4j.driver.Driver;
import org.neo4j.driver.Query;
import org.neo4j.driver.Result;
import org.neo4j.driver.Session;
import org.neo4j.driver.SessionConfig;
import org.neo4j.driver.Transaction;
import org.neo4j.driver.Values;
...

private final Driver driver;
...
public void printGreeting( String message,
Bookmark bookmark )
{
   SessionConfig sessionConfig = SessionConfig
.builder()
     .withDatabase( "neo4j" )
     .withDefaultAccessMode( AccessMode.WRITE )
     .withBookmarks( bookmark ).build();

   try ( Session session = driver.session(
sessionConfig );
         Transaction transaction = session
.beginTransaction() )
   {
      Query query = new Query( "CREATE
(a:Greeting) SET a.message = $message RETURN
a.message + ', from node ' + id(a)", Values
.parameters( "message", message ) );

      Result result = transaction.run( query );
      String greeting = result.single().get( 0
).asString();
      System.out.println( greeting );
      transaction.commit(); // commit
immediately here
   }
}
```

Example code for the 1.7 Java driver

```
import org.neo4j.driver.v1.AccessMode;
import org.neo4j.driver.v1.Driver;
import org.neo4j.driver.v1.Session;
import org.neo4j.driver.v1.Statement;
import org.neo4j.driver.v1.StatementResult;
import org.neo4j.driver.v1.Transaction;
import org.neo4j.driver.v1.Values;

...

private final Driver driver;
...
public void printGreeting( String message,
String bookmark )
{




   try ( Session session = driver.session(
AccessMode.WRITE, bookmark );
         Transaction transaction = session
.beginTransaction() )
   {
       Statement query = new Statement( "CREATE
(a:Greeting) SET a.message = $message RETURN
a.message + ', from node ' + id(a)", Values
.parameters( "message", message ) );


       StatementResult result = transaction.
run( query );
       transaction.success(); // mark success,
actually commit will happen in
transaction.close()
       String greeting = result.single().get( 0
).asString();
       System.out.println( greeting );
   }
}
```

6.5. Back-pressure

Neo4j 4.0 introduces client-side back-pressure. The concept of client-side back-pressure is as follows; the client will communicate with the remote server regarding how much data it is able to process, and will only request additional data when it is ready to consume more.

The back-pressure concept is naturally compatible with Reactive programming. As a result, Reactive API support is added into all language drivers in 4.0 driver releases.

The Java driver Reactive API exposes a raw Publisher-Subscriber API, which is defined by reactive streams. When using Java driver’s Reactive API, it is anticipated that it is used with a reactive library, such as Project Reactor and/or RxJava.

For the .NET and JavaScript drivers Reactive API, the drivers are already supplied with reactive libraries. The built-in System.Reactive has been used in .NET driver. As for JavaScript driver, the popular RxJs library is adopted. These two libraries, as well as RxJava all belong to the same reactive framework ReactiveX.

To use the drivers’ Reactive API, a preliminary knowledge of reactive programming is necessary. Details of how to use Neo4j Reactive Driver API can be found in the Neo4j Driver Manual.

However, back-pressure is not only limited to the driver’s Reactive APIs. All other APIs, such as simple and async, by default have back-pressure enabled when handling query execution results.

Table 4. How back-pressure is implemented in the different language driver session APIs

	Simple API	**Async API**	**Reactive API**
Java driver	Record buffer	Record buffer	Raw Publisher-Subscriber API
.NET driver	Record buffer	Record buffer	Record buffer
Javascript driver	Not applicable	Record buffer	Record buffer

6.5.1. Back-pressure with Bolt v4.0 and record buffer

The Neo4j 4.0 server and drivers implement Bolt v4.0. One of the main features introduced in this Bolt version is pulling query results (records) in batches. In previous Bolt versions, the complete result set is always pulled in one batch from a server to a driver. Bolt v4.0 enables us to pull these results in multiple batches where the size of each can be defined by a `fetchSize`. By default, the drivers use a `fetchSize` of 1000 records.

With the introduction of batching of records, drivers could now implement client-side back-pressure. For each result, the driver keeps a record buffer of unconsumed records. The buffer size is the same as `fetchSize` for each batch. The pulling of records from the server will be paused when the buffer is more than 70% full, and the record pulling is re-enabled once the buffer is less than 30% full. With the default `fetchSize` of 1000 records, record pulling is paused when more than 700 records are in the buffer and is resumed when the buffer drops below 300.

Example 8. Set default `fetchSize` on driver and alter the default value on session.

```
import org.neo4j.driver.AuthTokens;
import org.neo4j.driver.Config;
import org.neo4j.driver.Driver;
import org.neo4j.driver.GraphDatabase;
import org.neo4j.driver.Session;
import org.neo4j.driver.SessionConfig;
...

Config config = Config.builder().withFetchSize( 2000 ).build();
Driver driver = GraphDatabase.driver( uri, AuthTokens.basic( user, password ), config );

SessionConfig sessionConfig = SessionConfig.builder()
                                           .withDatabase( "neo4j" )
                                           .withFetchSize( 100 )
                                           .build();
try ( Session session = driver.session( sessionConfig ) ) {...}
```

6.5.2. Java driver Reactive API

The Java driver’s Reactive API exposes a very low level Publisher-Subscriber API. As a result, it will not perform any kind of back-pressure by default. Instead, we expect driver users to make use of a reactive framework to utilise back-pressure. Depending on the choice of the reactive framework, the framework may apply back-pressure by pausing the pulling of the data from a Neo4j server, or dropping data when there is too much to process.

6.6. Multiple databases

With the addition of multiple databases in 4.0, you can now specify which database to work with. When constructing a session you can specify in the session configuration which database the session is linked to. Queries will then be executed against that database for the duration of the session. Not specifying a database will result in the session being linked to the default database as specified in the server configuration, see Operations manual → The default database. When using 4.0 drivers with 4.0 Neo4j Servers, we always recommend to specify the database of each session explicitly.

Example 9. Selecting a database for a session.

```
import org.neo4j.driver.Driver;
import org.neo4j.driver.Session;
import org.neo4j.driver.SessionConfig;
...

try ( Session session = driver.session( SessionConfig.forDatabase( "neo4j" ) ) ) {...}
```

While managing multiple databases is primarily a feature for Neo4j Enterprise Edition, users of Neo4j Community Edition will still need to use the `system` database when carrying out administrative operations on the database. See Operations manual → The `system` database and Cypher manual → Administration for more information.

6.6.1. Bookmarks

Bookmarks are generally handled internally by the driver. Applications typically only need to work with bookmarks directly when chaining sessions.

When using bookmarks in a multiple database context, the base rule is that bookmarks can only be passed among sessions for the same database. This is because the bookmarks (and/or transactions) cannot cross multiple databases in Neo4j 4.0. There is one exception however, the bookmarks generated by the `system` database can be used with other databases.

Example 10. Using system bookmark with another database to ensure the updated system status.

```
import org.neo4j.driver.Bookmark;
import org.neo4j.driver.Driver;
import org.neo4j.driver.Result;
import org.neo4j.driver.Session;
import org.neo4j.driver.SessionConfig;
...

Bookmark sysBookmark;
try ( Session session = driver.session( SessionConfig.forDatabase( "system" ) ) )
{
   session.writeTransaction( tx -> {
       Result result = tx.run( "CREATE database foo" );
       return result.consume();
   } );
   sysBookmark = session.lastBookmark();
}

try ( Session session = driver.session( SessionConfig.builder().withDatabase( "foo" ).withBookmarks(
sysBookmark ).build() ) )
{
   session.writeTransaction( tx -> {
       Result result = tx.run( "CREATE (n)" );
       return result.consume();
   } );
}
```

6.7. Configure SSL Policy for Bolt server and HTTPS server

Neo4j 3.5 always allows encrypted connections with the default configuration. In case no certificate is installed before a server starts, self-signed certificates will be automatically generated. However, in 4.0

the default encryption setting is off and Neo4j will no longer generate certificates when none is provided. As a result, Bolt server only allows plaintext connections, and HTTPS server is not enabled by default. The table below summarizes the default behaviour change between 3.5 and 4.0 regarding encryption and certificates.

Table 5. Encryption and certificates differences between 3.5 and 4.0 servers.

	3.5 Neo4j Bolt Server	4.0 Neo4j Bolt Server	3.5 Neo4j HTTPS Server	4.0 Neo4j HTTPS Server
Server Enabled	Yes	Yes	Yes	No
Encryption on client connections	Optional	Not allowed	Always	Always
Certificates	Auto-generated self-signed certificates if not provided	None	Auto-generated self-signed certificates if not provided	None
Default Certificates Path	`$neo4jHome/certificates`	None	`$neo4jHome/certificates`	None
Default Certificate Names	`neo4j.key` `neo4j.cert`	`private.key` `public.crt`	`neo4j.key` `neo4j.cert`	`private.key` `public.crt`

In order to re-enable encryption in 4.0, we need to configure the SSL policy in the Neo4j config file. Given certificates named `public.crt` and `private.key` in folder `$neo4jHome/certificates/bolt` for Bolt server, and certificates with the same file names in folder `$neo4jHome/certificates/https` for HTTPS server. The example below shows how to turn encryption back on for the Bolt server and re-enable the HTTPS server.

Example 11. Turn encryption on for Bolt v4.0 server.

```
dbms.connector.bolt.enabled=true
dbms.connector.bolt.tls_level=OPTIONAL          # allows both encrypted and unencrypted driver
connections

dbms.ssl.policy.bolt.enabled=true
dbms.ssl.policy.bolt.base_directory=certificates/bolt
#dbms.ssl.policy.bolt.private_key=private.key   # Optional if the file name is the same as the
default.
#dbms.ssl.policy.bolt.public_certificate=public.crt     # Optional if the file name is the same as
the default.
```

Example 12. Enable the HTTPS 4.0 server.

```
dbms.connector.https.enabled=true

dbms.ssl.policy.https.enabled=true
dbms.ssl.policy.https.base_directory=certificates/https
#dbms.ssl.policy.https.private_key=private.key  # Optional if the file name is the same as the
default.
#dbms.ssl.policy.https.public_certificate=public.crt    # Optional if the file name is the same as
the default.
```

6.8. Additional URI Schemes

Since `v4.0.1` of the Java and .NET drivers, and `v4.0.2` of the JavaScript driver, you are able to configure the encryption and trust settings of the driver directly through the connection URI.

The `neo4j+s` and `bolt+s` schemes enable encryption and full certificate checks against the system's local CA store. The `neo4j+ssc` and `bolt+ssc` schemes also enable encryption with no certificate checks, typically for use with self-signed certificates.

Table 6. Available URIs

URI	Routing	Description
`neo4j`	Yes	Unsecured
`neo4j+s`	Yes	Secured with full certificate
`neo4j+ssc`	Yes	Secured with self-signed certificate
`bolt`	No	Unsecured
`bolt+s`	No	Secured with full certificate
`bolt+ssc`	No	Secured with self-signed certificate

Using these new URI schemes is not compatible with configuring encryption and trust with the Configuration API. Otherwise, this does not effect the behaviour of the existing `neo4j` and `bolt` schemes.

For more information, see Driver Manual Connection URIs.

Appendix A: Classes removed from public API

This appendix lists the classes that have been removed or excluded from the public API between Neo4j 3.5 and 4.0.

The following table lists classes that have been removed or excluded from the public API:

Classes removed or excluded from the public API
`org.neo4j.backup.BackupExtensionService`
`org.neo4j.backup.BackupTool`
`org.neo4j.backup.IncrementalBackupNotPossibleException`
`org.neo4j.backup.OnlineBackupExtensionFactory.Dependencies`
`org.neo4j.backup.OnlineBackupExtensionFactory`
`org.neo4j.backup.OnlineBackupKernelExtension.BackupProvider`
`org.neo4j.backup.OnlineBackupKernelExtension`

Classes removed or excluded from the public API
`org.neo4j.backup.OnlineBackupSettings`
`org.neo4j.backup.TheBackupInterface`
`org.neo4j.cypher.export.CypherResultSubGraph`
`org.neo4j.cypher.export.DatabaseSubGraph`
`org.neo4j.cypher.export.SubGraphExporter`
`org.neo4j.cypher.export.SubGraph`
`org.neo4j.graphalgo.CommonEvaluators`
`org.neo4j.graphalgo.MaxCostEvaluator`
`org.neo4j.graphdb.DatabaseShutdownException`
`org.neo4j.graphdb.DependencyResolver.Adapter`
`org.neo4j.graphdb.DependencyResolver.SelectionStrategy`
`org.neo4j.graphdb.DependencyResolver`
`org.neo4j.graphdb.DynamicLabel`
`org.neo4j.graphdb.DynamicRelationshipType`
`org.neo4j.graphdb.InvalidTransactionTypeException`
`org.neo4j.graphdb.PathExpanderBuilder`
`org.neo4j.graphdb.PathExpanders`
`org.neo4j.graphdb.PropertyContainer`
`org.neo4j.graphdb.ResourceUtils`
`org.neo4j.graphdb.TransactionGuardException`
`org.neo4j.graphdb.TransientDatabaseFailureException`
`org.neo4j.graphdb.TransientFailureException`
`org.neo4j.graphdb.TransientTransactionFailureException`
`org.neo4j.graphdb.config.BaseSetting`
`org.neo4j.graphdb.config.InvalidSettingException`
`org.neo4j.graphdb.config.ScopeAwareSetting`
`org.neo4j.graphdb.config.SettingGroup`
`org.neo4j.graphdb.config.SettingValidator`
`org.neo4j.graphdb.event.ErrorState`
`org.neo4j.graphdb.event.KernelEventHandler.ExecutionOrder`
`org.neo4j.graphdb.event.KernelEventHandler`
`org.neo4j.graphdb.event.TransactionEventHandler.Adapter`
`org.neo4j.graphdb.facade.GraphDatabaseDependencies`
`org.neo4j.graphdb.facade.GraphDatabaseFacadeFactory.Dependencies`
`org.neo4j.graphdb.facade.GraphDatabaseFacadeFactory`
`org.neo4j.graphdb.facade.embedded.EmbeddedGraphDatabase`
`org.neo4j.graphdb.facade.spi.ClassicCoreSPI`
`org.neo4j.graphdb.facade.spi.ProcedureGDBFacadeSPI`
`org.neo4j.graphdb.factory.Description`
`org.neo4j.graphdb.factory.EditionLocksFactories`

Classes removed or excluded from the public API
`org.neo4j.graphdb.factory.EnterpriseGraphDatabaseFactory`
`org.neo4j.graphdb.factory.GraphDatabaseBuilder.DatabaseCreator`
`org.neo4j.graphdb.factory.GraphDatabaseBuilder.Delegator`
`org.neo4j.graphdb.factory.GraphDatabaseBuilder`
`org.neo4j.graphdb.factory.GraphDatabaseFactoryState`
`org.neo4j.graphdb.factory.GraphDatabaseFactory`
`org.neo4j.graphdb.factory.GraphDatabaseSettings.Connector.ConnectorType`
`org.neo4j.graphdb.factory.GraphDatabaseSettings.Connector`
`org.neo4j.graphdb.factory.GraphDatabaseSettings.LabelIndex`
`org.neo4j.graphdb.factory.HighlyAvailableGraphDatabaseFactory`
`org.neo4j.graphdb.factory.module.DataSourceModule`
`org.neo4j.graphdb.factory.module.ModularDatabaseCreationContext`
`org.neo4j.graphdb.factory.module.PlatformModule`
`org.neo4j.graphdb.factory.module.ProcedureGDSFactory`
`org.neo4j.graphdb.factory.module.edition.AbstractEditionModule`
`org.neo4j.graphdb.factory.module.edition.CommunityEditionModule`
`org.neo4j.graphdb.factory.module.edition.DefaultEditionModule`
`org.neo4j.graphdb.factory.module.edition.context.DatabaseEditionContext`
`org.neo4j.graphdb.factory.module.edition.context.DefaultEditionModuleDatabaseContext`
`org.neo4j.graphdb.factory.module.id.DatabaseIdContext`
`org.neo4j.graphdb.factory.module.id.IdContextFactoryBuilder`
`org.neo4j.graphdb.factory.module.id.IdContextFactory`
`org.neo4j.graphdb.index.AutoIndexer`
`org.neo4j.graphdb.index.IndexHits`
`org.neo4j.graphdb.index.IndexManager`
`org.neo4j.graphdb.index.IndexPopulationProgress`
`org.neo4j.graphdb.index.Index`
`org.neo4j.graphdb.index.ReadableIndex`
`org.neo4j.graphdb.index.ReadableRelationshipIndex`
`org.neo4j.graphdb.index.RelationshipAutoIndexer`
`org.neo4j.graphdb.index.RelationshipIndex`
`org.neo4j.graphdb.index.UniqueFactory.UniqueEntity`
`org.neo4j.graphdb.index.UniqueFactory.UniqueNodeFactory`
`org.neo4j.graphdb.index.UniqueFactory.UniqueRelationshipFactory`
`org.neo4j.graphdb.index.UniqueFactory`
`org.neo4j.graphdb.security.AuthProviderFailedException`
`org.neo4j.graphdb.security.AuthProviderTimeoutException`
`org.neo4j.graphdb.security.AuthorizationExpiredException`
`org.neo4j.graphdb.security.AuthorizationViolationException`
`org.neo4j.graphdb.security.URLAccessRule`

Classes removed or excluded from the public API
org.neo4j.graphdb.security.URLAccessValidationError
org.neo4j.graphdb.security.WriteOperationsNotAllowedException
org.neo4j.graphdb.traversal.AlternatingSelectorOrderer
org.neo4j.graphdb.traversal.BidirectionalTraversalDescription
org.neo4j.graphdb.traversal.BidirectionalUniquenessFilter
org.neo4j.graphdb.traversal.BranchCollisionDetector
org.neo4j.graphdb.traversal.BranchCollisionPolicies
org.neo4j.graphdb.traversal.BranchCollisionPolicy
org.neo4j.graphdb.traversal.BranchOrderingPolicies
org.neo4j.graphdb.traversal.BranchOrderingPolicy
org.neo4j.graphdb.traversal.BranchSelector
org.neo4j.graphdb.traversal.BranchState
org.neo4j.graphdb.traversal.Evaluation
org.neo4j.graphdb.traversal.Evaluator.AsPathEvaluator
org.neo4j.graphdb.traversal.Evaluator
org.neo4j.graphdb.traversal.Evaluators
org.neo4j.graphdb.traversal.InitialBranchState.Adapter
org.neo4j.graphdb.traversal.InitialBranchState.State
org.neo4j.graphdb.traversal.InitialBranchState
org.neo4j.graphdb.traversal.LevelSelectorOrderer
org.neo4j.graphdb.traversal.PathEvaluator.Adapter
org.neo4j.graphdb.traversal.PathEvaluator
org.neo4j.graphdb.traversal.Paths.DefaultPathDescriptor
org.neo4j.graphdb.traversal.Paths.PathDescriptor
org.neo4j.graphdb.traversal.Paths
org.neo4j.graphdb.traversal.SideSelectorPolicies
org.neo4j.graphdb.traversal.SideSelectorPolicy
org.neo4j.graphdb.traversal.SideSelector
org.neo4j.graphdb.traversal.Sorting
org.neo4j.graphdb.traversal.TraversalBranch
org.neo4j.graphdb.traversal.TraversalContext
org.neo4j.graphdb.traversal.TraversalDescription
org.neo4j.graphdb.traversal.TraversalMetadata
org.neo4j.graphdb.traversal.Traverser
org.neo4j.graphdb.traversal.UniquenessFactory
org.neo4j.graphdb.traversal.UniquenessFilter
org.neo4j.graphdb.traversal.Uniqueness
org.neo4j.helpers.AdvertisedSocketAddress
org.neo4j.helpers.Args.ArgsParser
org.neo4j.helpers.Args.Option

Classes removed or excluded from the public API
`org.neo4j.helpers.Args`
`org.neo4j.helpers.ArrayUtil.ArrayEquality`
`org.neo4j.helpers.ArrayUtil`
`org.neo4j.helpers.Assertion`
`org.neo4j.helpers.Cancelable`
`org.neo4j.helpers.CancellationRequest`
`org.neo4j.helpers.Clock`
`org.neo4j.helpers.CloneableInPublic`
`org.neo4j.helpers.Exceptions`
`org.neo4j.helpers.Format`
`org.neo4j.helpers.FutureAdapter.Present`
`org.neo4j.helpers.FutureAdapter`
`org.neo4j.helpers.HostnamePort`
`org.neo4j.helpers.ListenSocketAddress`
`org.neo4j.helpers.Listeners.Notification`
`org.neo4j.helpers.Listeners`
`org.neo4j.helpers.MathUtil`
`org.neo4j.helpers.NamedThreadFactory.Monitor`
`org.neo4j.helpers.NamedThreadFactory`
`org.neo4j.helpers.Numbers`
`org.neo4j.helpers.PortBindException`
`org.neo4j.helpers.ProcessFailureException.Entry`
`org.neo4j.helpers.ProcessFailureException`
`org.neo4j.helpers.Reference`
`org.neo4j.helpers.RunCarefully`
`org.neo4j.helpers.Service.Implementation`
`org.neo4j.helpers.Service`
`org.neo4j.helpers.SocketAddressParser`
`org.neo4j.helpers.Strings`
`org.neo4j.helpers.TaskControl`
`org.neo4j.helpers.TaskCoordinator`
`org.neo4j.helpers.TextUtil`
`org.neo4j.helpers.ThisShouldNotHappenError`
`org.neo4j.helpers.TimeUtil`
`org.neo4j.helpers.TransactionTemplate.Monitor.Adapter`
`org.neo4j.helpers.TransactionTemplate.Monitor`
`org.neo4j.helpers.TransactionTemplate`
`org.neo4j.helpers.Uris`
`org.neo4j.helpers.collection.ArrayIterator`
`org.neo4j.helpers.collection.BoundedIterable`

Classes removed or excluded from the public API
`org.neo4j.helpers.collection.CachingIterator`
`org.neo4j.helpers.collection.CastingIterator`
`org.neo4j.helpers.collection.CatchingIteratorWrapper`
`org.neo4j.helpers.collection.CollectorsUtil`
`org.neo4j.helpers.collection.CombiningIterable`
`org.neo4j.helpers.collection.CombiningIterator`
`org.neo4j.helpers.collection.CombiningResourceIterator`
`org.neo4j.helpers.collection.ExceptionHandlingIterable`
`org.neo4j.helpers.collection.FilteringIterable`
`org.neo4j.helpers.collection.FilteringIterator`
`org.neo4j.helpers.collection.FirstItemIterable`
`org.neo4j.helpers.collection.IterableWrapper`
`org.neo4j.helpers.collection.Iterables`
`org.neo4j.helpers.collection.IteratorWrapper`
`org.neo4j.helpers.collection.Iterators`
`org.neo4j.helpers.collection.LimitingResourceIterable`
`org.neo4j.helpers.collection.LimitingResourceIterator`
`org.neo4j.helpers.collection.LruCache`
`org.neo4j.helpers.collection.MapUtil.MapBuilder`
`org.neo4j.helpers.collection.MapUtil`
`org.neo4j.helpers.collection.MappingResourceIterator`
`org.neo4j.helpers.collection.MultiSet`
`org.neo4j.helpers.collection.NestingIterable`
`org.neo4j.helpers.collection.NestingIterator`
`org.neo4j.helpers.collection.NestingResourceIterator`
`org.neo4j.helpers.collection.PagingIterator`
`org.neo4j.helpers.collection.Pair`
`org.neo4j.helpers.collection.PrefetchingIterator`
`org.neo4j.helpers.collection.PrefetchingResourceIterator`
`org.neo4j.helpers.collection.RangeIterator`
`org.neo4j.helpers.collection.ResourceClosingIterator`
`org.neo4j.helpers.collection.ResourceIterableWrapper`
`org.neo4j.helpers.collection.ReverseArrayIterator`
`org.neo4j.helpers.collection.Visitable`
`org.neo4j.helpers.collection.Visitor.SafeGenerics`
`org.neo4j.helpers.collection.Visitor`
`org.neo4j.index.lucene.LuceneKernelExtensionFactory.Dependencies`
`org.neo4j.index.lucene.LuceneKernelExtensionFactory`
`org.neo4j.index.lucene.LuceneKernelExtension`
`org.neo4j.index.lucene.LuceneTimeline`

Classes removed or excluded from the public API
`org.neo4j.index.lucene.QueryContext`
`org.neo4j.index.lucene.TimelineIndex`
`org.neo4j.index.lucene.ValueContext`
`org.neo4j.index.lucene.unsafe.batchinsert.LuceneBatchInserterIndexProvider`
`org.neo4j.jmx.Description`
`org.neo4j.jmx.JmxUtils`
`org.neo4j.jmx.Kernel`
`org.neo4j.jmx.ManagementInterface`
`org.neo4j.jmx.Primitives`
`org.neo4j.jmx.StoreFile`
`org.neo4j.jmx.StoreSize`
`org.neo4j.logging.AbstractLogProvider`
`org.neo4j.logging.AbstractLog`
`org.neo4j.logging.AbstractPrintWriterLogger`
`org.neo4j.logging.BufferingLog`
`org.neo4j.logging.DuplicatingLogProvider`
`org.neo4j.logging.DuplicatingLog`
`org.neo4j.logging.FormattedLog.Builder`
`org.neo4j.logging.FormattedLogProvider.Builder`
`org.neo4j.logging.FormattedLogProvider`
`org.neo4j.logging.FormattedLog`
`org.neo4j.logging.NullLogProvider`
`org.neo4j.logging.NullLog`
`org.neo4j.logging.NullLogger`
`org.neo4j.logging.PrintStreamLogger`
`org.neo4j.logging.RotatingFileOutputStreamSupplier.RotationListener`
`org.neo4j.logging.RotatingFileOutputStreamSupplier`
`org.neo4j.logging.slf4j.Slf4jLogProvider`
`org.neo4j.logging.slf4j.Slf4jLog`
`org.neo4j.management.BranchedStoreInfo`
`org.neo4j.management.BranchedStore`
`org.neo4j.management.CausalClustering`
`org.neo4j.management.ClusterDatabaseInfo`
`org.neo4j.management.ClusterMemberInfo`
`org.neo4j.management.Diagnostics`
`org.neo4j.management.HighAvailability`
`org.neo4j.management.IndexSamplingManager`
`org.neo4j.management.LockManager`
`org.neo4j.management.MemoryMapping`
`org.neo4j.management.Neo4jManager`

Classes removed or excluded from the public API
`org.neo4j.management.PageCache`
`org.neo4j.management.RemoteConnection`
`org.neo4j.management.TransactionManager`
`org.neo4j.management.WindowPoolInfo`
`org.neo4j.procedure.Admin`
`org.neo4j.procedure.PerformsWrites`
`org.neo4j.procedure.ProcedureTransaction`
`org.neo4j.procedure.TerminationGuard`
`org.neo4j.server.helpers.PropertyTypeDispatcher.PropertyArray`
`org.neo4j.server.helpers.PropertyTypeDispatcher`
`org.neo4j.server.plugins.BadPluginInvocationException`
`org.neo4j.server.plugins.ConfigAdapter`
`org.neo4j.server.plugins.DefaultPluginManager`
`org.neo4j.server.plugins.Description`
`org.neo4j.server.plugins.DisabledPluginManager`
`org.neo4j.server.plugins.Injectable`
`org.neo4j.server.plugins.MapTypeCaster`
`org.neo4j.server.plugins.Name`
`org.neo4j.server.plugins.ParameterDescriptionConsumer`
`org.neo4j.server.plugins.ParameterList`
`org.neo4j.server.plugins.Parameter`
`org.neo4j.server.plugins.PluginInvocationFailureException`
`org.neo4j.server.plugins.PluginInvocatorProvider`
`org.neo4j.server.plugins.PluginInvocator`
`org.neo4j.server.plugins.PluginLifecycle`
`org.neo4j.server.plugins.PluginLookupException`
`org.neo4j.server.plugins.PluginManager`
`org.neo4j.server.plugins.PluginPoint`
`org.neo4j.server.plugins.PluginTarget`
`org.neo4j.server.plugins.SPIPluginLifecycle`
`org.neo4j.server.plugins.ServerExtender`
`org.neo4j.server.plugins.ServerPlugin`
`org.neo4j.server.plugins.Source`
`org.neo4j.server.rest.repr.AuthorizationRepresentation`
`org.neo4j.server.rest.repr.BadInputException`
`org.neo4j.server.rest.repr.ConstraintDefinitionRepresentation`
`org.neo4j.server.rest.repr.CypherPlanRepresentation`
`org.neo4j.server.rest.repr.CypherRepresentationDispatcher`
`org.neo4j.server.rest.repr.CypherResultRepresentation`
`org.neo4j.server.rest.repr.CypherStatisticsRepresentation`

Classes removed or excluded from the public API
org.neo4j.server.rest.repr.DatabaseRepresentation
org.neo4j.server.rest.repr.DefaultFormat
org.neo4j.server.rest.repr.DiscoveryRepresentation
org.neo4j.server.rest.repr.EntityRepresentation
org.neo4j.server.rest.repr.ExceptionRepresentation
org.neo4j.server.rest.repr.ExtensionInjector
org.neo4j.server.rest.repr.ExtensionPointRepresentation
org.neo4j.server.rest.repr.FullPath
org.neo4j.server.rest.repr.IndexDefinitionRepresentation
org.neo4j.server.rest.repr.IndexRepresentation
org.neo4j.server.rest.repr.IndexedEntityRepresentation
org.neo4j.server.rest.repr.InputFormatProvider
org.neo4j.server.rest.repr.InputFormat
org.neo4j.server.rest.repr.InvalidArgumentsException
org.neo4j.server.rest.repr.ListRepresentation
org.neo4j.server.rest.repr.ListSerializer
org.neo4j.server.rest.repr.ListWriter
org.neo4j.server.rest.repr.MapRepresentation
org.neo4j.server.rest.repr.MappingRepresentation
org.neo4j.server.rest.repr.MappingSerializer
org.neo4j.server.rest.repr.MappingWriter
org.neo4j.server.rest.repr.MediaTypeNotSupportedException
org.neo4j.server.rest.repr.NodeIndexRepresentation
org.neo4j.server.rest.repr.NodeIndexRootRepresentation
org.neo4j.server.rest.repr.NodeRepresentation
org.neo4j.server.rest.repr.ObjectRepresentation
org.neo4j.server.rest.repr.ObjectToRepresentationConverter
org.neo4j.server.rest.repr.OutputFormatProvider
org.neo4j.server.rest.repr.OutputFormat
org.neo4j.server.rest.repr.PathRepresentation
org.neo4j.server.rest.repr.PropertiesRepresentation
org.neo4j.server.rest.repr.RelationshipIndexRepresentation
org.neo4j.server.rest.repr.RelationshipIndexRootRepresentation
org.neo4j.server.rest.repr.RelationshipRepresentation
org.neo4j.server.rest.repr.RepresentationDispatcher
org.neo4j.server.rest.repr.RepresentationExceptionHandlingIterable
org.neo4j.server.rest.repr.RepresentationFormatRepository
org.neo4j.server.rest.repr.RepresentationFormat
org.neo4j.server.rest.repr.RepresentationType
org.neo4j.server.rest.repr.RepresentationWriteHandler

Classes removed or excluded from the public API
`org.neo4j.server.rest.repr.Representation`
`org.neo4j.server.rest.repr.ScoredEntityRepresentation`
`org.neo4j.server.rest.repr.ScoredNodeRepresentation`
`org.neo4j.server.rest.repr.ScoredRelationshipRepresentation`
`org.neo4j.server.rest.repr.ServerExtensionRepresentation`
`org.neo4j.server.rest.repr.ServerListRepresentation`
`org.neo4j.server.rest.repr.StreamingFormat`
`org.neo4j.server.rest.repr.ValueRepresentation`
`org.neo4j.server.rest.repr.WeightedPathRepresentation`
`org.neo4j.server.rest.web.BatchOperationService`
`org.neo4j.server.rest.web.CollectUserAgentFilter`
`org.neo4j.server.rest.web.CorsFilter`
`org.neo4j.server.rest.web.CustomStatusType`
`org.neo4j.server.rest.web.CypherService`
`org.neo4j.server.rest.web.DatabaseActions.Provider`
`org.neo4j.server.rest.web.DatabaseActions.RelationshipDirection`
`org.neo4j.server.rest.web.DatabaseActions`
`org.neo4j.server.rest.web.DatabaseMetadataService`
`org.neo4j.server.rest.web.ExtensionService`
`org.neo4j.server.rest.web.HttpConnectionInfoFactory`
`org.neo4j.server.rest.web.InternalJettyServletRequest.RequestData`
`org.neo4j.server.rest.web.InternalJettyServletRequest`
`org.neo4j.server.rest.web.InternalJettyServletResponse`
`org.neo4j.server.rest.web.NoSuchPropertyException`
`org.neo4j.server.rest.web.NodeNotFoundException`
`org.neo4j.server.rest.web.PropertyValueException`
`org.neo4j.server.rest.web.RelationshipNotFoundException`
`org.neo4j.server.rest.web.RestfulGraphDatabase.AmpersandSeparatedCollection`
`org.neo4j.server.rest.web.RestfulGraphDatabase`
`org.neo4j.server.rest.web.StreamingBatchOperations`
`org.neo4j.server.rest.web.Surface`
`org.neo4j.server.rest.web.TransactionUriScheme`
`org.neo4j.server.rest.web.TransactionalService.TransactionUriBuilder`
`org.neo4j.server.rest.web.TransactionalService`
`org.neo4j.unsafe.batchinsert.BatchInserterIndexProvider`
`org.neo4j.unsafe.batchinsert.BatchInserterIndex`
`org.neo4j.unsafe.batchinsert.BatchInserter`
`org.neo4j.unsafe.batchinsert.BatchInserters`
`org.neo4j.unsafe.batchinsert.BatchRelationship`

Appendix B: External dependencies

This appendix lists the external dependencies in Neo4j 4.0.

The following table lists the external dependencies in Neo4j 4.0:

Group Id	Artifact Id	Version
com.fasterxml.jackson.core	jackson-annotations	2.10.0
com.fasterxml.jackson.core	jackson-core	2.10.0
com.fasterxml.jackson.core	jackson-databind	2.10.0
com.fasterxml.jackson.jaxrs	jackson-jaxrs-base	2.10.0
com.fasterxml.jackson.jaxrs	jackson-jaxrs-json-provider	2.10.0
com.fasterxml.jackson.module	jackson-module-jaxb-annotations	2.10.0
com.github.ben-manes.caffeine	caffeine	2.8.0
com.github.luben	zstd-jni	1.4.3-1
commons-beanutils	commons-beanutils	1.9.4
commons-collections	commons-collections	3.2.2
commons-configuration	commons-configuration	1.10
commons-io	commons-io	2.6
commons-lang	commons-lang	2.6
commons-logging	commons-logging	1.2
com.profesorfalken	jPowerShell	3.0
com.profesorfalken	WMI4Java	1.6.3
com.sun.activation	jakarta.activation	1.2.1
com.sun.istack	istack-commons-runtime	3.0.8
com.sun.xml.fastinfoset	FastInfoset	1.2.16
com.typesafe.akka	akka-actor_2.12	2.5.22
com.typesafe.akka	akka-cluster_2.12	2.5.22
com.typesafe.akka	akka-cluster-tools_2.12	2.5.22
com.typesafe.akka	akka-coordination_2.12	2.5.22
com.typesafe.akka	akka-distributed-data_2.12	2.5.22
com.typesafe.akka	akka-protobuf_2.12	2.5.22
com.typesafe.akka	akka-remote_2.12	2.5.22
com.typesafe.akka	akka-stream_2.12	2.5.22
com.typesafe	config	1.3.3
com.typesafe	ssl-config-core_2.12	0.3.7
info.picocli	picocli	4.0.4
io.aeron	aeron-client	1.15.1
io.aeron	aeron-driver	1.15.1
io.dropwizard.metrics	metrics-core	4.1.0
io.dropwizard.metrics	metrics-graphite	4.1.0

Group Id	Artifact Id	Version
io.dropwizard.metrics	metrics-jmx	4.1.0
io.netty	netty-all	4.1.35.Final
io.netty	netty	3.10.6.Final
io.projectreactor	reactor-core	3.2.10.RELEASE
io.prometheus	simpleclient_common	0.7.0
io.prometheus	simpleclient_dropwizard	0.7.0
io.prometheus	simpleclient_httpserver	0.7.0
io.prometheus	simpleclient	0.7.0
jakarta.activation	jakarta.activation-api	1.2.1
jakarta.annotation	jakarta.annotation-api	1.3.4
jakarta.ws.rs	jakarta.ws.rs-api	2.1.5
jakarta.xml.bind	jakarta.xml.bind-api	2.3.2
javax.activation	activation	1.1.1
javax.servlet	javax.servlet-api	3.1.0
javax.validation	validation-api	2.0.1.Final
javax.ws.rs	javax.ws.rs-api	2.1.1
javax.xml.bind	jaxb-api	2.3.0
jline	jline	2.14.3
net.java.dev.jna	jna	5.4.0
net.jpountz.lz4	lz4	1.3.0
org.agrona	agrona	0.9.31
org.apache.commons	commons-compress	1.19
org.apache.commons	commons-lang3	3.9
org.apache.commons	commons-text	1.7
org.apache.lucene	lucene-analyzers-common	8.2.0
org.apache.lucene	lucene-codecs	8.2.0
org.apache.lucene	lucene-core	8.2.0
org.apache.lucene	lucene-queryparser	8.2.0
org.apache.shiro	shiro-cache	1.4.1
org.apache.shiro	shiro-config-core	1.4.1
org.apache.shiro	shiro-config-ogdl	1.4.1
org.apache.shiro	shiro-core	1.4.1
org.apache.shiro	shiro-crypto-cipher	1.4.1
org.apache.shiro	shiro-crypto-core	1.4.1
org.apache.shiro	shiro-crypto-hash	1.4.1
org.apache.shiro	shiro-event	1.4.1
org.apache.shiro	shiro-lang	1.4.1
org.bitbucket.inkytonik.kiama	kiama_2.12	2.1.0
org.bouncycastle	bcpkix-jdk15on	1.63
org.bouncycastle	bcprov-jdk15on	1.63

Group Id	Artifact Id	Version
org.eclipse.collections	eclipse-collections-api	10.0.0
org.eclipse.collections	eclipse-collections	10.0.0
org.eclipse.jetty	jetty-client	9.4.17.v20190418
org.eclipse.jetty	jetty-http	9.4.17.v20190418
org.eclipse.jetty	jetty-io	9.4.17.v20190418
org.eclipse.jetty	jetty-security	9.4.17.v20190418
org.eclipse.jetty	jetty-server	9.4.17.v20190418
org.eclipse.jetty	jetty-servlet	9.4.17.v20190418
org.eclipse.jetty	jetty-util	9.4.17.v20190418
org.eclipse.jetty	jetty-webapp	9.4.17.v20190418
org.eclipse.jetty	jetty-xml	9.4.17.v20190418
org.glassfish.hk2.external	jakarta.inject	2.5.0
org.glassfish.hk2	hk2-api	2.5.0
org.glassfish.hk2	hk2-locator	2.5.0
org.glassfish.hk2	hk2-utils	2.5.0
org.glassfish.jaxb	jaxb-runtime	2.3.2
org.glassfish.jaxb	txw2	2.3.2
org.glassfish.jersey.containers	jersey-container-servlet-core	2.29
org.glassfish.jersey.containers	jersey-container-servlet	2.29
org.glassfish.jersey.core	jersey-client	2.29
org.glassfish.jersey.core	jersey-common	2.29
org.glassfish.jersey.core	jersey-server	2.29
org.glassfish.jersey.inject	jersey-hk2	2.29
org.glassfish.jersey.media	jersey-media-jaxb	2.29
org.javassist	javassist	3.22.0-CR2
org.jprocesses	jProcesses	1.6.5
org.jvnet.staxex	stax-ex	1.8.1
org.neo4j.licensing-proxy	zstd-proxy	4.0.0-SNAPSHOT
org.ow2.asm	asm-analysis	7.2
org.ow2.asm	asm	7.2
org.ow2.asm	asm-tree	7.2
org.ow2.asm	asm-util	7.2
org.parboiled	parboiled-core	1.2.0
org.parboiled	parboiled-scala_2.12	1.2.0
org.reactivestreams	reactive-streams	1.0.2
org.rogach	scallop_2.12	2.1.1
org.scala-lang.modules	scala-java8-compat_2.12	0.8.0
org.scala-lang.modules	scala-parser-combinators_2.12	1.1.1
org.scala-lang	scala-library	2.12.7
org.scala-lang	scala-reflect	2.12.7

Group Id	Artifact Id	Version
`org.slf4j`	`slf4j-api`	`1.7.25`
`org.slf4j`	`slf4j-nop`	`1.7.25`