# AdminManual Metastore Administration

## Hive Metastore Administration



This page only documents the MetaStore in Hive 2.x and earlier. For 3.x and later releases please see AdminManual Metastore 3.0 Administration

### Introduction

All the metadata for Hive tables and partitions are accessed through the Hive Metastore. Metadata is persisted using JPOX ORM solution (Data Nucleus) so any database that is supported by it can be used by Hive. Most of the commercial relational databases and many open source databases are supported. See the list of supported databases in section below.

You can find an E/R diagram for the metastore here.

There are 2 different ways to setup the metastore server and metastore database using different Hive configurations:

Configuration options for **metastore database** where metadata is persisted:

- Local/Embedded Metastore Database (Derby)
- Remote Metastore Database

Configuration options for **metastore server**:

- Local/Embedded Metastore Server
- Remote Metastore Server

#### Basic Configuration Parameters

The relevant configuration parameters are shown here. (Non-metastore parameters are described in Configuring Hive. Also see the Language Manual's Hive Configuration Properties, including Metastore and Hive Metastore Security.)

Also see hivemetastore-site.xml documentation under Configuring Hive.

| Configuration Parameter | Description |
|---|---|
| javax.jdo.option.ConnectionURL | JDBC connection string for the data store which contains metadata |
| javax.jdo.option.<br>ConnectionDriverName | JDBC Driver class name for the data store which contains metadata |
| hive.metastore.uris | Hive connects to one of these URIs to make metadata requests to a remote Metastore (comma separated list of URIs) |
| hive.metastore.local | local or remote metastore (removed as of Hive 0.10: If `hive.metastore.uris` is empty `local` mode is assumed, `remote` otherwise) |
| hive.metastore.warehouse.dir | URI of the default location for native tables |

The Hive metastore is stateless and thus there can be multiple instances to achieve High Availability. Using `hive.metastore.uris` it is possible to specify multiple remote metastores. Hive will use the first one from the list by default but will pick a random one on connection failure and will try to reconnect.

#### Additional Configuration Parameters

The following metastore configuration parameters were carried over from old documentation without a guarantee that they all still exist. See the `HiveConf` Java class for current Hive configuration options, and see the Metastore and Hive Metastore Security sections of the Language Manual's Hive Configuration Properties for user-friendly descriptions of the metastore parameters.

| Configuration Parameter | Description | Default Value |
|---|---|---|
| hive.metastore.metadb.dir | The location of filestore metadata base directory. (Functionality removed in 0.4.0 with HIVE-143.) | |
| hive.metastore.rawstore.impl | Name of the class that implements the org.apache.hadoop.hive.metastore.rawstore interface. This class is used to store and retrieval of raw metadata objects such as table, database. (Hive 0.8.1 and later.) | |
| org.jpox.autoCreateSchema | Creates necessary schema on startup if one doesn't exist. (The schema includes tables, columns, and so on.) Set to false after creating it once. | |
| org.jpox.fixedDatastore | Whether the datastore schema is fixed. | |
| datanucleus.autoStartMechanism | Whether to initialize on startup. | |
| hive.metastore.ds.connection.url.hook | Name of the hook to use for retriving the JDO connection URL. If empty, the value in javax.jdo.option.ConnectionURL is used as the connection URL. (Hive 0.6 and later.) | |
| hive.metastore.ds.retry.attempts | The number of times to retry a call to the backing datastore if there were a connection error. (Hive 0.6 through 0.12; removed in 0.13.0 – use hive.hmshandler.retry.attempts instead.) | 1 |
| hive.metastore.ds.retry.interval | The number of miliseconds between datastore retry attempts. (Hive 0.6 through 0.12; removed in 0.13.0 – use hive.hmshandler.retry.interval instead.) | 1000 |
| hive.metastore.server.min.threads | Minimum number of worker threads in the Thrift server's pool. (Hive 0.6 and later.) | 200 |
| hive.metastore.server.max.threads | Maximum number of worker threads in the Thrift server's pool. (Hive 0.6 and later.) | 100000 since Hive 0.8.1 |
| hive.metastore.filter.hook | Metastore hook class for further filtering the metadata read results on client side. (Hive 1.1.0 and later.) | org.apache.hadoop.hive.metastore.DefaultMetaStoreFilterHookImpl |
| hive.metastore.port | Hive metastore listener port. (Hive 1.3.0 and later.) | 9083 |

### Data Nucleus Auto Start

Configuring datanucleus.autoStartMechanism is highly recommended

Configuring auto start for data nucleus is highly recommended. See HIVE-4762 for more details.

```
<property>
   <name>datanucleus.autoStartMechanism</name>
   <value>SchemaTable</value>
 </property>
```

### Default Configuration

The default configuration sets up an embedded metastore which is used in unit tests and is described in the next section. More practical options are described in the subsequent sections.

## Local/Embedded Metastore Database (Derby)

**An embedded metastore database is mainly used for unit tests. Only one process can connect to the metastore database at a time, so it is not really a practical solution but works well for unit tests.**

For unit tests Local/Embedded Metastore Server configuration for the metastore server is used in conjunction with embedded database.

Derby is the default database for the embedded metastore.

| Config Param | Config Value | Comment |
|---|---|---|

| | | |
|---|---|---|
| javax.jdo.option.ConnectionURL | `jdbc:derby:;databaseName=../build/test/junit_metastore_db;create=true` | Derby database located at hive/trunk/build... |
| javax.jdo.option.ConnectionDriverName | `org.apache.derby.jdbc.EmbeddedDriver` | Derby embeded JDBC driver class. |
| hive.metastore.warehouse.dir | `file://${user.dir}/../build/ql/test/data/warehouse` | Unit test data goes in here on your local filesystem. |

If you want to run Derby as a network server so the metastore can be accessed from multiple nodes, see Hive Using Derby in Server Mode.

## Remote Metastore Database

In this configuration, you would use a traditional standalone RDBMS server. The following example configuration will set up a metastore in a MySQL server. This configuration of metastore database is recommended for any real use.

| Config Param | Config Value | Comment |
|---|---|---|
| javax.jdo.option.ConnectionURL | `jdbc:mysql://<host name>/<database name>?createDatabaseIfNotExist=true` | metadata is stored in a MySQL server |
| javax.jdo.option.ConnectionDriverName | `com.mysql.jdbc.Driver` | MySQL JDBC driver class |
| javax.jdo.option.ConnectionUserName | `<user name>` | user name for connecting to MySQL server |
| javax.jdo.option.ConnectionPassword | `<password>` | password for connecting to MySQL server |

## Local/Embedded Metastore Server

In local/embedded metastore setup, the metastore server component is used like a library within the Hive Client. Each Hive Client will open a connection to the database and make SQL queries against it. Make sure that the database is accessible from the machines where Hive queries are executed since this is a local store. Also make sure the JDBC client library is in the classpath of Hive Client. This configuration is often used with HiveServer2 (to use embedded metastore only with HiveServer2 add "--hiveconf hive.metastore.uris=' '" in command line parameters of the hiveserver2 start command or use hiveserver2-site.xml (available in Hive 0.14)).

| Config Param | Config Value | Comment |
|---|---|---|
| hive.metastore.uris | *not needed because this is local store* | |
| hive.metastore.local | `true` | this is local store (removed in Hive 0.10, see configuration description section) |
| hive.metastore.warehouse.dir | `<base hdfs path>` | Points to default location of non-external Hive tables in HDFS. |

## Remote Metastore Server

In remote metastore setup, all Hive Clients will make a connection to a metastore server which in turn queries the datastore (MySQL in this example) for metadata. Metastore server and client communicate using Thrift Protocol. Starting with Hive 0.5.0, you can start a Thrift server by executing the following command:

```
hive --service metastore
```

In versions of Hive earlier than 0.5.0, it's instead necessary to run the Thrift server via direct execution of Java:

```
$JAVA_HOME/bin/java  -Xmx1024m -Dlog4j.configuration=file://$HIVE_HOME/conf/hms-log4j.properties -Djava.library.path=$HADOOP_HOME/lib/native/Linux-amd64-64/ -cp $CLASSPATH org.apache.hadoop.hive.metastore.HiveMetaStore
```

If you execute Java directly, then JAVA_HOME, HIVE_HOME, HADOOP_HOME must be correctly set; CLASSPATH should contain Hadoop, Hive (lib and auxlib), and Java jars.

**Server Configuration Parameters**

The following example uses a Remote Metastore Database.

| Config Param | Config Value | Comment |
|---|---|---|
| javax.jdo.option.ConnectionURL | `jdbc:mysql://<host name>/<database name>?createDatabaseIfNotExist=true` | metadata is stored in a MySQL server |
| javax.jdo.option.ConnectionDriverName | `com.mysql.jdbc.Driver` | MySQL JDBC driver class |
| javax.jdo.option.ConnectionUserName | `<user name>` | user name for connecting to MySQL server |
| javax.jdo.option.ConnectionPassword | `<password>` | password for connecting to MySQL server |
| hive.metastore.warehouse.dir | `<base hdfs path>` | default location for Hive tables. |
| hive.metastore.thrift.bind.host | <host_name> | Host name to bind the metastore service to. When empty, "localhost" is used. This configuration is available Hive 4.0.0 onwards. |

From Hive 3.0.0 (HIVE-16452) onwards the metastore database stores a GUID which can be queried using the Thrift API get_metastore_db_uuid by metastore clients in order to identify the backend database instance. This API can be accessed by the HiveMetaStoreClient using the method getMetastoreDbUuid().

### Client Configuration Parameters

| Config Param | Config Value | Comment |
|---|---|---|
| hive.metastore.uris | `thrift://<host_name>:<port>` | host and port for the Thrift metastore server. If hive.metastore.thrift.bind.host is specified, host should be same as that configuration. Read more about this in dynamic service discovery configuration parameters. |
| hive.metastore.local | `false` | Metastore is remote.  Note: This is no longer needed as of Hive 0.10.  Setting *hive.metastore.uri* is sufficient. |
| hive.metastore.warehouse.dir | `<base hdfs path>` | Points to default location of non-external Hive tables in HDFS. |

#### Dynamic Service Discovery Configuration Parameters

From Hive 4.0.0 (HIVE-20794) onwards, similar to HiveServer2, a ZooKeeper service can be used for dynamic service discovery of a remote metastore server. Following parameters are used by both metastore server and client.

| Config Param | Config Value | Comment |
|---|---|---|
| hive.metastore.service.discovery.mode | service discovery mode | When it is set to "zookeeper", ZooKeeper is used for dynamic service discovery of a remote metastore. In that case, a metastore adds itself to the ZooKeeper when it is started and removes itself when it shuts down. By default it is empty. Both the client and server should have same value for this parameter. |
| hive.metastore.uris | <host_name>:<port>, <host_name>:<port>, ... | One or more host and port pairs of ZooKeeper servers forming a ZooKeeper ensemble. Used when hive.metastore.service.discovery.mode is set to "zookeeper". The configuration is not used by server otherwise. If all the servers are using the same port you may specify the port using hive.metastore.zookeeper.client.port instead of specifying it with every server separately. Both the client and server should have same value for this parameter. |
| hive.metastore.zookeeper.client.port | <port> | Port number when same port number is used by all the ZooKeeper servers in the ensemble. Both the client and server should have same value for this parameter. |
| hive.metastore.zookeeper.namespace | <namespace name> | The parent node under which all ZooKeeper nodes for metastores are created. |

| | | |
|---|---|---|
| hive.metastore.zookeeper.session.timeout | <time in milliseconds> | ZooKeeper client's session timeout (in milliseconds). The client is disconnected if a heartbeat is not sent in the timeout. |
| hive.metastore.zookeeper.connection.timeout | <time in seconds> | ZooKeeper client's connection timeout in seconds. Connection timeout * hive.metastore.zookeeper.connection.max.retries with exponential backoff is when curator client deems connection is lost to zookeeper. |
| hive.metastore.zookeeper.connection.max.retries | <number> | Max number of times to retry when connecting to the ZooKeeper server. |
| hive.metastore.zookeeper.connection.basesleeptime | <time in milliseconds> | Initial amount of time (in milliseconds) to wait between retries when connecting to the ZooKeeper server when using ExponentialBackoffRetry policy. |

If you are using MySQL as the datastore for metadata, put MySQL jdbc libraries in HIVE_HOME/lib before starting Hive Client or HiveMetastore Server.

To change the metastore port, use this `hive` command:

```
hive --service metastore -p <port_num>
```

## Supported Backend Databases for Metastore

| Database | Minimum Supported Version | Name for Parameter Values | See Also |
|---|---|---|---|
| MySQL | 5.6.17 | `mysql` | |
| Postgres | 9.1.13 | `postgres` | |
| Oracle | 11g | `oracle` | hive.metastore.orm.retrieveMapNullsAsEmptyStrings |
| MS SQL Server | 2008 R2 | `mssql` | |

## Metastore Schema Consistency and Upgrades

Version

Introduced in Hive 0.12.0. See HIVE-3764.

Hive now records the schema version in the metastore database and verifies that the metastore schema version is compatible with Hive binaries that are going to accesss the metastore. Note that the Hive properties to implicitly create or alter the existing schema are disabled by default. Hive will not attempt to change the metastore schema implicitly. When you execute a Hive query against an old schema, it will fail to access the metastore.

To suppress the schema check and allow the metastore to implicitly modify the schema, you need to set a configuration property `hive.metastore.schema.verification` to false in `hive-site.xml`.

Starting in release 0.12, Hive also includes an off-line schema tool to initialize and upgrade the metastore schema. Please refer to the details here.