CLI Commands

Database Credentials

The command descriptions assume that database credentials are stored in the project (the default) or in environment variables.

If you have `runtimeCredentials` set and the credentials are not stored in environment variables, then you need to specify the database credentials on the command line.

We strongly recommend to use runtime credentials because it is more secure than stored credentials (due to the possibility of the stored password being decoded).

See Database Credential Management.

Single-credential

> Commands that work on a single database (dbdef) require the `-un <username>` and -pw `<password>` options to specify the database credentials.

Two-credential

> Commands that work on two databases require credentials for the reference (desired state) and target (changed) databases

- Reference database: Use `-ru <username>` and `-rpw <password>` options.
- Target database: Use `-tu <username>` and `-tpw <password>` options.

Getting Help

To get a list of available commands, run `hammer help` at a system command prompt.

```
Datical DB CLI Help
```

---

add-to-property

Add a path to a property

USAGE

```
add-to-property <property> <path>
```

EXAMPLE

```
add-to-property drivers /home/user/newdrivers
```

NOTES

This command appends a new path to an existing property that expects path values.

---

addContexts

Add contexts to changesets. Adds context attributes to a set of changesets in the changelog. The attributes are selected using a label expression. This command is used by the Deployment Packager to indicate that a group of changesets is ready to be deployed to a new environment.

> **Note**
> Starting with v4.32, please use the `modifyChangeSet` command. The `addContexts` command is deprecated and support may be removed in the future.

USAGE

```
addContexts --lookupChangesets=<label-expression> --context=<contexts> [--scriptChecksum=<checksum>] [
--matchAll=true|false]
```

OPTIONS

- `lookupChangesets` - Label expression used to matchthe change sets (required)
- `context` - Comma-separated list of contexts (required)
- `scriptChucksum` - SQL script checksum value
- `matchAll` - SQL Script checksums must match for change set lookup. Default value is true.

NOTES

The value of the `scriptChecksum` parameter can be used to further qualify change sets that match the label expression. This is used primarily in the Deployment Packager to confirm that the change set was generated by the expected version of a script.

The value of the `matchAll` parameter impacts the matching behavior of the `scriptChecksum` parameter.

- If `matchAll` is true, all change sets identified by the label expression must have a `scriptChecksum` attribute that matches the attribute passed on the command line.
- If `matchAll` is false, only one change set returned by the label expression needs to have a matching `scriptChecksum` attribute.

addLabels

Add labels to changesets. Adds label attributes to a set of changesets in the changelog that are selected using a label expression.

> **Note**
> Starting with v4.32, please use the `modifyChangeSet` command. The `addLabels` command is deprecated and support may be removed in the future.

USAGE

```
addLabels --lookupChangesets=<label-expression> --context=<contexts> [--scriptChecksum=<checksum>] [--matchAll=true|false]
```

OPTIONS

- `lookupChangesets` - Label expression used to match the change sets (required)
- `labels` - Comma-separated list of labels (required)
- `scriptChecksum` - SQL script checksum value
- `matchAll` - SQL Script checksums must match for change set lookup. Default value is true

NOTES

The value of the `scriptChecksum` parameter can be used to further qualify change sets that match the label expression. This is used primarily in the Deployment Packager to confirm that the change set was generated by the expected version of a script.

The value of the `matchAll` parameter impacts the matching behavior of the `scriptChecksum` parameter.

- If `matchAll` is true, all change sets identified by the label expression must have a `scriptChecksum` attribute that matches the attribute passed on the command line.
- If `matchAll` is false, only one change set returned by the label expression needs to have a matching `scriptChecksum` attribute.

---

assignMetadata

Assign metadata information to a label. Used to associate more information with a label than could be expressed in the label itself. The metadata is stored in the project's associated DMC database, so this command requires that an DMC database has been configured for the project.

USAGE

```
assignMetadata --metdataLabel=<label name> --metadataFile=<path to file>
```

OPTIONS

- `metadataLabel` - Name of the label
- `metadataFile` - Path to a properties file containing the metadata key-value pairs

---

changelogSync

Mark all change sets as executed in the database. Marks all changesets as applied to the specified database.

USAGE

```
changelogSync <dbdef> <options>
```

EXAMPLE

```
changelogSync myAppDev --mergeLabels=merge --pipeline=Feature1
```

OPTIONS

- `labels - Comma-separated list of labels`

- `mergeLabels` - Label merge strategy, one of override (default), merge (AND), append (OR). Specifies how to combine labels set on the command line with labels set on the step.
- `pipeline` - Specifies the pipeline associated with the changelogSync for reporting purposes.
    - If not set and the dbDef is a member of only one pipeline, then that pipeline is used.
    - If not set and the dbDef is a member of multiple pipelines, then an error is returned.

NOTES

USE CAUTION! Before running changelogSync, verify that the specified database is up to date!

Running `changelogSync` against a database that is not complete can lead to future deployment failures.

---

checkDrivers

Check that JDBC driver jars are installed or on the specified path. Scans the path(s) specified by the 'drivers' property for accessible JDBC drivers.

USAGE

```
checkDrivers
```

EXAMPLE OUTPUT

```
Found installed driver: com.ibm.db2.jcc.DB2Driver driver: (v3.63) jar: (v1.0.11.20181126111708)
Found installed driver: com.datical.jdbc.oracle.DaticalOracleDriver jar: (v0.0.60.20181126114644)
Found installed driver: com.microsoft.sqlserver.jdbc.SQLServerDriver driver: (v6.3) jar: (v1.0.11)
Found installed driver: oracle.jdbc.OracleDriver driver: (v18.3) jar: (v1.0.11)
```

NOTES

There are several ways to configure Datical DB to use the JDBC drivers that communicate with the databases being managed. The preferred method is to install the drivers into the OSGi environment using the `installDrivers` command. Drivers can also be found by creating a properties file in the project and then either

- manually editing the file
- using the command `set property drivers <path-to-drivers>`

For more information on setting the drivers property, see the help: `help set property`

> **Note**
> For backward compatibility, `checkdrivers` (all lowercase) also works.

---

checkRules

Compiles rules and reports any errors.

USAGE

```
checkRules [PreForecast|Forecast|PostForecast|SqlRules]
```

EXAMPLE

```
checkRules PreForecast
```

OPTIONS

- `rulesFolder` - specifies the folder to check, one of PreForecast, Forecast, PostForecast, SqlRules. These are the standard folders under the Rules folder for the project. If rulesFolder is not specified, then rules in all the standard folders are compiled.

NOTES

Run the command from a valid Datical DB project directory.

---

clearCheckSums

Delete the checksums used to detect altered change sets from the tracking tables.

USAGE

clearCheckSums <dbdef>

EXAMPLE

clearCheckSums myAppDevDB

OPTIONS

NOTES

Use only under the direction of a Datical Support Engineer or if you are very experienced with Datical DB.

This command is helpful when you need to respond to a change to an existing object that is made outside of Datical DB. For example, someone uses a SQL script to change a database directly. It allows you to update the change set that creates this object and clear the existing checksums to avoid validation errors. The checksums are recreated on the next `deploy` or `changeLogSync` command.

---

create

Create various project artifacts.

USAGE

create <object> <options>

OBJECTS

- DbDef <json_file> - create a new DbDef in the current project using <json_file>, a file containing JSON key-value pairs.

```
{
"DbDefClass":"OracleDbDef",
"name":"MyNewOracleDB",
"hostname":"127.0.0.1",
"port":"1521",
"username":"orclUser",
"password":"hammer01@",
"sid":"ORCLSID",
"contexts":"DEV,QA"
}
```

Keys

- `DbDefClass` - Database type (OracleDbDef | Db2DbDef | MysqlDbDef | SqlServerDbDef | PostgresqlDbDef)
- `name` - Name for the dbDef. Used as an alias in the command interpreter and the UI.
- `hostname` - Host name or IP address of the target database server
- `port` - Port number for JDBC connections to the target database server
- `username` - Database user name to use for the connection
- `password` - Password for the database user
- `database` - (for MySQL, DB2, Postgres) The name of the database you wish to manage.
- `databaseName` - (for SQLServer) The name of the database you wish to manage.
- `instanceName` - (for SQLServer) The name of the instance you wish to manage.

- `applicationName` - (for SQLServer) The name of the database application you wish to manage.
- `sid` - (for Oracle) - Oracle SID for the connection
- `serviceName` - (for Oracle) - Oracle Service for the connection
- `defaultSchemaName` - Name of the schema/catalog to manage.
- `contexts` - Comma-separated list of contexts for the DbDef.

---

dbshow

List all databases for a project, along with extra info about each.

USAGE

```
dbshow
```

```
dbshow <dbdef>...
```

EXAMPLE

```
dbshow DEVdb QAdb STAGEdb
```

NOTES

If you use the command without specifying a database, information for all dbdefs is shown.

To obtain information on a subset of database definitions, use their names as command line parameters.

OUTPUT

Lists the name, URL, database name, alternate schema, and default context.

---

debug export

Export and scrub sensitive data from files to be sent to Datical support for debugging or troubleshooting.

USAGE

```
debug export [--scrub=true|false] [--include=<searchstring>[,<searchstring>,...]] [--
exclude=<searchstring>[,<searchstring>,...]] [--lastmodified=<N>] [--report=<path/file>]
```

EXAMPLE

```
debug export
```

OPTIONS

- `--scrub=true|false` - Default is true. When set, scrub will replace sensitive information (such as usernames, hostnames, ports) with generic strings (tokens).
- `--include=<searchstring>[,<searchstring>,...]` - include only files specified by one or more search strings (filters).
- `--exclude=<searchstring>[,<searchstring>,...]` - include all files except those specified by one or more search strings (filters).
- `--lastmodified=<N>` - include only files modified within the last <N> hours.Specifies the pipeline associated with the changelogSync for reporting purposes.
- `--report=<path/file>` - put the resulting zip file in a specified file. By default it is placed in `<project>/Reports/debug/<project>.zip`

NOTES

Run the command at the root of a project directory, `<project>`.

For use case scenarios and examples, see Assembling Data for Datical Support.

To include an external file, place it in the project directory, preferably in the root, `<project>`.

Do not use : or / characters in `<searchstring>`. If you search for a URL, do not include the prefix (`http://, https://`)

OUTPUT

Creates a zip file of the specified project files. The zip file is placed in `<project>/Reports/debug/<project>.zip` by default.

---

deploy

Deploy the changelog, updating the target database schema to the latest version. Deploys all change sets that have not been applied to the indicated database in the project.

USAGE

```
deploy <dbdef> [–log=] [--enableRowCount=exact|approximate|disabled] [--pipeline=<name>] [--
report=<path>] [--context=context1,context2] [--labels=<label expression>][--
mergeLabels=override|merge|append][--immutableProject=<true|false>][--invalidsAction=<warn|fail>]
```

EXAMPLE

```
deploy myAppDevDb --log=/home/user/dbadmin/daticallogs/ --report=/home/user/dbadmin/daticalreports --
context=dev,integration --labels="JUN AND (poolApp or beachApp)" --invalidsAction=fail
```

DATICAL SERVICE OPTIONS

Use these options to access a project that is stored on Datical Service.

- `--daticalServer=<cluster-name>` - The cluster name set for Datical Service. The cluster name is set during Datical Service installation.
- `--daticalUsername=<user>` - User name, must be configured in Datical Service. Use the DATICAL_PASSWORD environment variable to provide the password.
- `--projectKey=<projectRef>` - Project to use, specified `<projectRef>`, a project name or project key. The project key is defined in Datical Service only.

OPTIONS

- `log` - When set, the `daticaldb.log` file will be written to the directory specified. Log is written to `Logs` directory of project by default.
- `enableRowCount - exact | approximate | disabled`, method for counting rows in tables. Default is disabled.
  - Before v5.7 used `true | false`, with default as true. New default is `disabled`. Older projects are interpreted correctly and mapped to the new values.
- `pipeline` - Specifies the pipeline associated with the changelogSync for reporting purposes.
  - If not set and the dbDef is a member of only one pipeline, then that pipeline is used.
  - If not set and the dbDef is a member of multiple pipelines, then an error is returned.
- `report` - When set, deploy report will be written to the directory specified. Report is written to the `Reports` directory by default.
- `context` - When set, only change sets marked with the contexts specified will be executed. To run all contexts, specify `$all`.
- `labels` - When set, only change sets marked with the label expression will be executed. To run all labels, specify `$all`.
- `mergeLabels` - Label merge strategy, one of override (default), merge (AND), append (OR). Specifies how to combine labels set on the command line with labels set on the step.
- `deployMode` - Override the `deployMode` project setting, either `full` (default), which includes forecasting and rules checking, or `quick`.
- `limitForecast` - Override the `limitForecast` project setting. Set to either `false` (default), where forecasting is done for all objects in the target database, or `true`, which limits forecasting to objects directly affected by the deployment.
- `immutableProject` - When running with Datical Service options, set to `true` to use the local project files rather than download the project information from Datical Service. Helpful in automated environments where project files are kept in artifact repositories.
- `invalidsAction` - Override the storedLogicValidityAction project setting for this deploy.
  - `warn`: Set to either warn (default), current behavior. New invalid stored logic will cause the Deploy to flag as deployment warning.
  - `fail`: New invalid stored logic will cause a deployment to be marked as a failure.
    Option would be applied when Stored Logic Validity Check is not disabled.

---

diff

Compare two databases and produce a report. Compares two database definitions in a project and produces a text report of the differences.

USAGE

```
diff <referenceDB> <targetDb> [--output=/path/to/report]
```

EXAMPLE

```
diff myAppDevDB myAppQaDB --output=<path>
```

OPTIONS

- `referenceDB` - database that represents the desired state
- `targetDB` - database that is out of synchronization and needs to be caught up to the state of `referenceDB`.
- `output` - If present, specifies a file for the report. If not specified, the report is sent to STDOUT

---

diffChangelog

Compares two database definitions in a project and produces the change sets necessary to synchronize the databases.

USAGE

```
diffChangelog <referenceDB> <targetDb> [--output=<path>] [--assignLabels=<labels>] [--
assignContexts=<contexts>] [–-scriptChecksum=<checksum>]
```

EXAMPLE

```
myAppDevDB myAppQaDB --output=/home/user/diffChangeLogsdir/diffChangeLog.xml --assignLabels=label1,
label2 --assignContexts="(contextOne and contextTwo)"
```

OPTIONS

- `referenceDB` - database that represents the desired state, either a dbdef in the project or a snapshot file.
- `targetDB` - database that is out of synchronization and needs to be caught up to the state of `referenceDB`, either a dbdef in the project or a snapshot file
- `output` - if present, specifies the file for the generated changelog. Use file extension .xml. If not specified, output is written to the `Reports/DiffChangeLogs` subdirectory under the project directory. If the file exists, the results of the command (changesets) are appended to the existing file.
- `assignLabels` - apply this set of labels applied to all changesets created. Accepts a comma separated list of labels.
- `assignContexts` - apply this set of contexts to to all changesets created. Accepts a complex context expression.
- `scriptChecksum` - SQL script checksum value

---

forceSync

The forceSync forces a re-sync/re-base to DMC DB using the files currently active in the workspace available to hammer. Note that some warning messages refer to this forcesync command as "the resync command in the CLI".

USAGE

```
forceSync
```

EXAMPLE

```
forceSync
```

---

forecast

Simulate unexecuted change sets and run rules against the specified database.

USAGE

```
forecast <dbdef> [--log=/path/to/log]  [--enableRowCount=exact|approximate|disabled] [--report=/path
/to/report] [--context=context1,context2] [--labels=<label expression>][--
mergeLabels=override|merge|append][--limitForecast=true|false][--immutableProject=<true|false]
```

EXAMPLE

```
forecast myAppDevDb --log=/home/user/logs/ --report=/home/user/reports --context=dev,qa --labels="JUN
AND (poolApp or beachApp)" --limitForecast=true
```

DATICAL SERVICE OPTIONS

Use these options to access a project that is stored on Datical Service.

- `--daticalServer=<cluster-name>` - The cluster name set for Datical Service. The cluster name is set during Datical Service installation.
- `--daticalUsername=<user>` - User name, must be configured in Datical Service. Use the DATICAL_PASSWORD environment variable to provide the password.
- `--projectKey=<projectRef>` - Project to use, specified `<projectRef>`, a project name or project key. The project key is defined in Datical Service only.

OPTIONS

- `<dbdef>` - project dbdef name of the database to use for forecasting
- `log` - Location to write the daticaldb.log file. Log is written to `Logs` directory of project by default.
- `enableRowCount - exact | approximate | disabled`, method for counting rows in tables. Default is disabled.
    - Before v5.7 used `true | false`, with default as true. New default is `disabled`. Older projects are interpreted correctly and mapped to the new values.
- `report` - Location to write the forecast report. Report is written to the `Reports` directory by default.
- `context` - Only change sets marked with the contexts specified are executed. To run all contexts, specify `$all`.
- `labels` - When set, only change sets marked with the label expression are executed. To run all labels, specify `$all`.
- `mergeLabels` - Label merge strategy, one of override (default), merge (AND), append (OR). Specifies how to combine labels set on the command line with labels set on the step.
- `limitForecast` - Override the `limitForecast` project setting. Set to either `false` (default), where forecasting is done for all objects in the target database, or `true`, which limits forecasting to objects directly affected by the potential deployment.
- `immutableProject` - When running with Datical Service options, set to `true` to use the local project files rather than download the project information from Datical Service. Helpful in automated environments where project files are kept in artifact repositories.

---

groovy

Run a Groovy script. Runs a script of Datical DB commands using the Groovy script engine.

USAGE

```
groovy <script-name> <script-options>
```

EXAMPLE

```
groovy myScript.groovy show(project, ["dbprop", "myProject", "enableStorageOptions"])
```

OPTIONS

- `<script-name>` - path to a script name
- `<script-options>` - Options to pass to the script

NOTES

To access Datical DB commands through the script, access them through the DaticalDb object. The parameters for a command are `(project, List<String> <[parameters]>)`. Use the word "project" as shown below.

Exceptions: `add-to-property` is accessed by `addToProperty` and `deploy-autoRollback` is accessed by `deploy-AutoRollback`.

Examples:

```
help(project, [])

help(project, ["groovy"])

show(project, ["dbprop", "myProject", "enableStorageOptions"])
```

---

help

Show help on Datical DB commands.

USAGE

```
help
```

```
help <command>|options
```

```
set help
```

```
show help
```

EXAMPLE

```
help diff
```

OPTIONS

- `<command>` - Name of the command you'd like to learn about.
- `options` - list of options used by the commands

NOTES

When run without a command option, lists all available commands and a brief description of each.

SEE ALSO

The `set` and `show` commands.

---

history

Show the applied changeset history of a database. Shows all changesets that have been applied to the specified DbDef.

USAGE

```
history <dbdef>
```

EXAMPLE

```
history QAdb
```

OPTIONS

- `<dbdef>` - project dbdef name of the database to use.

---

insertSavedAuditData (deprecated)

Insert all audit entries temporarily saved in the project into the user-configured DMC DB.

USAGE

```
insertSavedAuditData --project=<path>
insertSavedAuditData
```

OPTIONS

- `project` - Path to the project directory to use for inserting saved DMC DB data.

NOTES

If run without an option, the command runs in the current directory. The command fails if the directory is not a valid project directory.

Sometimes an operation (like deploy) encounters a condition that prevents information from being written to the DMC database. In that case, the information is saved to a temporary on-disk database. Use this command to move the on-disk data to the configured DMC database once connectivity to the DMC database has been restored.

---

installDrivers

Run the custom script `installDrivers.groovy`. Installs OSGi packaged JDBC drivers from the specified update site

USAGE

```
hammer installDrivers <driver_update_site>
```

EXAMPLE

```
hammer installDrivers http://update.datical.com/latest/thirdparty
```

NOTES

Update sites can be either HTTP URLs (as shown above) or directories on a file system.

Examples of `<driver_update_site>`:

- **Composite Archive** (Windows and Linux):
  `jar:file:/C:/Users/Pete/DaticalDBCompositeRepo-4.26.4467.zip!/`
  `jar:file:/home/Users/Pete/DaticalDBCompositeRepo-4.26.4467.zip\!/`
- **Extracted Archive** (Windows and Linux):
  `file:/C:/Users/Pete/DaticalDBCompositeRepo-4.26.4467/`
  `file:/home/Pete/DaticalDBCompositeRepo-4.26.4467/`

---

installLicense

Install a license file to allow continued use of Datical DB. Once a license is installed on a machine once, it does not typically need to be installed again.

USAGE

```
installLicense <path>
```

EXAMPLE

```
installLicense /home/datical/datdb.lic
```

OPTIONS

- `<path>` - Specifies where to find the file that contains the Datical DB product license.

NOTES

If you need to obtain or renew a license please contact our Technical Support team.

---

licenseCounter

Count the DbDefs in use for licensing validation.

USAGE

```
hammer licenseCounter <projects_dir> [ showDebug=true ]
```

EXAMPLE

Linux: `hammer licenseCounter ~/datical`
Windows: `hammer.bat licenseCounter C:\Datical\Projects`

OPTIONS

- `<projects_dir>` - Specifies the top directory to begin scanning project files. It can be a relative or absolute path. All project files in the file hierarchy under this directory are scanned.
- `showDebug=true` - Show additional detailed information as the command runs.

NOTES

Output is sent to `stdout`. Redirect to a file to save it.

modifyChangeSet

Modify changeset attributes

USAGE

```
modifyChangeSet --action=<action> --<modify-option>[=<value>, ...]... [--<search-option>=<value>,
...]...
```

EXAMPLE

```
hammer modifyChangeSet --action=ADD --modifyLabels=history --searchIgnore=true

hammer modifyChangeSet --action=ADD --modifyLabels=abandoned --searchLabels=MyProcedureName123.sql

hammer modifyChangeSet --action=REMOVE --modifyContexts=QA4 --searchContexts=QA4

hammer modifyChangeSet --action=ADDALL --modifyContexts=DEV,PATCH01

hammer modifyChangeSet --action=REMOVEALL --modifyContexts=PATCH01

hammer modifyChangeSet --action=ADD --modifyIgnore --searchLabels=MyScriptName123.sql

hammer modifyChangeSet --action=ADD --modifyContexts=DEV,NEWCONTEXT --modifyLabels=MyTestSsisPackage
--searchOrigFilePaths=ssis/MyTestSsisPackage
```

OPTIONS

Action

- `action` - Specifies the action to perform, one of ADD, REMOVE, ADDALL, REMOVEALL
  - All actions require at least one modify option.
  - If you specify ADD or REMOVE, you must also specify a search option.
  - If you specify ADDALL or REMOVEALL, do not specify a search option. If you do, the command fails with an error.

Modify Option

At least one modify option is required. It specifies *what change to make*. It can be one of the following:

- `modifyContexts` - One or a comma-separated list of values for the `context` attribute.
- `modifyIgnore` - Change the value of the `ignore` attribute. It does not take a value. The ADD action sets `ignore=true`. The REMOVE action sets `ignore=false`.
- `modifyLabels` - One or a comma-separated list of values for the `label` attribute.

When a value is added to an attribute, it is added at the end of the existing list and the order of the list is preserved. If the value already exists, it is not added again.

When a value is removed from an attribute, the order of the remaining values is preserved.

When you specify more than one modify option, all modifications of the same type (context, ID, label) are put together and applied at one time.

Search Option

The search options determine *which changesets to modify*. They select a list of changesets for the modify options to work on.

Searches are not case-sensitive.

- Multiple values in one statement are OR'ed together.
- Multiple search option statements are ANDed together.

Search option statements take one or a comma-separated list of values. They are provided for changeset attributes as follows:

- `searchIds` - `id`
- `searchLabels` - `label`
- `searchContexts` - `context`
- `searchIgnore` - `ignore` - `true` | `false` - Searches for false also return changesets where the ignore attribute is not set. `true` | `false`
- `searchOrigFilePaths` - `origFilePath`
- `searchOrigFileNames` - `origFileName`

- `searchVersion` - `version`, an integer.

OUTPUT

The command reports the number of changesets that met the search criteria. If no changesets meet the criteria, the command succeeds and reports that no changesets met the search criteria.

ERRORS

The command fails and reports an error when:

- No action is specified or an invalid value is specified.
- An action is valid but no modify options are specified.
- ADDALL or REMOVEALL is specified as an action and a search option is specified.
- ADD or REMOVE is specified as an action and a search option is not specified.
- An invalid value is specified for `searchIgnore`. It must be `true` or `false`.

NOTES

The `modifyChangeSet` command is intended to make it easier to work with labels and contexts.

- Searching - Make it easier to find a changeset: Search by criteria other than a label name, particularly criteria that can be unique (origFilePath, origFileName, ID).
- Labels - Manage labels for a release, especially for later steps in a pipeline.
- Contexts - Use context settings as a gating mechanism for deploying to a specified environment (context)

Context expressions and label expressions are not currently supported. Use comma-separated lists for multiple values.

Starting with version 4.32, Use this command rather than the following commnds, which are deprecated and may be removed in a future release:

- `addLabels`
- `removeLabels`
- `addContexts`
- `removeContexts`

---

newDbDef

Creates a new dbDef based on the key/value pairs specified as arguments.

- Oracle
  - Inline Credentials: **OracleDbDef**
  - Run-Time Supplied Credentials: **DelayedCredentialOracleDbDef**
- SQL Server
  - Inline Credentials: **SqlServerDbDef**
  - Run-Time Supplied Credentials: **DelayedCredentialSqlServerDbDef**
- Postgres Enterprise DB
  - Inline Credentials: **PostgresqlDbDef**
  - Run-Time Supplied Credentials: **DelayedCredentialPostgresDbDef**
- DB2
  - Inline Credentials: **Db2DbDef**
  - Run-Time Supplied Credentials: **DelayedCredentialDb2DbDef**
- (deprecated) MySQL Inline Credentials: MysqlDbDef

USAGE

newDbDef DbDefClass <value> name <value> hostname <value> port <value> username <value> password <value> database <value> ... <key> <value>

AVAILABLE KEYS

- DbDefClass - Database Type. (OracleDbDef|Db2DbDef|MysqlDbDef|SqlServerDbDef|PostgresqlDbDef|DelayedCredentialOracleDbDef|DelayedCredentialSqlServerDbDef|DelayedCredentialPostgresDbDef|DelayedCredentialDb2DbDef)
- name - The name for the dbDef. Used in the repl and UI as an alias for this connection.
- hostname - The hostname/ip of the target database server.

- port - The port number for JDBC connections to the target database server.
- username - The database user name to use for the connection.
- password - The password for the database user specified.
- database - (only applies to MySQL|DB2|Postgres) The name of the database you wish to manage.
- databaseName - (only applies to SQLServer) The name of the database you wish to manage.
- instanceName - (only applies to SQLServer) The name of the instance you wish to manage.
- dbType - (only applies to SQLServer) The type of SQLServer.
    - Available values: mssql (default), az_sqlmi, az_sqldb.
- authentication - (only applies to SQLServer) The type of authentication for Azure Databases.
    - Available values: none (default), activeDirectoryIntegrated, activeDirectoryPassword, activeDirectoryMSI.
- msiClientId - (only applies to SQLServer) Managed identity for Azure resources.
- applicationName - (only applies to SQLServer) The name of the database application you wish to manage.
- azureClientId - (Azure SQL MI RefDB Only) Application (client) id of service principles.
- azureTenantId - (Azure SQL MI RefDB Only) Id of tenant on Azure.
- azureClientSecret - (Azure SQL MI RefDB Only) Secret key of service principles on Azure.
- azureResourceGroup - (Azure SQL MI RefDB Only) Name of resource group on Azure
- azureSubscriptionId - (Azure SQL MI RefDB Only) Id of subscription on Azure.
- azureRestAuthenticationOption - (Azure SQL MI RefDB Only) Azure REST API authentication type. Available Values: certificate, clientSecret (default)
- sid - (only applies to Oracle) The name of the Oracle SID to which you wish to connect.
- serviceName - (only applies to Oracle) The name of the Oracle Service to which you wish to connect.
- defaultSchemaName - The name of the schema you wish to manage.
- defaultCatalogName - The name of the catalog you wish to manage.
- contexts - A comma separated list of contexts to associate with the new dbDef.
- labels - A comma separated list of labels to associate with the new dbDef.
- dbDefType - One of either 'standard' or 'dmcdb' to specify whether this is a standard dbDef or the dbdef for the DMC Web Application. Defaults to 'standard'.
- storageOptionCollectedAtSnapshot - whether the system will pay attention to storage options on table, indexes, etc.
- tnsName - (only applies to Oracle) The TNS name for the Oracle host and service you wish to manage. Requires tnsnames.ora to be appropriately configured on the client.
- kerberos - (only applies to Oracle) true or false. Set to true if using Kerberos authentication to connect. If true, tnsName must also be specified. Requires additional client configuration.

---

newProject

Creates a new empty project in the current directory.

USAGE

```
newProject [projectName]
```

EXAMPLE

```
newProject myProject
```

OPTIONS

- `projectName` - Specifies the project name. If not specified, the name **newProject** is used.

NOTES

Create the project directory before creating the project. This command does NOT create a new directory for the project.

---

registerProjectWithDMCDB

Registers the project with the DMC database. This command is for DMC versions 7.0 and higher.

USAGE

registerProjectWithDMCDB

EXAMPLE

registerProjectWithDMCDB

NOTES

You may need to run the hammer newDbDef command first if the dbDef for DMC hasn't been created yet.

releaseLocks

Reset the lock Datical creates in an environment to prevent concurrent Datical operations (DATABASECHANGELOGLOCK table) for all supported database platforms.

With versions 7.3 or higher, if the target is an Oracle database this command will also look for the `DATICAL_SPERRORLOG` table and attempt to drop it if it's found. The `DATICAL_SPERRORLOG` table is used to capture output for scripts executed using `sqlplus` during a deployment.

USAGE

hammer releaseLocks <dbDef>

EXAMPLE

hammer releaseLocks PROD_DB

NOTES

Typically this lock is cleared at the end of a deployment but a deployment that ends prematurely may leave this lock in place. Subsequent deployments to such an environment will fail until the lock is released.

---

removeContexts

Remove contexts from changesets.

> **Note**
> Starting with v4.32, please use the `modifyChangeSet` command. The `removeContexts` command is deprecated and support may be removed in the future.

USAGE

```
removeContexts --lookupChangeset=<label-expression> --context=<contexts> [--scriptChecksum=][--
matchAll=true|false]
```

OPTIONS

- `lookupChangesets` - Label expression used to matchthe change sets (required)
- `context` - Comma-separated list of contexts (required)
- `scriptChucksum` - SQL script checksum value
- `matchAll` - SQL Script checksums must match for change set lookup. Default value is true.

NOTES

The value of the `scriptChecksum` parameter can be used to further qualify change sets that match the label expression. This is used primarily in the Deployment Packager to confirm that the change set was generated by the expected version of a script.

The value of the `matchAll` parameter impacts the matching behavior of the `scriptChecksum` parameter.

- If `matchAll` is true, all change sets identified by the label expression must have a `scriptChecksum` attribute that matches the attribute passed on the command line.
- If `matchAll` is false, only one change set returned by the label expression needs to have a matching `scriptChecksum` attribute.

---

removeLabels

Remove labels from changesets.

> **Note**

Starting with v4.32, please use the `modifyChangeSet` command. The `removeLabels` command is deprecated and support may be removed in the future.

USAGE

```
removeLabels --lookupChangeset=<label-expression> --context=<contexts> [--scriptChecksum=][--
matchAll=true|false]
```

OPTIONS

- `lookupChangesets` - Label expression used to matchthe change sets (required)
- `context` - Comma-separated list of contexts (required)
- `scriptChucksum` - SQL script checksum value
- `matchAll` - SQL Script checksums must match for change set lookup. Default value is true.

NOTES

The value of the `scriptChecksum` parameter can be used to further qualify change sets that match the label expression. This is used primarily in the Deployment Packager to confirm that the change set was generated by the expected version of a script.

The value of the `matchAll` parameter impacts the matching behavior of the `scriptChecksum` parameter.

- If `matchAll` is true, all change sets identified by the label expression must have a `scriptChecksum` attribute that matches the attribute passed on the command line.
- If `matchAll` is false, only one change set returned by the label expression needs to have a matching `scriptChecksum` attribute.

---

rollback

Roll back the target database to a specified revision, date or by number of steps.

USAGE

```
rollback <dbdef> lastlabel
rollback <dbdef> changeid:<changesetId>
rollback <dbdef> count:<num>
rollback <dbdef> date:<date>
rollback <dbdef> tag:<tag_name>
rollback <dbdef> lastDeploy
```

Note
If you are using a current packaging methodology (Deployment Packager with SCM), use only the `lastDeploy` option. Other options support older methodologies.

EXAMPLE

```
rollback <dbdef> lastDeploy --pipeline=FeatureA
```

OPTIONS

- `lastlabel` - Remove all changes associated with the last label deployed. Certain preconditions must be met for this mode to succeed.
- `changeid:<changesetID>` - Roll back to the `<changesetId>` change set in the change log.
- `count:<num>` - Roll back the last `<num>` change sets.
- `date:<date>` - Roll back all change sets applied after `<date>`.
- `tag:<tag_name>` - Roll back to `<tag_name>`. The tag `<tag_name>` must have been specified for a previous changeset.
- `lastDeploy` - Remove all changes associated with the last `deploy` operation.
- `pipeline` - Specifies the pipeline associated with the changelogSync for reporting purposes.
    - If not set and the dbDef is a member of only one pipeline, then that pipeline is used.
    - If not set and the dbDef is a member of multiple pipelines, then an error is returned.

OPTION VALUES

- `<dbdef>` - Name of a DbDef in the current project.

- `<changesetId>` - The author and id of a change set, separated by two pipe characters. Format: "id=createTable1||author=Datical User"
- `<num>` - Number of change sets to revert.
- `<date>` - Date boundary used to determine which change sets will be reverted. Format: yyyy-MM-dd
- `<tag_name>` - Datical tag name to which you wish to rollback. Requires that you tag the database in a previous change set.

---

runRules

Execute the project's 'sqlRules' against an individual SQL script, a list of scripts or a folder of scripts.

USAGE

```
runRules [SQL file | List of SQL files | SQL files folder]
```

EXAMPLES

```
runRules git/sql/ddl/create_user_table.sql
```
OR
```
runRules git/sql/ddl/create_user_table.sql,git/sql/ddl/add_user_index.sql,git/sql/ddl/create_co_table.sql
```

OR
```
runRules "git/sql/ddl/create_user_table.sql;git/sql/ddl/add_user_index.sql;git/sql/ddl/create_co_table.sql"
```

OR
```
runRules git/sql/ddl
```

NOTES

Deployment Packager runs the `runRules` command as part of packaging and in `preview` mode. You can use it to check SQL files before checking them in for packaging.

The command must be run from a project directory: `datical/<project>`.

In the workspace (`datical`), the `SQLRules` directory is under the `Rules` directory in a project: `datical/<project-name>/Rules/SQLRules`.

Use commas or semicolons to separate lists of files passed to the command. If semicolons are used, the list should surrounded by double quotes.

---

schemaStats

Display summary schema info. Shows summary info about the database model represented by the changelog.

USAGE

```
schemaStats
```

---

set

Set properties for the project and database.

The set command provides help for the subcommands.

USAGE

```
set &lt;property&gt; &lt;values&gt;
```

```
set help
```

PROPERTIES

- `alternateSchema` - Set the alternate schema for the project.

  USAGE: `set alternateSchema <schema name>`

- `autoGenSQL` – control whether or not SQL scripts are automatically generated during Forecast, Deploy, and Rollback operations.

  USAGE: `set autoGenSQL true|false`

- `dbprop` - Set a database property in a project.

  USAGE: `set dbprop [dbdef] [propertyName] [propertyValue]`

  - `dbdef` - the name of a database definition in the current project.
  - propertyName - a property name. The only property currently supported is `enableStorageOptions`. This property controls whether or not storage options are collected for a specific database definition during the snapshot command. Storage Option information includes storage configuration, tablespace/filegroup assignments, large object storage parameters and partitioning information for tables and indexes. This defaults to `false`.
  - propertyValue - a property value (must be valid for the type of the property).

  EXAMPLE: `set dbProp QDdb enableStorageOptions true`

- `deployMode` – controls how much work is done during Deploy operations.

  USAGE: `set deployMode quick|full`

  - `full` - performs rules validation and runs a forecast before deploying the changes, subject to `limitForecast`.
  - `quick` - deploys the changes (no rules valildation or forecast)

- `deployThreshold` – controls what happens when a full deploy detects errors or warnings in rules or in forecast.

  USAGE: `set deployThreshold stopOnError|stopOnWarn|deployAlways>`

  - `stopOnError` - Deployment is not performed when errors are present in Pre-Deployment validation
  - `stopOnWarn` - Deployment is not performed when errors and/or Warnings are present in Pre-Deployment validation
  - `deployAlways` - Deploy proceeds regardless of validation results. NOT RECOMMENDED

- enableRowCount - set method for counting rows in tables.
  Changed in v5.7 from `true | false` with default of true. New default is `disabled`. Older projects are interpreted correctly and mapped to the new values.

  USAGE: `set enableRowCount exact | approximate | disabled`

  - `exact` - Uses a table scan to count the number of rows in each table.
  - `approximate` - Uses database statistics to approximate the number of rows.
  - `disable` - Does not count table rows.

- enableSqlParser - set use of SQL Parser to allow forecasting of DIRECT, DDL_DIRECT, and SQLFILE scripts

  USAGE: `set enableSqlParser true | false`

  - `true` - use SQL Parser to parse DIRECT, DDL_DIRECT, and SQLFILE scripts into Datical's object model for subsequent validation with rules and forecast.
  - `false` - disable SQL Parser.

- `externalStoredLogic` - Set whether stored logic definitions (stored procedures, etc.) are stored in external files or within the project changelog.

  USAGE: `set externalStoredLogic <true|false>`

- `forecastDML` - For Oracle only, controls whether Datical DB attempts to forecast DML (Data Manipulation Language).

  USAGE: set forecastDML `true | false`

  - `true` - forecasts predict whether a DML change will be successful
  - `false` - forecasts warn that any DML change encountered will not be forecast

- `help <property>` - show more help about a property.
- `invalidsCheck` – controls how to check the validity of stored logic after deployment

  USAGE: `set invalidsCheck disabled|limited|local|global`

  - `disabled` - do not perform the check
  - `limited` - only check objects targeted by a deployment and their dependencies
  - `local` - check the target schema only
  - `global` - check all schema in the database

- `invalidsAction` – controls how to check the validity of stored logic after deployment

  USAGE: `set invalidsAction warn|fail`

  - `warn` - print a warning message if invalidsCheck does not pass.
  - fail - print an error message if `invalidsCheck` does not pass for new stored logic objects. Print a warning if the check does not pass for objects that already exist in the database. If set to fail and a package operation encounters the error during deployment to REF, the package operations fails and the database is restored to its state before the packaging operation was started.

- `limitForecast` - controls how Datical DB behaves during forecast operations. Forecast operations run faster if set to true, but some rules are not checked.

USAGE: `set limitForecast true|false`

- `true` - limit object profiling to those directly affected by the forecast
- `false` - profile all objects in the target database

- `nlsLang` - allows the user to set NLS_LANG value to interact with a database in their own language, as defined by the NLS_LANG parameter

  USAGE: `set nlsLang <String>`
  EXAMPLE: `set nlsLang SPANISH_SPAIN.WE8ISO8859P1`

- property - Sets the specified property to a given value. The only property currently supported is `drivers`. This points to the directory where you've stored your JDBC drivers.

  USAGE: `set property [propertyName] [propertyValue]`

  EXAMPLE: `set property drivers /path/to/driver/dir`

- `requireOptions` – controls whether or not labels or contexts are required for Forecast and Deploy

  USAGE: `set requireOptions true|false`

- `runtimeCredentials` - controls whether database credentials are stored or prompted at run time. Default: false. If set to true, all stored credentials are removed and you are prompted for database credentials during forecasting and deployment. If set to false, database credentials are encoded and stored in the project. ***We strongly recommend to use runtime credentials because it is more secure than stored credentials (due to the possibility of the stored password being decoded)***.

  USAGE `set runtimeCredentials true|false`

  - `true` - all stored credentials (if any) are removed. You are prompted for database credentials during forecasting and deployment.
  - `false` - (default) database credentials are stored in the project.

- `scriptExecutionTimeout` - sets a limit on how long Datical DB will wait for a script to run. We recommend setting a value for this for all of your REF dbDefs, but only your REF dbDefs.

  USAGE: `set scriptExecutionTimeout <dbdefName> <seconds>`

  - `dbdefName` - name of the dbdef where the limit is applied
  - `seconds` - number of seconds. If set to 0, there is no timeout limit. The default value is 0.

- `storageOptionsScope` - Specifies what storage information to use in snapshot and diff operations for all steps/dbdefs in the project. For `<arg>`, specify one of `all`, `lob`, `tablespace`, `tablespaceAndLob`. The default is `all`. This setting works only for steps/dbdefs that have `enableStorageSettings` set to `true`.

  USAGE: `set storageOptionsScope <arg>`

  - `<arg>` - one of `all, lob, tablespace, or tablespaceAndLob`

SEE ALSO

`show` command

---

show

Display properties of the project and Datical DB.

USAGE

`show <property>`

PROPERTIES

- `autoGenSQL` - show whether the project automatically generates SQL when deploying
- `bundles` - lists the OSGI bundles in the runtime environment. Used by Datical support for diagnostics.
- `databases` - lists all the databases that are in the project.
- `dbprop` - shows a database property in a project.

  USAGE: `show dbprop [dbdef] [propertyName]`

    - dbdef - the name of a database definition in the current project.
    - propertyName - a property name. Currently only `enableStorageOptions` is supported.
- `deployMode` - full or quick
- `deployThreshold` - stopOnError, stopOnWarn, or deployAlways
- `enableRowCount - exact | approximate | disabled`, method for counting rows in tables. Default is disabled.
    - Before v5.7 used `true | false`, with default as true. New default is `disabled`. Older projects are interpreted correctly and mapped to the new values.
- `externalStoredLogic` - true or false. If true, store external functions (like stored procedures) in external files. Otherwise they are stored in the changelog.
- `help <property>` - show more help about a property.
- `invalidsCheck - global|local|limited|disabled` - how to check stored logic validity during deployment: `local` - target schema (default), `global` - all schema in the target database, `limited` - available with 5.3 or higher, only objects targeted by a deployment and their dependencies, `disabled` - no validation check
- `invalidsAction - warn|fail`
- `languages` - lists all scripting languages that are supported by the current Java runtime
- `license` - shows status of the license: whether it is installed, issue date, expiration date
- `limitForecast` - true or false, limit object profiling to only affected. All objects are profiled if set to true.
- `project` - name and description of the current project.
- `properties` - shows all the properties that have been set.
- `property` - shows the value of the given propertyname.

    USAGE: show property <propertyName>

- `requireoptions` - true or false, whether contexts and labels are required for Forecast and Deploy
- `runtimeCredentials` - true or false, whether user is prompted at run time for database user credentials. Otherwise stored encoded credentials are used. ***We strongly recommend to use runtime credentials because it is more secure than stored credentials (due to the possibility of a stored password being decoded)***.
- `scriptExecutionTimeout` - number of seconds, 0 if unlimited.

    USAGE: `show scriptExecutionTimeout [dbdef]`

    - dbdef - name of a database definition in the current project
- `version` - shows version information for the components of Datical DB

SEE ALSO

`set` command

---

snapshot

Capture a snapshot of a database as either a persistent object file or a changelog

USAGE

```
snapshot <deploymentStep> [--format=dbobject|changelog][--output=][--ddlExcludeList=<object-list>][--snapshotSchemaList=<schema-list]
```

EXAMPLE

```
snapshot DEVdb --output=/home/user/snapshots/
```

OPTIONS

- `format` - Specifies the format of the output, `dbobject` (default) or `changelog`.
    - `dbobject` - object list that can be used by the `diff` and `diffChangelog` commands
    - `changelog` - XML file suitable for an initial project changelog
- `output=<path>` - When present, the changelog generated is written to the directory specified by `<path>`. By default file is written to the `Snapshots` directory of the project.
- `ddlExcludeList` - comma-separated list of object types to exclude from the snapshot. Can be `all`, `none`, or a list containing any combination of `procedure`, `function`, `package`, `packagebody`, `trigger`, `view`.
- `snapshotSchemaList` - comma-separated list of schemas to include in the snapshot

NOTES

If the current project is set to externalize stored logic, the stored logic is written to a `Resources` subdirectory. See the set command for a subcommand to see the values for a subcommand. within the date stamped directory created by the snapshot process.

---

status

Show the deployment status of a database.

USAGE

```
status [dbdef]
```

EXAMPLE

```
status QAdb
```

```
hammer --daticalServer=dmc_server_url --daticalUsername=user_name status datical_project_name
```

DATICAL SERVICE OPTIONS

Use these options to access a project that is stored on Datical Service.

- `--daticalServer=<cluster-name>` - The cluster name set for Datical Service. The cluster name is set during Datical Service installation.
- `--daticalUsername=<user>` - User name, must be configured in Datical Service. Use the DATICAL_PASSWORD environment variable to provide the password.
- status - keyword that identifies this as a status command
- `<projectRef>` - Project to use, specified `<projectRef>`, a project name or project key. The project key is defined in Datical Service only.

OPTIONS

- `dbdef` - Name of a database definition in the project.

OUTPUT

For each dbdef, the status reported is one of the following.

- Deployed - all changesets in REF are deployed on the dbdef (managed database)
- Undeployed - not all changesets in REF are deployed on the dbdef
- Error **-** cannot contact database, no JDBC found, others

NOTES

The status command reports on all dbdefs (databases) in the project unless you specify an individual dbdef.

Status is determined by comparing the changesets in the REF database (the changelog) to the changesets listed in the tracking tables for each database.

---

statusDetails

Show the detailed status for each of the changesets in the changelog.

USAGE

```
statusDetails <DbDef>
```

EXAMPLE

```
statusDetails QAdb
```

NOTES

If you run `statusDetails` without a DbDef, a message prompts you to provide one and lists the available DbDefs.

Changesets are shown in three categories: Deployed, Undeployed, Ignored. The total number if changesets in each state is shown.

OUTPUT

statusDetails Output

testConnect

Test the database connection for one or more databases.

USAGE

```
testConnect <DbDef>
```

EXAMPLE

```
testConnect QAdb
```

NOTES

If run without a DbDef, it tests the connections to all DbDefs in the project.

---

uninstallDrivers

Run the custom script `uninstallDrivers.groovy`. Removes JDBC drivers.

USAGE

```
uninstallDrivers
```

---

uninstallLicense

Removes any installed DaticalDB licenses.

USAGE

```
uninstallLicense
```

---

validateStoredLogic

Compile stored logic and report any validation errors.

USAGE

```
validateStoredLogic <dbdef>
```

OPTIONS

- `dbdef` - Name of a database definition in the project.

---

versionChangeLog

Upgrade project changelog to the current version.

USAGE

```
versionChangeLog [--force]
```

OPTIONS

- `force` - also forces the XML header to be checked, validated, and updated. Output includes the message "Forcing XML header check..."

NOTES

Required: the current directory is a valid project directory.

Output tells you the current version and upgraded version of the project changelog.

You can only upgrade the changelog version. Downgrading the version is not supported.