Commands

This article is part of the Getting Started Section. It lists all database commands supported by BaseX.

Commands can be executed from the Command Line, as part of Scripts, via the Clients, REST, the input field in the GUI, and other ways. If the GUI is used, all commands that are triggered by the GUI itself will show up in the Info View. The Permission fields indicate which rights are required by a user to perform a command in the client/server architecture.

Basics

Command Scripts

Multiple commands can be written down in a single line – separated by semicolons – or put into a command script: Database commands in both string and XML syntax can be placed in a text file and stored as file with the BaseX command script suffix `.bxs`. If the path to a script file is passed on to BaseX on command-line, or if it is opened in the GUI editor, it will be recognized and evaluated as such.

String Syntax

Lines starting with `#` are interpreted as comments and are skipped. With the following script, a database is created, two documents are added to it, and a query is performed:

```
CREATE DB test
ADD TO embedded.xml <root>embedded</root>
# run query
XQUERY <hits>{ count(//text()) }</hits>
CLOSE
```

XML Syntax

The string syntax is limited when XML snippets need to be embedded in a command, or when complex queries are to be specified.

The XML syntax provides more flexibility here. Multiple commands can be enclosed by a `<commands/>` root element. Some commands, such as `ADD`, allow you to directly embed XML documents. If you want to embed XML in XQuery expressions, entities should be encoded, or the `CDATA` syntax should be used:

```
<commands>
  <create-db name='test'/>
  <add path='embedded.xml'><root>embedded</root></add>
  <!-- run query -->
  <xquery><![CDATA[
    <hits>{ count(//text()) }</hits>
  ]]></xquery>
  <close/>
</commands>
```

Glob Syntax

Some commands support the glob syntax to address more than one database or user. Question marks and asterisks can be used to match one or more characters, and commas can be used to separate multiple patterns. Some examples:

- AB? addresses all names with the characters AB and one more character.
- *AB addresses all names ending with the characters AB.
- X*,Y*,Z* addresses all names starting with the characters X, Y, or Z.

Valid Names

Updated with Version 8.5: allow dots (except as first and last character).

Database and user names follow the same naming constraints: Names are restricted to ASCII characters. They must at least have one character, and they may contain letters, numbers and any of the special characters !#$%&'()+-=@[]^_`{}~. The following characters are reserved for other features:

- ,?*: glob syntax
- ;: Separator for multiple database commands on the command line
- \/: Directory path separators
- :*?\"<>|}: invalid filename characters on Windows
- Names starting or ending with .: hidden folders (e.g. the .logs directory)

Aliases

In all commands, the DB keyword can be replaced by DATABASE.

Database Operations

CREATE DB

Syntax CREATE DB [name] ([input])

XML Syntax <create-db name='...'>([input])</create-db>

Permission *CREATE*

Summary Creates a new database with the specified name and, optionally, an initial input, and opens it. An existing database will be overwritten.

The input may either be a reference to a single XML document, a directory, a remote URL, or a string containing XML:

- name must be a valid database name
- database creation can be controlled by setting Create Options

Errors The command fails if a database with the specified name is currently used by another process, if one of the documents to be added is not well-formed or if it cannot be parsed for some other reason.

Examples

- CREATE DB input
  creates an empty database input.
- CREATE DB xmark [1]
  creates the database xmark, containing a single initial document called xmark.xml.
- CREATE DATABASE coll /path/to/input
  creates the database coll with all documents found in the input directory.
- SET INTPARSE false; CREATE DB input input.xml
  creates a database input with input.xml as initial document, which will be parsed with Java's default XML parser.
- <create-db name='simple'><hello>Universe</hello></create-db>
  creates a database named simple with an initial document <hello>Universe</hello>.

OPEN

Syntax `OPEN [name] ([path])`

XML Syntax `<open name='...' (path='...')/>`

Permission *READ*

Summary Opens the database specified by `name`. The documents to be opened can be specified by the [path] argument.

Errors The command fails if the specified database does not exist, is currently being updated by another process or cannot be opened for some other reason.

CHECK

Syntax `CHECK [input]`

XML Syntax `<check input='...'/>`

Permission *READ/CREATE*

Summary This convenience command combines `OPEN` and `CREATE DB`: if a database with the name `input` exists, it is opened. Otherwise, a new database is created; if the specified input points to an existing resource, it is stored as initial content.

Errors The command fails if the addressed database could neither be opened nor created.

CLOSE

Syntax `CLOSE`

XML Syntax `<close/>`

Permission *READ*

Summary Closes the currently opened database.

Errors The command fails if the database files could not be closed for some reason.

EXPORT

Syntax `EXPORT [path]`

XML Syntax `<export path='...'/>`

Permission *CREATE*

Summary Exports all documents in the database to the specified file `path`, using the serializer options specified by the `EXPORTER` option.

Errors The command fails if no database is opened, if the target path points to a file or is invalid, if the serialization parameters are invalid, or if the documents cannot be serialized for some other reason.

CREATE INDEX

Updated with Version 8.4: Token index added.

Syntax `CREATE INDEX [TEXT|ATTRIBUTE|TOKEN|FULLTEXT]`

XML Syntax `<create-index type='text|attribute|token|fulltext'/>`

Permission *WRITE*

Summary Creates the specified Value Index. The current Index Options will be considered when creating the index.

Errors The command fails if no database is opened, if the specified index is unknown, or if indexing fails for some other reason.

DROP INDEX

Syntax `DROP INDEX [TEXT|ATTRIBUTE|TOKEN|FULLTEXT]`

XML Syntax `<drop-index type='text&#x7c;attribute&#x7c;token&#x7c;fulltext'/>`

Permission *WRITE*

Summary Drops the specified Value Index.

Errors The command fails if no database is opened, if the specified index is unknown, or if it could not be deleted for some other reason.

Administration

ALTER DB

Syntax `ALTER DB [name] [newname]`

XML Syntax `<alter-db name='...' newname='...'/>`

Permission *CREATE*

Summary Renames the database specified by `name` to `newname`. `newname` must be a valid database name.

Errors The command fails if the target database already exists, if the source database does not exist or is currently locked, or if it could not be renamed for some other reason.

Examples

- `ALTER DB db tempdb`
  renames the database `db` into `tempdb`.

DROP DB

Syntax `DROP DB [name]`

XML Syntax `<drop-db name='...'/>`

Permission *CREATE*

Summary Drops the database with the specified `name`. The Glob Syntax can be used to address more than one database.

Errors The command fails if the specified database does not exist or is currently locked, or if the database could not be deleted for some other reason.

CREATE BACKUP

Syntax `CREATE BACKUP [name]`

XML Syntax `<create-backup name='...'/>`

Permission *CREATE*

Summary Creates a zipped backup of the database specified by `name`. The backup file will be suffixed with the current timestamp and stored in the database directory. The Glob Syntax can be used to address more than one database.

Errors The command fails if the specified database does not exist, or if it could not be zipped for some other reason.

Examples

- `BACKUP db`
  creates a zip archive of the database `db` (e.g. `db-2014-04-01-12-27-28.zip`) in the database directory.

RESTORE

Syntax `RESTORE [name]`

XML Syntax `<restore name='...'/>`

Permission *CREATE*

Summary Restores a database with the specified `name`. The name may include the timestamp of the backup file.

Errors The command fails if the specified backup does not exist, if the database to be restored is currently locked, or if it could not be restored for some other reason.

INSPECT

Syntax `INSPECT`

XML Syntax `<inspect/>`

Permission *READ*

Summary Performs some integrity checks on the opened database and returns a brief summary.

DROP BACKUP

Syntax `DROP BACKUP [name]`

XML Syntax `<drop-backup name='...'/>`

Permission *CREATE*

Summary Drops all backups of the database with the specified `name`. The Glob Syntax can be used to address more than one database.

Examples

- `DROP BACKUP abc*`
  deletes the backups of all databases starting with the characters `abc`.

SHOW BACKUPS

Syntax `SHOW BACKUPS`

XML Syntax `<show-backups/>`

Permission *CREATE*

Summary Shows all database backups.

COPY

Syntax `COPY [name] [newname]`

XML Syntax `<copy name='...' newname='...'/>`

Permission *CREATE*

Summary Creates a copy of the database specified by `name`. `newname` must be a valid database name.

Errors The command fails if the target database already exists, or if the source database does not exist.

INFO DB

Syntax `INFO DB`

XML Syntax `<info-db/>`

Permission *READ*

Summary Shows information on the currently opened database.

Errors The command fails if no database is opened.

INFO INDEX

Syntax `INFO INDEX ([ELEMNAME|ATTRNAME|PATH|TEXT|ATTRIBUTE|TOKEN|FULLTEXT])`

XML Syntax `<info-index type='elemname&#x7c;attrname&#x7c;path&#x7c;text&#x7c;attribute&#x7c;token&#x7c;fulltext'/>`

Permission *READ*

Summary Shows information on the existing index structures. The output can be optionally limited to the specified index.

Errors The command fails if no database is opened, or if the specified index is unknown.

INFO STORAGE

Syntax `INFO STORAGE [start end]`

XML Syntax `<info-storage (start='...') (end='...')/>`

Permission *READ*

Summary Shows the internal main table of the currently opened database. An integer range may be specified as argument.

Errors The command fails if no database is opened, or if one of the specified arguments is invalid.

Querying

LIST

Syntax `LIST ([name] ([path]))`

XML Syntax `<list (name='...' (path='...'))/>`

Permission *NONE*

Summary Lists all available databases. If `name` is specified, the resources of a database are listed. The output can be further restricted to the resources matching the specified `path`.
If database resources are listed, the size is either the number of nodes (for XML resources) or the number of bytes (for binary resources).

Errors The command fails if the optional database cannot be opened, or if the existing databases cannot be listed for some other reason.

XQUERY

Syntax `XQUERY [query]`

XML Syntax `<xquery>[query]</xquery>`

Permission *depends on query*

Summary Runs the specified `query` and prints the result.

Errors The command fails if the specified query is invalid.

Examples

- `XQUERY 1 to 10`
  returns the sequence `(1, 2, 3, 4, 5, 6, 7, 8, 9, 10).`
- `SET RUNS 10; XQUERY 1 to 10`
  returns the results after having run the query 10 times.
- `SET XMLPLAN true; XQUERY 1 to 10`
  returns the result and prints the query plan as XML.

RETRIEVE

Syntax `RETRIEVE [path]`

XML Syntax `<retrieve path='...'/>`

Permission *READ*

Summary Retrieves a raw file from the opened database at the specified `path`.

Errors The command fails if no database is opened, if the source path is invalid or if the data cannot not be retrieved for some other reason.

FIND

Syntax `FIND [query]`

XML Syntax `<find>[query]</find>`

Permission *READ*

Summary Builds and runs a query for the specified `query` terms. Keywords can be enclosed in quotes to look for phrases. The following modifiers can be used to further limit search: `=` looks for exact text nodes
`~` looks for approximate hits
`@=` looks for exact attribute values
`@` looks for attributes

Errors The command fails if no database is opened.

TEST

Syntax `TEST [path]`

XML Syntax `<test path='...'/>`

Permission *ADMIN*

Summary Runs all XQUnit tests in the specified `path`. The path can point to a single file or a directory.
Unit testing can also be triggered via `-t` on command line.

Errors The command fails if at least one test fails.

Examples

- `TEST project/tests`
  runs all tests in the directory `project/tests`.

REPO INSTALL

Syntax `REPO INSTALL [path]`

XML Syntax `<repo-install path='...'/>`

Permission *CREATE*

Summary Installs the package with path `path`.

Errors The command fails in the following cases:

- The package to be installed is not a xar file.
- The package to be installed does not exist or is already installed.
- The package descriptor is with invalid syntax.
- The package to be installed depends on a package which is not installed.
- The package is not supported by the current version of BaseX.
- A component of the package is already installed as part of another package.

REPO LIST

Syntax `REPO LIST`

XML Syntax `<repo-list/>`

Permission *READ*

Summary Lists all installed packages.

REPO DELETE

Syntax `REPO DELETE [name]`

XML Syntax `<repo-delete name='...'/>`

Permission *CREATE*

Summary Deletes the package with name `name`, optionally followed by a version.

Errors The command fails if the package to be deleted participates in a dependency.

Updates

ADD

Syntax `ADD (TO [path]) [input]`

XML Syntax `<add (path='...')>[input]</add>`

Permission *WRITE*

Summary Adds a file, directory or XML string specified by `input` to the currently opened database at the specified `path`:

- `input` may either be a single XML document, a directory, a remote URL or a plain XML string.
- A document with the same path may occur than once in a database. If this is unwanted, the `REPLACE` command can be used.
- If a file is too large to be added in one go, its data structures will be cached to disk first. Caching can be enforced by turning the `ADDCACHE` option on.

If files are to be added to an empty database, it is usually faster to use the `CREATE DB` command and specify the initial input as argument.

Errors The command fails if no database is opened, if one of the documents to be added is not well-formed, or if it could not be parsed for some other reason.

Examples

- `ADD input.xml`
  adds the file `input.xml` to the database.
- `ADD TO temp/one.xml input.xml`
  adds `input.xml` to the database and moves it to `temp/one.xml`.
- `ADD TO target/ xmldir`
  adds all files from the `xmldir` directory to the database in the `target` path.

DELETE

Syntax `DELETE [path]`

XML Syntax `<delete path='...'/>`

Permission *WRITE*

Summary Deletes all documents from the currently opened database that start with the specified `path`.

Errors The command fails if no database is opened.

RENAME

Syntax `RENAME [path] [newpath]`

XML Syntax `<rename path='...' newpath='...'/>`

Permission *WRITE*

Summary Renames all document paths in the currently opened database that start with the specified `path`. The command may be used to either rename single documents or directories.

Errors The command fails if no database is opened, or if the target path is empty.

Examples

- `RENAME one.xml two.xml`
  renames the document `one.xml` to `two.xml`.
- `RENAME / TOP`
  moves all documents to a `TOP` root directory.

REPLACE

Syntax `REPLACE [path] [input]`

XML Syntax `<replace path='...'>[input]</replace>`

Permission *WRITE*

Summary Replaces a document in the currently opened database, addressed by `path`, with the file or XML string specified by `input`, or adds a new document if the resource does not exist yet.

Errors The command fails if no database is opened, if the specified path points to a database directory, or if the input is not found.

Examples

- `REPLACE one.xml input.xml`
  replaces the document `one.xml` with the contents of the file `input.xml`.
- `REPLACE top.xml <xml/>`
  replaces the document `top.xml` with the document `<xml/>`.

STORE

Syntax `STORE (TO [path]) [input]`

XML Syntax `<store (path='...')>[input]</store>`

Permission *WRITE*

Summary Stores a raw file in the opened database to the specified `path`. `input` may either be a file reference, a remote URL, or a plain string. If the path denotes a directory, it needs to be suffixed with a slash (`/`).

Errors The command fails if no database is opened, if the specified resource is not found, if the target path is invalid or if the data cannot not be written for some other reason.

OPTIMIZE

Syntax `OPTIMIZE (ALL)`

XML Syntax `<optimize/>`
`<optimize-all/>`

Permission *WRITE*

Summary Optimizes the index structures, meta data and statistics of the currently opened database:

- If `ALL` is specified, all database structures are completely reconstructed. The database size will be reduced, and all orphaned data will be deleted.
- Without `ALL`, only the outdated index structures and database statistics will be updated. If the database is completely up-to-date, nothing will be done.
- Database options will be adopted from the original database. Only `AUTOOPTIMIZE` and (if `ALL` is specified) `UPDINDEX` will be adopted from the current options.

Errors The command fails if no database is opened, or if the currently opened database is a main-memory instance.

FLUSH

Syntax `FLUSH`

XML Syntax `<flush/>`

Permission *WRITE*

Summary Explicitly flushes the buffers of the currently opened database to disk. This command is applied if `AUTOFLUSH` has been set to `false`.

Errors The command fails if no database is opened.

Server Administration

SHOW SESSIONS

Syntax `SHOW SESSIONS`

XML Syntax `<show-sessions/>`

Permission *ADMIN*

Summary Shows all sessions that are connected to the current server instance.

SHOW USERS

Syntax `SHOW USERS (ON [database])`

XML Syntax `<show-users (database='...')/>`

Permission *ADMIN*

Summary Shows all users that are registered in the database. If a `database` is specified, all user will be shown for which a pattern was specified that matches the database name.

Errors The command fails if the optional database could not be opened.

KILL

Syntax `KILL [target]`

XML Syntax `<kill target='...'/>`

Permission *ADMIN*

Summary Kills sessions of a user or an IP:port combination, specified by `target`. The Glob Syntax can be used to address more than one user.

Errors The command fails if a user tried to kill his/her own session.

User Management

CREATE USER

Syntax `CREATE USER [name] ([password])`

XML Syntax `<create-user name='...'>([password])</create-user>`

Permission *ADMIN*

Summary Creates a user with the specified `name` and `password`. If no password is specified, it is requested via the chosen frontend (GUI or bash).

Errors The command fails if the specified user already exists.

ALTER USER

Syntax `ALTER USER [name] ([newname])`

XML Syntax `<alter-user name='...' newname='...'/>`

Permission *ADMIN*

Summary Renames the user with the specified `name` to `newname`.

Errors The command fails if the specified user does not exist, or if the new user already exists.

ALTER PASSWORD

Syntax `ALTER PASSWORD [name] ([password])`

XML Syntax `<alter-password name='...'>([password])</alter-password>`

Permission *ADMIN*

Summary Alters the `password` of the user with the specified `name`. If no password is specified, it is requested via the chosen frontend (GUI or bash).

Errors The command fails if the specified user does not exist.

DROP USER

Syntax `DROP USER [name] (ON [pattern]):`

XML Syntax `<drop-user name='...' (pattern='...')/>`

Permission *ADMIN*

Summary Drops the user with the specified `name`. The Glob Syntax can be used to address more than one database or user. If a glob `pattern` is specified, only the pattern will be removed.

Errors The command fails if `admin` is specified as user name, or if the specified user does not exist or is currently logged in.

GRANT

Syntax `GRANT [NONE|READ|WRITE|CREATE|ADMIN] (ON [pattern]) TO [user]`

XML Syntax `<grant name='...' permission='none&#x7c;read&#x7c;write&#x7c;create&#x7c;admin' (pattern='...')/>`

Permission *ADMIN*

Summary Grants the specified permission to the specified `user`. The Glob Syntax can be used to address more than one user. If a glob `pattern` is specified, the permission will be applied to all databases that match this pattern.

Errors The command fails if `admin` is specified as user name or if the specified user does not exist.

Examples

- `GRANT READ TO JoeWinson`
  grants `READ` permission to the user `JoeWinson`.
- `GRANT WRITE ON Wiki TO editor*`
  grants `WRITE` permissions on the `Wiki` database to all users starting with the characters `editor*`.

PASSWORD

Syntax `PASSWORD ([password])`

XML Syntax `<password>([password])</password>`

Permission *NONE*

Summary Changes the `password` of the current user. If no password is specified, it is requested via the chosen frontend (GUI or bash).

General Commands

RUN

Syntax `RUN [file]`

XML Syntax `<run file='...'/>`

Permission *depends on input*

Summary Evaluates the contents of `file` as XQuery expression. If the file ends with the suffix `.bxs`, the file content will be evaluated as command script. This command can be used to run several commands in a single transaction.

Errors The command fails if the specified file does not exist, or if the retrieved input is invalid. It will be canceled as soon as one of the executed commands fails.

Examples

- `RUN query.xq`
  will evaluated the specified file as XQuery expression
- `RUN commands.bxs`
  will evaluated the specified file as command script

EXECUTE

Syntax `EXECUTE [input]`

XML Syntax `<execute>[input]</execute>`

Permission *depends on input*

Summary Evaluates the specified `input` as command script. This command can be used to run several commands in a single transaction.

Errors The command fails if the syntax of the specified input is invalid. It will be canceled as soon as one of the executed commands fails.

Examples

- `EXECUTE "create db db1; create db db2"`
- `EXECUTE "<commands><create-db name='db1'/><create-db name='db2'/></commands>"`
  both commands will create two databases `db1` and `db2` in a single transaction.

GET

Syntax `GET [option]`

XML Syntax `<get (option='...')/>`

Permission *NONE*

Summary Returns the current value of the Option specified via `option`. Global options can only be requested by users with ADMIN permissions.

Errors The command fails if the specified option is unknown.

SET

Syntax `SET [option] ([value])`

XML Syntax `<set option='...'>([value])</set>`

Permission *NONE*

Summary Sets the Option specified by `option` to a new `value`. Only local options can be modified. If no value is specified, and if the value is boolean, it will be inverted.

Errors The command fails if the specified option is unknown or if the specified value is invalid.

INFO

Syntax `INFO`

XML Syntax `<info/>`

Permission *READ*

Summary Shows global information.

HELP

Syntax `HELP ([command])`

XML Syntax `<help>([command])</help>`

Permission *NONE*

Summary If `command` is specified, information on the specific command is printed; otherwise, all commands are listed.

Errors The command fails if the specified command is unknown.

EXIT

Syntax	`EXIT`
XML Syntax	`<exit/>`
Permission	*NONE*
Summary	Exits the console mode.

QUIT

Syntax	`QUIT`
XML Syntax	`<quit/>`
Permission	*NONE*
Summary	Exits the console mode (alias of `EXIT`).

Changelog

Version 8.5

- Updated: Valid Names: allow dots (except as first and last character)

Version 8.4

- Updated: `CREATE INDEX`, `DROP INDEX`, `INFO INDEX`: token index added
- Updated: `INFO STORAGE`: Query argument removed, start/end added to XML syntax.
- Updated: `INFO INDEX`: Token index added; index `TAG` renamed to `ELEMNAME`; index `ATTNAME` renamed to `ATTRNAME`
- Updated: `OPTIMIZE`: adopt original index options

Version 8.2

- Removed: `CREATE EVENT`, `DROP EVENT` and `SHOW EVENTS` command

Version 8.0

- Updated: commands for User Management
- Updated: `OPEN`: path argument added
- Removed: `CS` command
- Added: `QUIT`

Version 7.9

- Added: `TEST` runs XQUnit tests.

Version 7.7

- Updated: syntax of valid names.

Version 7.5

- Added: `EXECUTE` executes a command script.
- Added: `INSPECT` performs integrity checks.
- Added: automatic detection of Command Scripts.
- Removed: `SHOW DATABASES`; information is also returned by `SHOW SESSIONS`.
- Removed: `OPEN`: path argument.

Version 7.3

- Added: XML Syntax added.

- Updated: `CHECK` can now be used to create empty databases.
- Updated: Names and paths in `OPEN` and `LIST` are now specified as separate arguments.

Version 7.2.1

- Updated: permissions for `GET` and `SET` changed from `READ` to `NONE`.

Version 7.2

- Updated: `CREATE INDEX`, `DROP INDEX` (`PATH` argument removed. Path summary is always available now and updated with `OPTIMIZE`).
- Updated: permissions for `REPO DELETE`, `REPO INSTALL` and `REPO LIST`.

Version 7.1

- Updated: `KILL` (killing sessions by specifying IP:port)

Version 7.0

- Added: `FLUSH`, `RETRIEVE`, `STORE`.
- Updated: `ADD`: simplified arguments.

References

[1] http://files.basex.org/xml/xmark.xml

Article Sources and Contributors

Commands *Source*: http://docs.basex.org/index.php?oldid=12594 *Contributors*: AW, Arve, CG, Dimitar, Dirk, Holu, Jan.vlcinsky, Jens Erat, Lukas.kircher, Michael

License