

MySQL Configuration Settings

Server options and system variables

MySQL maintains *more than 820 server configuration options and system variables*—a blizzard of conveniences, complexities and inconsistencies:

- *A cross-version master list* lives [here](#). (This also lists 275 *status variables* but fails to identify them as such. See also [Appendix C](#).)
- *Server Command-line Options*: The master list names about 450 command line settings. They are also listed [here](#). Running `mysqld -help` lists about two-thirds of of them, plus some other settings.
- *Option file settings*: Most server command options are also settable in a MySQL option file ([Chapter 3](#), Configuring MySQL)
- *System Variables* are accessible via SQL commands. When **dynamic**, they can also be SET via SQL. The master list for MySQL 6.0 names nearly 400. The [Dynamic System Variables page for 6.0](#) lists well over 200 (it grows with each version). SHOW VARIABLES lists many of those, plus nearly a hundred others which are not dynamic, i.e. they *cannot* be SET.
- Some variables are read-only, and some settings cannot be read at all.
- Some variables are *both* Server Command Options *and* Dynamic Variables. They tend to be spelled with dashes on the command line and in option files, but with underscores when SELECTed or SET!
- One variable can be set only in the server connection string; some can be set only by recompiling the server;
- Variables change significantly from MySQL release to release, as you might expect in a dynamic product; some vanish outright.
- Running `mysqladmin -uUSR -pPWD variables` lists more than 330 settings in version 6.0.10. Half of them are dynamic. Since 5.1.22 this query retrieves the same variables:

```
SELECT variable_name, variable_value
FROM information_schema.{global|session}_variables
[WHERE variable_name LIKE '...'] [ORDER BY variable_name];
```

Thus Table B-1, for quickly looking up how to set and read MySQL configuration variables.

GLOBAL system variables (*Type G* in Table B-1) persist while the server is up. LOCAL (SESSION) settings (*Type L* in Table B-1) die with the connection. The usual syntax for *retrieving* SELECTable variable values is ...

SELECT @@[GLOBAL|LOCAL|SESSION].variable_name;

but in some cases, for example `autocommit`, the leading @@ is optional.

Some system variables are dynamic (can be set with SQL), some not. There are two syntaxes for *SETTING* dynamic system variables:

SET [ONE-SHOT] [GLOBAL|SESSION|LOCAL] mysql_option=value[,...]

SET [ONE_SHOT] @@[GLOBAL.|SESSION.|LOCAL.]mysql_option=value[,...]

ONE_SHOT has effect until the next non-SET command; it is for `char*`, `coll*` and `time_zone` variables only.

SET ... in the *Syntax for Setting* column of Table B-1 means you can use either of the above SET syntaxes; **[SET ...]** means the variable can also be set in an option file or on the command line. But beware exceptions, e.g. in SET NAMES and SET PASSWORD, omit '='. Overall conventions in Table B-1 are:

- The *Syntax for Setting* column shows '#' for numeric values, '.' for string and date values, and follows these rules:

<i>If ...</i>	<i>then Syntax for Setting shows...</i>	<i>and these conventions apply...</i>
the variable can be SET with SQL ...	SET ... varName=value	If the variable is only GLOBAL or LOCAL, modifier is usually optional. But beware of SET syntax variation from variable to variable!
the variable can be set in an option file or on the server command-line ...	varName [=value]	On the command line, prefix with two hyphens. In an option file, don't. If there is an abbreviation, the <i>Abbr</i> column shows it.
the variable can be set by both the above methods...	[SET ...] varName [=value]	<i>Type</i> in Table B-1 indicates what [SET ...] accepts.
the variable is set on off by --[skip-]variablename ...	variablename skip-variablename	On the command line, preface with two hyphens. Not in an option file. Most skip-variablename settings are reported only by variablename.
the variable is controlled by CHANGE MASTER TO...	CHANGE MASTER TO ...	Use underscores in variable names
MySQL accepts either dashes or underscores in the name ...		Table B-1 shows underscores
the variable cannot be set...	Blank	

- The *Abbr* column shows the abbreviation, if any, for setting the variable on the server command line.
- The *Type* column shows G if the variable can be GLOBAL, L if it can be LOCAL | SESSION.
- The *Shown by Select* column indicates whether the variable is shown by SELECT [@@GLOBAL.|LOCAL.] ... syntax. SELECT ... without GLOBAL | LOCAL | SESSION, MySQL returns the LOCAL value if it exists, otherwise it returns the GLOBAL value.

- The *Shown by Show Variables* column indicates whether SHOW [GLOBAL | LOCAL] VARIABLES lists the variable.
- The *Shown by mysqld -help* column shows whether the command *mysqld --verbose --help* lists the variable.
- The *Shown by mysqladmin variables* column shows whether the command *mysqladmin variables* lists the variable.

Security-related variable names are shown in red, replication-related variable names in plum; obsolete settings are greyed out. Nobody can remember all this detail! We suggest you keep a printout near your computer.

To read the rest of this and other chapters, [buy a copy of the book](#)

[TOC](#) [Previous](#) [Next](#)
