git-update-ref

git-update-ref(1) ==========

NAME ---- git-update-ref - Update the object name stored in a ref safely

SYNOPSIS --------

git update-ref [-m <reason>] [--no-deref] -d <ref> [<old-oid>] git update-ref [-m <reason>] [--no-deref] [--create-reflog] <ref> <new-oid> [<old-oid>] git update-ref [-m <reason>] [--no-deref] --stdin [-z] [--batch-updates]

DESCRIPTION ----------- Given two arguments, stores the <new-oid> in the <ref>, possibly dereferencing the symbolic refs. E.g. `git` `update-ref` `HEAD` `<new-oid>` updates the current branch head to the new object.

Given three arguments, stores the <new-oid> in the <ref>, possibly dereferencing the symbolic refs, after verifying that the current value of the <ref> matches <old-oid>. E.g. `git` `update-ref` `refs/heads/master` `<new-oid>` `<old-oid>` updates the master branch head to <new-oid> only if its current value is <old-oid>. You can specify 40 "0" or an empty string as <old-oid> to make sure that the ref you are creating does not exist.

The final arguments are object names; this command without any options does not support updating a symbolic ref to point to another ref (see linkgit:git-symbolic-ref[1]). But `git` `update-ref` `--stdin` does have the `symref-*` commands so that regular refs and symbolic refs can be committed in the same transaction.

If --no-deref is given, <ref> itself is overwritten, rather than the result of following the symbolic pointers.

With `-d`, it deletes the named <ref> after verifying that it still contains <old-oid>.

With `--stdin`, update-ref reads instructions from standard input and performs all modifications together. Specify commands of the form:

```
update SP <ref> SP <new-oid> [SP <old-oid>] LF
create SP <ref> SP <new-oid> LF
delete SP <ref> [SP <old-oid>] LF
verify SP <ref> [SP <old-oid>] LF
symref-update SP <ref> SP <new-target> [SP (ref SP <old-target> | oid SP <old-oid>)] LF
symref-create SP <ref> SP <new-target> LF
symref-delete SP <ref> [SP <old-target>] LF
```

```
symref-verify SP <ref> [SP <old-target>] LF
option SP <opt> LF
start LF
prepare LF
commit LF
abort LF
```

With `--create-reflog`, update-ref will create a reflog for each ref even if one would not ordinarily be created.

With `--batch-updates`, update-ref executes the updates in a batch but allows individual updates to fail due to invalid or incorrect user input, applying only the successful updates. However, system-related errors—such as I/O failures or memory issues—will result in a full failure of all batched updates. Any failed updates will be reported in the following format:

```
rejected SP (<old-oid> | <old-target>) SP (<new-oid> | <new-target>) SP <rejection-reason> LF
```

Quote fields containing whitespace as if they were strings in C source code; i.e., surrounded by double-quotes and with backslash escapes. Use 40 "0" characters or the empty string to specify a zero value. To specify a missing value, omit the value and its preceding SP entirely.

Alternatively, use `-z` to specify in NUL-terminated format, without quoting:

```
update SP <ref> NUL <new-oid> NUL [<old-oid>] NUL
create SP <ref> NUL <new-oid> NUL
delete SP <ref> NUL [<old-oid>] NUL
verify SP <ref> NUL [<old-oid>] NUL
symref-update SP <ref> NUL <new-target> [NUL (ref NUL <old-target> | oid NUL <old-oid>)] NUL
symref-create SP <ref> NUL <new-target> NUL
symref-delete SP <ref> [NUL <old-target>] NUL
symref-verify SP <ref> [NUL <old-target>] NUL
option SP <opt> NUL
start NUL
prepare NUL
commit NUL
abort NUL
```

In this format, use 40 "0" to specify a zero value, and use the empty string to specify a missing value.

In either format, values can be specified in any form that Git recognizes as an object name. Commands in any other format or a repeated <ref> produce an error. Command meanings are:

update
Set <ref> to <new-oid> after verifying <old-oid>, if given. Specify a zero <new-oid> to ensure the ref does not exist after the update and/or a zero <old-oid> to make sure the ref does not exist before the update.

create
Create <ref> with <new-oid> after verifying that it does not exist. The given <new-oid> may not be zero.

delete
Delete <ref> after verifying that it exists with <old-oid>, if given. If given, <old-oid> may not be zero.

symref-update
Set <ref> to <new-target> after verifying <old-target> or <old-oid>, if given. Specify a zero <old-oid> to ensure that the ref does not exist before the update.

verify
Verify <ref> against <old-oid> but do not change it. If <old-oid> is zero or missing, the ref must not exist.

symref-create
Create symbolic ref <ref> with <new-target> after verifying that it does not exist.

symref-delete
Delete <ref> after verifying that it exists with <old-target>, if given.

symref-verify
Verify symbolic <ref> against <old-target> but do not change it. If <old-target> is missing, the ref must not exist. Can only be used in `no-deref` mode.

option
Modify the behavior of the next command naming a <ref>. The only valid option is `no-deref` to avoid dereferencing a symbolic ref.

start
Start a transaction. In contrast to a non-transactional session, a transaction will automatically abort if the session ends without an explicit commit. This command may create a new empty transaction when the current one has been committed or aborted already.

prepare
Prepare to commit the transaction. This will create lock files for all queued reference updates. If one reference could not be locked, the transaction will be aborted.

commit
Commit all reference updates queued for the transaction, ending the transaction.

abort
Abort the transaction, releasing all locks if the transaction is in prepared state.

If all <ref>s can be locked with matching <old-oid>s simultaneously, all modifications are performed. Otherwise, no modifications are performed. Note that while each individual <ref> is updated or deleted atomically, a concurrent reader may still see a subset of the modifications.

LOGGING UPDATES ---------- If config parameter "core.logAllRefUpdates" is true and the ref is one under "refs/heads/", "refs/remotes/", "refs/notes/", or a pseudoref like HEAD or ORIG_HEAD; or the file "$GIT_DIR/logs/<ref>" exists then `git update-ref` will append a line to the log file "$GIT_DIR/logs/<ref>" (dereferencing all symbolic refs before creating the log name) describing the change in ref value. Log lines are formatted as:

```
oldsha1 SP newsha1 SP committer LF
```

Where "oldsha1" is the 40 character hexadecimal value previously stored in <ref>, "newsha1" is the 40 character hexadecimal value of <new-oid> and "committer" is the committer’s name, email address and date in the standard Git committer ident format.

Optionally with -m:

```
oldsha1 SP newsha1 SP committer TAB message LF
```

Where all fields are as described above and "message" is the value supplied to the -m option.

An update will fail (without changing <ref>) if the current user is unable to create a new log file, append to the existing log file or does not have committer information available.

NOTES -----

Symbolic refs were initially implemented using symbolic links. This is now deprecated since not all filesystems support symbolic links.

This command follows **real** symlinks only if they start with "refs/": otherwise it will just try to read them and update them as a regular file (i.e. it will allow the filesystem to follow them, but will overwrite such a symlink to somewhere else with a regular filename).

SEE ALSO -------- linkgit:git-symbolic-ref[1]

GIT --- Part of the linkgit:git[1] suite

□□□ #1
□ □□□ □□□ 2026-10-31 20:13:26 UTC
□ □□□ □□□ 2026-10-31 20:13:26 UTC