NAME

git-check-ref-format - Ensures that a reference name is well formed

SYNOPSIS

git check-ref-format [--normalize]
[--[no-]allow-onelevel] [--refspec-pattern]
<refname>
git check-ref-format --branch <branchname-shorthand>

DESCRIPTION

Checks if a given *refname* is acceptable, and exits with a non-zero status if it is not.

A reference is used in Git to specify branches and tags. A branch head is stored in the refs/heads hierarchy, while a tag is stored in the refs/tags hierarchy of the ref namespace (typically in $GIT_DIR/refs/heads and $GIT_DIR/refs/tags directories or, as entries in file $GIT_DIR/packed-refs if refs are packed by git gc).

Git imposes the following rules on how references are named:

1. They can include slash / for hierarchical (directory) grouping, but no slash-separated component can begin with a dot . or end with the sequence .lock.
2. They must contain at least one /. This enforces the presence of a category like heads/, tags/ etc. but the actual names are not restricted. If the --allow-onelevel option is used, this rule is waived.
3. They cannot have two consecutive dots .. anywhere.
4. They cannot have ASCII control characters (i.e. bytes whose values are lower than 040, or 177 DEL), space, tilde ~, caret ^, or colon : anywhere.
5. They cannot have question-mark ?, asterisk *, or open bracket [anywhere. See the --refspec-pattern option below for an exception to this rule.
6. They cannot begin or end with a slash / or contain multiple consecutive slashes (see the --normalize option below for an exception to this rule)
7. They cannot end with a dot ..
8. They cannot contain a sequence @{.
9. They cannot be the single character @.
10. They cannot contain a .

These rules make it easy for shell script based tools to parse reference names, pathname expansion by the shell when a reference name is used unquoted (by mistake), and also avoids ambiguities in certain reference name expressions (see **gitrevisions(7)**):

1. A double-dot .. is often used as in ref1..ref2, and in some contexts this notation means ^ref1 ref2 (i.e. not in ref1 and in ref2).
2. A tilde ~ and caret ^ are used to introduce the postfix *nth parent* and *peel onion* operation.
3. A colon : is used as in srcref:dstref to mean use srcref's value and store it in dstref in fetch and push operations. It may also be used to select a specific object such as with *git cat-file*: git cat-file blob v1.3.3:refs.c.
4. at-open-brace @{ is used as a notation to access a reflog entry.

With the --branch option, it expands the "previous branch syntax" @{-n}. For example, @{-1} is a way to refer the last branch you were on. This option should be used by porcelains to accept this syntax anywhere a branch name is expected, so they can act as if you typed the branch name.

OPTIONS

--[no-]allow-onelevel
: Controls whether one-level refnames are accepted (i.e., refnames that do not contain multiple /-separated components). The default is --no-allow-onelevel.

--refspec-pattern
: Interpret <refname> as a reference name pattern for a refspec (as used with remote repositories). If this option is enabled, <refname> is allowed to contain a single * in place of a

one full pathname component (e.g., foo/*/bar but not foo/bar*).

--normalize
: Normalize *refname* by removing any leading slash (/) characters and collapsing runs of adjacent slashes between name components into a single slash. Iff the normalized refname is valid then print it to standard output and exit with a status of 0. (--print is a deprecated way to spell --normalize.)

EXAMPLES

- Print the name of the previous branch:

  $ git check-ref-format --branch @{-1}

- Determine the reference name to use for a new branch:

  $ ref=$(git check-ref-format --normalize refs/heads/$newbranch) ||
  die we do not like $newbranch as a branch name.

GIT

Part of the **git(1)** suite