## NAME

gitnamespaces - Git namespaces

## SYNOPSIS

GIT_NAMESPACE=<namespace> *git upload-pack*
GIT_NAMESPACE=<namespace> *git receive-pack*

## DESCRIPTION

Git supports dividing the refs of a single repository into multiple namespaces, each of which has its own branches, tags, and HEAD. Git can expose each namespace as an independent repository to pull from and push to, while sharing the object store, and exposing all the refs to operations such as **git-gc(1)**.

Storing multiple repositories as namespaces of a single repository avoids storing duplicate copies of the same objects, such as when storing multiple branches of the same source. The alternates mechanism provides similar support for avoiding duplicates, but alternates do not prevent duplication between new objects added to the repositories without ongoing maintenance, while namespaces do.

To specify a namespace, set the GIT_NAMESPACE environment variable to the namespace. For each ref namespace, Git stores the corresponding refs in a directory under refs/namespaces/. For example, GIT_NAMESPACE=foo will store refs under refs/namespaces/foo/. You can also specify namespaces via the --namespace option to **git(1)**.

Note that namespaces which include a / will expand to a hierarchy of namespaces; for example, GIT_NAMESPACE=foo/bar will store refs under refs/namespaces/foo/refs/namespaces/bar/. This makes paths in GIT_NAMESPACE behave hierarchically, so that cloning with GIT_NAMESPACE=foo/bar produces the same result as cloning with GIT_NAMESPACE=foo and cloning from that repo with GIT_NAMESPACE=bar. It also avoids ambiguity with strange namespace paths such as foo/refs/heads/, which could otherwise generate directory/file conflicts within the refs directory.

**git-upload-pack(1)** and **git-receive-pack(1)** rewrite the names of refs as specified by GIT_NAMESPACE. git-upload-pack and git-receive-pack will ignore all references outside the specified namespace.

The smart HTTP server, **git-http-backend(1)**, will pass GIT_NAMESPACE through to the backend programs; see **git-http-backend(1)** for sample configuration to expose repository namespaces as repositories.

For a simple local test, you can use **git-remote-ext(1)**:

git clone ext::git --namespace=foo %s /tmp/prefixed.git

## SECURITY

Anyone with access to any namespace within a repository can potentially access objects from any other namespace stored in the same repository. You can't directly say give me object ABCD if you don't have a ref to it, but you can do some other sneaky things like:

1. Claiming to push ABCD, at which point the server will optimize out the need for you to actually send it. Now you have a ref to ABCD and can fetch it (claiming not to have it, of course).
2. Requesting other refs, claiming that you have ABCD, at which point the server may generate deltas against ABCD.

None of this causes a problem if you only host public repositories, or if everyone who may read one namespace may also read everything in every other namespace (for instance, if everyone in an organization has read permission to every repository).