ProFTPD: Configuring `<Limits>`

ProFTPD's `<Limit>` configuration sections allow for powerful fine-grained control over who is allowed to use which FTP commands. This power comes at the price of complexity, however. This document describes some of the things to keep in mind when writing `<Limit>` sections.

Precedence
Perhaps the hardest part of using `<Limit>` is understanding its rules of precedence, which dictate which `<Limit>`'s restrictions apply when. Precedence is discussed in the directive documentation, and will be mentioned here. First, there are three types of parameters in a `<Limit>` directive: "raw" FTP commands, FTP command groups, and the `ALL` keyword.

"Raw" FTP commands are listed here, including the RFC-mandated X-variant FTP commands, which are often missing from a thorough `<Limit>` configuration.

The FTP command groups are:

- ALL
  Covering: all FTP commands (but **not** `LOGIN`)
- DIRS
  Covering: CDUP, CWD, LIST, MDTM, MLSD, MLST, NLST, PWD, RNFR, STAT, XCUP, XCWD, XPWD
- LOGIN
  Covering: client logins
- READ
  Covering: RETR, SIZE
- WRITE
  Covering: APPE, DELE, MKD, RMD, RNTO, STOR, STOU, XMKD, XRMD

`<Limit>`s that use "raw" FTP commands have the highest precedence, followed by `<Limit>`s that use the command groups, and, having the lowest precedence, the `ALL` keyword. If a `<Limit>` has both "raw" commands and command groups, then it boils down to the order of appearance of `<Limit>` sections in `proftpd.conf` that use the "raw" command in question.

`SITE` Commands
To apply a `<Limit>` to a `SITE` command, combine "SITE" and the command (*e.g.* "CHMOD") by an underscore ("_"), like so:

```
  <Limit SITE_command>
```

Thus, in order to place a limit on `SITE CHMOD`, one would have:

```
<Limit SITE_CHMOD>
  DenyAll
</Limit>
```

Inheritance

Most `<Limit>` sections appear within `<Directory>` sections in `proftpd.conf`. This means that, like the other `<Directory>` configuration effects, the `<Limit>`s will be inherited by all subdirectories that appear in the `<Directory>` path, unless explicitly overridden by a "closer" `<Limit>` section. This means that one could configure a `<Limit>` section denying all FTP commands for all directories, and then explicitly allow the `READ` or `WRITE` FTP command groups in appropriate subdirectories (*e.g.* `pub/` or `incoming/` directories).

Using `AllowUser` and `DenyUser`

There is a catch to using the `AllowUser` configuration directive that causes confusion, primarily when a single `AllowUser` directive is being used to allow access to some FTP commands only to certain users. ProFTPD uses the same function for parsing the `AllowUser` and `AllowGroup` (and other) directives. This function parses the list of names for such directives as a Boolean AND list, which means that each name on the list must evaluate to TRUE (must match) for the current user in order for the directive to apply. For `AllowGroup`, this makes sense, and allows a great deal of flexibility. However, it does not make sense for `AllowUser`, because a user may not be multiple users at the same time. This is a known issue, and a proper, thorough solution is being developed. In the meantime, however, there is a workaround for allowing multiple users via the `AllowUser` directive. Rather than listing the users using a single `AllowUser`, using a separate `AllowUser` for each user. For example, instead of:

```
AllowUser bob,dave,wendy
```

try using:

```
AllowUser bob
AllowUser dave
AllowUser wendy
```

All of this applies to the `DenyUser` directive as well.

Another important item to keep in mind is that the names used in `<Limit>` sections, *e.g.* using `AllowUser`, `DenyUser`, `AllowGroup`, and `DenyGroup`, are **not** resolved to an ID and then applied; the limits are applied only to the names. Why is this important? Consider the case where the site is using virtual users, where two different user names are assigned the same UID. Different limits can be applied to each name separately. Do not assume that the limits are applied to the underlying IDs.

Using `Order`

One thing that sometimes trips up some administrators is the difference between ProFTPD's and Apache's `Order` configuration directives. For Apache, an `Order` of "Allow,Deny" means that access is **denied** by default, unless an `Allow` directive explicitly allows access; an `Order` of "Deny,Allow" means that access is **allowed** by default, unless a `Deny` directive explicitly denies access. This is different from ProFTPD, where an `Order` of "Allow,Deny" **allows** access by default, unless denied by a `Deny` directive; "Deny,Allow" **denies** access by default, unless explicitly granted by an `Allow` directive. The developers of ProFTPD felt their interpretation to be the more "common sense" interpretation, even though it does not match Apache's interpretation.

Examples

Here are examples to help illustrate the use of `<Limit>`. First, a common configuration: an upload-only directory.

```
<Directory /path/to/uploads>
  <Limit ALL>
    DenyAll
  </Limit>

  <Limit CDUP CWD PWD XCWD XCUP>
    AllowAll
  </Limit>

  <Limit STOR STOU>
    AllowAll
  </Limit>
</Directory>
```

The first `<Limit ALL>` section blocks use of *all* FTP commands within the `/path/to/uploads` directory. Having denied use of all commands, we then proceed to define which commands *can* be used. The `CDUP` and `CWD` commands (and their X variants) should be allowed so that clients can change into and out of the directory. Next, `STOR` and `STOU` are allowed, so that clients can actually upload files into the directory (assuming that the filesystem permissions allow for the client to write files in the directory as well). The `WRITE` command group might have been used, but that also allows things like creating and deleting subdirectories, which is usually not wanted in an upload-only configuration.

This next example shows a "blind" directory, where clients can upload and download files from the directory, but they cannot see what is in the directory:

```
<Directory /path/to/dir>
  <Limit LIST NLST MLSD MLST STAT>
    DenyAll
  </Limit>
</Directory>
```

That's it. By default, all commands are allowed in a directory. By blocking the FTP commands used to list a directory's contents (*i.e.* `LIST`, `MLSD`, `MLST`, and `NLST`), we have effectively blocked the client from seeing anything in the directory. Not many clients use the `STAT` command, but it also needs to be limited, as it can return information about files in a directory as well.

Cautious system administrators may want only a few select system users to be able to connect to their `proftpd` server--all other users are to be denied access. The `LOGIN` command group is designed for just this scenario:

```
<Limit LOGIN>
  AllowUser barb
  AllowUser dave
  AllowGroup ftpuser
  DenyAll
</Limit>
```

This allows the users `barb` and `dave`, as well as any user in the `ftpuser` group, to login. All other users will be denied.

What if a site wished to allow **only** anonymous access? This would be configured using the `LOGIN` command group, as above:

```
<Limit LOGIN>
  DenyAll
</Limit>

<Anonymous ~ftp>
  <Limit LOGIN>
    AllowAll
  </Limit>
  ...
</Anonymous>
```

The `<Limit>` section outside of the `<Anonymous>` section denies logins to everyone. However, the `<Anonymous>` section has a `<Limit>` that allows everyone to login; anonymous logins are allowed, and non-anonymous logins are denied.

Another related question often asked is "How can I limit a user to only being able to login from a specific range of IP addresses?" The `<Limit LOGIN>` can be used, in conjunction with the mod_ifsession module and a Class, to configure this:

```
<Class friends>
  From 1.2.3.4/8
</Class>

<IfUser dave>
  <Limit LOGIN>
    AllowClass friends
    DenyAll
  </Limit>
</IfUser>
```

Or if you want to have a specific IP address, rather than a range, you can do this without classes (but still requiring `mod_ifsession`):

```
<IfUser dave>
  <Limit LOGIN>
    Deny from 1.2.3.4
  </Limit>
</IfUser>
```

Note that the same effect can be achieved by using the mod_wrap2 module to configure user-specific allow/deny files.

One issue that you should avoid is having multiple different `<Limit LOGIN>` sections in your config *for the same vhost*. Consider a config like this:

```
<VirtualHost 1.2.3.4>
  ...
  <Limit LOGIN>
    Order allow, deny
    Allow from 192.168.0.0/16
    DenyAll
  </Limit>
  ...
  <Limit LOGIN>
    Order deny, allow
    Deny from 192.168.0.0/16
  </Limit>
  ...
</VirtualHost>
```

The two `<Limit LOGIN>` sections conflict; which one will `proftpd` actually use for deciding whether to allow a connection to that `<VirtualHost>`? Answer: the **last** `<Limit LOGIN>` section defined. To avoid confusion, then, it is best to consolidate all of your `<Limit LOGIN>` rules into a single section for a given vhost.

In Apache, it is possible to configure password-protected directories. Some sysadmins attempt to configure `proftpd` similarly, by unsuccessfully attempting something like this in the `proftpd.conf` file:

```
<Directory /some/path>
  <Limit LOGIN>
    DenyUser foo
  </Limit>
</Directory>
```

The above will **not** work. FTP clients (unlike HTTP clients) login to the *server*, not into specific directories.

One situation that often arises is one where the administrator would like to give users the ability to upload and dowload files from a given directory, but not to be able to delete files from that directory. This cannot be accomplished using normal Unix filesystem permissions, for if a user has write permission on a directory (necessary for uploading files to that directory) they also have delete permissions. In Unix, a directory file serves as a sort of "table of contents", tracking the files in the directory. Adding or removing a file are thus changes on the directory file, and do not involve checking the permissions on the file being added or removed. This is also how a non-root user can delete files that are owned by root and only have user-write permissions. So how then can a site be configured to allow writes but not deletes? By using a configuration similar to the following:

```
<Directory /path/to/dir>
  <Limit DELE>
    AllowUser ftpadm
    DenyAll
  </Limit>
</Directory>
```

This will allow the user `ftpadm` to delete files in the `/path/to/dir`, but no other users.

The FTP protocol has two types of data transfers: active and passive. In some configurations, only one type of transfer is allowed by the network (*e.g.* active transfers should be denied because clients are sending the wrong IP addresses). The ability to place a `<Limit>` on the FTP commands response for active and passive data transfers was added to ProFTPD in 1.2.10rc1. If you are using that version or later, you can use the following to block active transfers:

```
<Limit EPRT PORT>
  DenyAll
</Limit>
```

Or, conversely, to block passive data transfers:

```
<Limit EPSV PASV>
  DenyAll
</Limit>
```

Another common question is: "How can I create a read-only account using `<Limit>` sections"? Here's how:

```
# Assumes that the user is chrooted into their home directory
<Directory ~user>
  <Limit CWD PWD DIRS READ>
    AllowUser user
  </Limit>

  <Limit ALL>
    DenyUser user
  </Limit>
</Directory>
```

What if you want to prevent a certain directory from being deleted, but you *do* want to allow sub-directories in that directory to be deletable? Using two `<Directory>` sections with `<Limit>` sections, you can do this, *e.g.*:

```
<Directory /path/to/dir>
  <Limit RMD XRMD>
    DenyAll
  </Limit>
</Directory>

<Directory /path/to/dir/*>
  <Limit RMD XRMD>
    AllowAll
  </Limit<
>/Directory>
```

Note the trailing "/*" suffix in the second `<Directory>` section; this means that the second `<Directory>` section configuration applies to the sub-directories, but *not* to the parent directory itself (which is covered by the first `<Directory>` section).

What if you want to make sure the directory cannot be renamed, in addition to ensuring that it cannot be deleted? Simply include the `RNFR` and `RNTO` FTP commands in the list of denied commands, *e.g.*:

```
<Directory /path/to/dir>
  <Limit RMD RNFR RNTO XRMD>
    DenyAll
  </Limit>
</Directory>
```