NAME

nfs4_acl − NFSv4 Access Control Lists

DESCRIPTION

An ACL is a list of permissions associated with a file or directory and consists of one or more Access Control Entries (ACEs). NFSv4 ACLs provide finer granularity than typical POSIX read/write/execute permissions and are similar to CIFS ACLs.

A sample NFSv4 file ACL might look like the following (see the **ACL FORMAT** section for detailed information):

```
A::OWNER@:rwatTnNcCy
A::alice@nfsdomain.org:rxtncy
A::bob@nfsdomain.org:rwadtTnNcCy
A:g:GROUP@:rtncy
D:g:GROUP@:waxTC
A::EVERYONE@:rtncy
D::EVERYONE@:waxTC
```

Some observations:

- In the example output above, the user ‘alice@nfsdomain.org’ has the equivalent of "read" and "execute" permissions, ‘bob@nfsdomain.org’ has "read" and "write", and both ‘GROUP@’ and ‘EVERYONE@’ have "read".
- NFSv4 ACLs are "default-deny"; that is, if a permission is not explicitly granted by an Allow ACE, it is denied. Because of this, the two Deny ACEs above are superfluous and could be excluded by the server. See the **A WARNING ABOUT DENY ACES** section for more information.
- NFSv4 servers may return an ACL slightly different than one you set. For example, a server that always allows reading the attributes of a file may silently turn on the read-attributes *permission*, and a server that does not support separate write-data and append-data *permissions*, e.g., may choose to turn off both if you set only one. In extreme cases the server may also reorder or combine ACEs. As a general rule, however, servers will attempt to ensure that the ACLs they return are no more permissive than the ones you set.

ACL FORMAT

An NFSv4 ACL is written as an *acl_spec*, which is a comma- or whitespace-delimited string consisting of one or more *ace_specs*. A single NFSv4 ACE is written as an *ace_spec*, which is a colon-delimited, 4-field string in the following format:

type:*flags*:*principal*:*permissions*

ACE TYPES:

There are four *types* of ACEs, each represented by a single character. An ACE must have exactly one *type*.

A Allow - allow *principal* to perform actions requiring *permissions*.

D Deny - prevent *principal* from performing actions requiring *permissions*.

U Audit - log any attempted access by *principal* which requires *permissions*. Requires one or both of the successful-access and failed-access *flags*. System-dependent; not supported by all servers.

L Alarm - generate a system alarm at any attempted access by *principal* which requires *permissions*. Requires one or both of the successful-access and failed-access *flags*. System-dependent; not supported by all servers.

ACE FLAGS:

There are three kinds of ACE *flags*: group, inheritance, and administrative. An Allow or Deny ACE may contain zero or more *flags*, while an Audit or Alarm ACE must contain at least one of the successful-access and failed-access *flags*.

Note that ACEs are inherited from the parent directory's ACL at the time a file or subdirectory is created. Accordingly, inheritance flags can be used only in ACEs in a directory's ACL (and are therefore stripped from inherited ACEs in a new file's ACL). Please see the **INHERITANCE FLAGS COMMENTARY** section for more information.

GROUP FLAG - can be used in any ACE

g group - indicates that *principal* represents a group instead of a user.

INHERITANCE FLAGS - can be used in any directory ACE

d directory-inherit - newly-created subdirectories will inherit the ACE.

f file-inherit - newly-created files will inherit the ACE, minus its inheritance *flags*. Newly-created subdirectories will inherit the ACE; if directory-inherit is not also specified in the parent ACE, inherit-only will be added to the inherited ACE.

n no-propagate-inherit - newly-created subdirectories will inherit the ACE, minus its inheritance *flags*.

i inherit-only - the ACE is not considered in permissions checks, but it is heritable; however, the inherit-only *flag* is stripped from inherited ACEs.

ADMINISTRATIVE FLAGS - can be used in **Audit** and **Alarm** ACEs

S successful-access - trigger an alarm/audit when *principal* is allowed to perform an action covered by *permissions*.

F failed-access - trigger an alarm/audit when *principal* is prevented from performing an action covered by *permissions*.

ACE PRINCIPALS:

A *principal* is either a named user (e.g., 'myuser@nfsdomain.org') or group (provided the group *flag* is also set), or one of three special *principals*: 'OWNER@', 'GROUP@', and 'EVERYONE@', which are, respectively, analogous to the POSIX user/group/other distinctions used in, e.g., **chmod**(1).

ACE PERMISSIONS:

There are a variety of different ACE *permissions* (13 for files, 14 for directories), each represented by a single character. An ACE should have one or more of the following *permissions* specified:

r read-data (files) / list-directory (directories)

w write-data (files) / create-file (directories)

a append-data (files) / create-subdirectory (directories)

x execute (files) / change-directory (directories)

d delete - delete the file/directory. Some servers will allow a delete to occur if either this *permission* is set in the file/directory or if the delete-child *permission* is set in its parent direcory.

D delete-child - remove a file or subdirectory from within the given directory (directories only)

t read-attributes - read the attributes of the file/directory.

T write-attributes - write the attributes of the file/directory.

n read-named-attributes - read the named attributes of the file/directory.

N write-named-attributes - write the named attributes of the file/directory.

c read-ACL - read the file/directory NFSv4 ACL.

C write-ACL - write the file/directory NFSv4 ACL.

o write-owner - change ownership of the file/directory.

y synchronize - allow clients to use synchronous I/O with the server.

INHERITANCE FLAGS COMMENTARY

Inheritance *flags* can be divided into two categories: "primary" (file-inherit and directory-inherit); and "secondary" (no-propagate-inherit and inherit-only), which are significant only insofar as they affect the two "primary" *flags*.

The no-propagate-inherit and inherit-only *flags* can be tricky to remember: the former determines whether or not a new child directory's inherited ACE is itself heritable by a grandchild subdirectory; the latter determines whether or not a heritable ACE affects the parent directory itself (in addition to being heritable). They can be used in-tandem.

When a subdirectory inherits an ACE from its parent directory's ACL, this can happen in one of two different ways, depending on the server implementation:

- In the simple case, that exact same ACE is set in the subdirectory's ACL.
- In the other case, two different ACEs will instead be set in the subdirectory's ACL: one with all inheritance *flags* removed, and one with the inherit-only *flag* added. The former is the "effective" inherited ACE (used in the subdirectory's own permissions checks); the latter is the "heritable" inherited ACE (when the subdirectory has directories created within it, they inherit it). This approach makes it easier to modify access rights to the subdirectory itself without modifying its heritable ACEs.

A WARNING ABOUT DENY ACES

Deny ACEs should be avoided whenever possible. Although they are a valid part of NFSv4 ACLs, Deny ACEs can be confusing and complicated. This stems primarily from the fact that, unlike POSIX ACLs and CIFS ACLs, the ordering of ACEs within NFSv4 ACLs affects how they are evaluated.

First, it is important to note that (despite some unfortunate ambiguity in *RFC3530*) NFSv4 ACLs are "default-deny" in practice. That is, if a *permission* is not explicitly granted, it is denied.

In general, when a *principal* is attempting to perform an action over NFSv4 which requires one or more *permissions*, an access check is performed. The NFSv4 ACL (assuming one is present) is evaluated ACE-by-ACE until every one of those *permissions* has been addressed, or until the end of the ACL is reached. If every requisite *permission* was granted by Allow ACEs and was not forbidden by Deny ACEs (see next paragraph), the action is allowed to proceed. Otherwise, the action is forbidden.

Note that each requisite *permission* is only addressed once -- that is, after a *permission* has been explicitly Allowed or Denied once during an access check, any subsequent ACEs in the ACL which affect that *permission* are no longer considered. This often introduces problematic ordering issues when Deny ACEs are present.

Additionally, in some cases Group-Deny ACEs can be difficult (if not impossible) to enforce, since a server might not know about all of a given *principal*'s memberships in remote groups, e.g.

Because NFSv4 ACLs are "default-deny", the use of Deny ACEs can (and should) be avoided entirely in most cases.

AUTHORS

Tools for viewing and manipulating NFSv4 ACLs, **nfs4_getfacl** and **nfs4_setfacl**, were written by people at CITI, the Center for Information Technology Integration (*http://www.citi.umich.edu*). This manpage was written by David Richter and J. Bruce Fields.

CONTACT

Please send bug reports, feature requests, and comments to <*nfsv4@linux-nfs.org*>.

SEE ALSO

nfs4_getfacl(1), **nfs4_setacl**(1), *RFC3530* (NFSv4.0), NFSv4.1 Minor Version Draft.