Internet Engineering Task Force (IETF)
Request for Comments: 7531
Category: Standards Track
ISSN: 2070-1721

T. Haynes, Ed.
Primary Data
D. Noveck, Ed.
Dell

March 2015

Network File System (NFS) Version 4 External Data Representation Standard (XDR) Description

Abstract

The Network File System (NFS) version 4 protocol is a distributed file system protocol that owes its heritage to NFS protocol version 2 (RFC 1094) and version 3 (RFC 1813). Unlike earlier versions, the NFS version 4 protocol supports traditional file access while integrating support for file locking and the MOUNT protocol. In addition, support for strong security (and its negotiation), COMPOUND operations, client caching, and internationalization has been added. Of course, attention has been applied to making NFS version 4 operate well in an Internet environment.

RFC 7530 formally obsoletes RFC 3530. This document, together with RFC 7530, replaces RFC 3530 as the definition of the NFS version 4 protocol.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7531.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction 3
2. XDR Description of NFSv4.0 3
3. Security Considerations 39
4. Normative References 39
Acknowledgments 39
Authors' Addresses 39

1. Introduction

This document contains the External Data Representation (XDR) [RFC4506] description of the NFSv4.0 protocol [RFC7530].

2. XDR Description of NFSv4.0

The XDR description is provided in this document in a way that makes it simple for the reader to extract it into a form that is ready to compile. The reader can feed this document in the following shell script to produce the machine-readable XDR description of NFSv4.0:

```
#!/bin/sh
grep "^  *///" | sed 's?^  */// ??' | sed 's?^  *///$??'
```

That is, if the above script is stored in a file called "extract.sh", and this document is in a file called "spec.txt", then the reader can do:

```
sh extract.sh < spec.txt > nfs4_prot.x
```

The effect of the script is to remove leading white space from each line, plus a sentinel sequence of "///".

The XDR description, with the sentinel sequence, follows:

```
/// /*
///  * This file was machine generated for [RFC7530].
///  *
///  * Last updated Tue Mar 10 11:51:21 PDT 2015.
///  */
///
/// /*
///  * Copyright (c) 2015 IETF Trust and the persons identified
///  * as authors of the code.  All rights reserved.
///  *
///  * Redistribution and use in source and binary forms, with
///  * or without modification, are permitted provided that the
///  * following conditions are met:
///  *
///  * - Redistributions of source code must retain the above
///  *   copyright notice, this list of conditions and the
///  *   following disclaimer.
///  *
///  * - Redistributions in binary form must reproduce the above
///  *   copyright notice, this list of conditions and the
///  *   following disclaimer in the documentation and/or other
///  *   materials provided with the distribution.
///  *
///  * - Neither the name of Internet Society, IETF or IETF
///  *   Trust, nor the names of specific contributors, may be
///  *   used to endorse or promote products derived from this
///  *   software without specific prior written permission.
///  *
///  *   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS
///  *   AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
///  *   WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
///  *   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
///  *   FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO
///  *   EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
///  *   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
///  *   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
///  *   NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
///  *   SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
///  *   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
///  *   LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
///  *   OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
///  *   IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
///  *   ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
///  */
///
```

```
/// /*
///  * This code was derived from RFC 7531.
///  */
///
/// /*
///  *      nfs4_prot.x
///  *
///  */
///
/// /*
///  * Basic typedefs for RFC 1832 data type definitions
///  */
/// /*
///  * typedef int          int32_t;
///  * typedef unsigned int uint32_t;
///  * typedef hyper                int64_t;
///  * typedef unsigned hyper       uint64_t;
///  */
///
/// /*
///  * Sizes
///  */
/// const NFS4_FHSIZE              = 128;
/// const NFS4_VERIFIER_SIZE       = 8;
/// const NFS4_OTHER_SIZE          = 12;
/// const NFS4_OPAQUE_LIMIT        = 1024;
///
/// const NFS4_INT64_MAX           = 0x7fffffffffffffff;
/// const NFS4_UINT64_MAX          = 0xffffffffffffffff;
/// const NFS4_INT32_MAX           = 0x7fffffff;
/// const NFS4_UINT32_MAX          = 0xffffffff;
///
///
/// /*
///  * File types
///  */
/// enum nfs_ftype4 {
///         NF4REG = 1,     /* Regular File */
///         NF4DIR = 2,     /* Directory */
///         NF4BLK = 3,     /* Special File - block device */
///         NF4CHR = 4,     /* Special File - character device */
///         NF4LNK = 5,     /* Symbolic Link */
///         NF4SOCK = 6,    /* Special File - socket */
```

```
///         NF4FIFO = 7,    /* Special File - fifo */
///         NF4ATTRDIR
///                 = 8,    /* Attribute Directory */
///         NF4NAMEDATTR
///                 = 9     /* Named Attribute */
/// };
///
/// /*
///  * Error status
///  */
/// enum nfsstat4 {
///  NFS4_OK                = 0,    /* everything is okay       */
///  NFS4ERR_PERM           = 1,    /* caller not privileged    */
///  NFS4ERR_NOENT          = 2,    /* no such file/directory   */
///  NFS4ERR_IO             = 5,    /* hard I/O error           */
///  NFS4ERR_NXIO           = 6,    /* no such device           */
///  NFS4ERR_ACCESS         = 13,   /* access denied            */
///  NFS4ERR_EXIST          = 17,   /* file already exists      */
///  NFS4ERR_XDEV           = 18,   /* different file systems   */
///  /* Unused/reserved        19 */
///  NFS4ERR_NOTDIR         = 20,   /* should be a directory    */
///  NFS4ERR_ISDIR          = 21,   /* should not be directory  */
///  NFS4ERR_INVAL          = 22,   /* invalid argument         */
///  NFS4ERR_FBIG           = 27,   /* file exceeds server max  */
///  NFS4ERR_NOSPC          = 28,   /* no space on file system  */
///  NFS4ERR_ROFS           = 30,   /* read-only file system    */
///  NFS4ERR_MLINK          = 31,   /* too many hard links      */
///  NFS4ERR_NAMETOOLONG    = 63,   /* name exceeds server max  */
///  NFS4ERR_NOTEMPTY       = 66,   /* directory not empty      */
///  NFS4ERR_DQUOT          = 69,   /* hard quota limit reached */
///  NFS4ERR_STALE          = 70,   /* file no longer exists    */
///  NFS4ERR_BADHANDLE      = 10001,/* Illegal filehandle       */
///  NFS4ERR_BAD_COOKIE     = 10003,/* READDIR cookie is stale  */
///  NFS4ERR_NOTSUPP        = 10004,/* operation not supported  */
///  NFS4ERR_TOOSMALL       = 10005,/* response limit exceeded  */
///  NFS4ERR_SERVERFAULT    = 10006,/* undefined server error   */
///  NFS4ERR_BADTYPE        = 10007,/* type invalid for CREATE  */
///  NFS4ERR_DELAY          = 10008,/* file "busy" - retry      */
///  NFS4ERR_SAME           = 10009,/* nverify says attrs same  */
///  NFS4ERR_DENIED         = 10010,/* lock unavailable         */
///  NFS4ERR_EXPIRED        = 10011,/* lock lease expired       */
///  NFS4ERR_LOCKED         = 10012,/* I/O failed due to lock   */
///  NFS4ERR_GRACE          = 10013,/* in grace period          */
///  NFS4ERR_FHEXPIRED      = 10014,/* filehandle expired       */
///  NFS4ERR_SHARE_DENIED   = 10015,/* share reserve denied     */
///  NFS4ERR_WRONGSEC       = 10016,/* wrong security flavor    */
///  NFS4ERR_CLID_INUSE     = 10017,/* clientid in use          */
///  NFS4ERR_RESOURCE       = 10018,/* resource exhaustion      */
```

```
///  NFS4ERR_MOVED          = 10019,/* file system relocated    */
///  NFS4ERR_NOFILEHANDLE   = 10020,/* current FH is not set    */
///  NFS4ERR_MINOR_VERS_MISMATCH = 10021,/* minor vers not supp */
///  NFS4ERR_STALE_CLIENTID = 10022,/* server has rebooted      */
///  NFS4ERR_STALE_STATEID  = 10023,/* server has rebooted      */
///  NFS4ERR_OLD_STATEID    = 10024,/* state is out of sync     */
///  NFS4ERR_BAD_STATEID    = 10025,/* incorrect stateid        */
///  NFS4ERR_BAD_SEQID      = 10026,/* request is out of seq.   */
///  NFS4ERR_NOT_SAME       = 10027,/* verify - attrs not same  */
///  NFS4ERR_LOCK_RANGE     = 10028,/* lock range not supported */
///  NFS4ERR_SYMLINK        = 10029,/* should be file/directory */
///  NFS4ERR_RESTOREFH      = 10030,/* no saved filehandle      */
///  NFS4ERR_LEASE_MOVED    = 10031,/* some file system moved   */
///  NFS4ERR_ATTRNOTSUPP    = 10032,/* recommended attr not sup */
///  NFS4ERR_NO_GRACE       = 10033,/* reclaim outside of grace */
///  NFS4ERR_RECLAIM_BAD    = 10034,/* reclaim error at server  */
///  NFS4ERR_RECLAIM_CONFLICT = 10035,/* conflict on reclaim    */
///  NFS4ERR_BADXDR         = 10036,/* XDR decode failed        */
///  NFS4ERR_LOCKS_HELD     = 10037,/* file locks held at CLOSE */
///  NFS4ERR_OPENMODE       = 10038,/* conflict in OPEN and I/O */
///  NFS4ERR_BADOWNER       = 10039,/* owner translation bad    */
///  NFS4ERR_BADCHAR        = 10040,/* UTF-8 char not supported */
///  NFS4ERR_BADNAME        = 10041,/* name not supported       */
///  NFS4ERR_BAD_RANGE      = 10042,/* lock range not supported */
///  NFS4ERR_LOCK_NOTSUPP   = 10043,/* no atomic up/downgrade   */
///  NFS4ERR_OP_ILLEGAL     = 10044,/* undefined operation      */
///  NFS4ERR_DEADLOCK       = 10045,/* file locking deadlock    */
///  NFS4ERR_FILE_OPEN      = 10046,/* open file blocks op.     */
///  NFS4ERR_ADMIN_REVOKED  = 10047,/* lock-owner state revoked */
///  NFS4ERR_CB_PATH_DOWN   = 10048 /* callback path down       */
/// };
///
/// /*
///  * Basic data types
///  */
/// typedef opaque  attrlist4<>;
/// typedef uint32_t        bitmap4<>;
/// typedef uint64_t        changeid4;
/// typedef uint64_t        clientid4;
/// typedef uint32_t        count4;
/// typedef uint64_t        length4;
/// typedef uint32_t        mode4;
/// typedef uint64_t        nfs_cookie4;
/// typedef opaque  nfs_fh4<NFS4_FHSIZE>;
/// typedef uint32_t        nfs_lease4;
/// typedef uint64_t        offset4;
/// typedef uint32_t        qop4;
/// typedef opaque  sec_oid4<>;
```

```
/// typedef uint32_t        seqid4;
/// typedef opaque  utf8string<>;
/// typedef utf8string      utf8str_cis;
/// typedef utf8string      utf8str_cs;
/// typedef utf8string      utf8str_mixed;
/// typedef utf8str_cs      component4;
/// typedef opaque  linktext4<>;
/// typedef utf8string      ascii_REQUIRED4;
/// typedef component4      pathname4<>;
/// typedef uint64_t        nfs_lockid4;
/// typedef opaque  verifier4[NFS4_VERIFIER_SIZE];
///
///
/// /*
///  * Timeval
///  */
/// struct nfstime4 {
///         int64_t         seconds;
///         uint32_t        nseconds;
/// };
///
/// enum time_how4 {
///         SET_TO_SERVER_TIME4 = 0,
///         SET_TO_CLIENT_TIME4 = 1
/// };
///
/// union settime4 switch (time_how4 set_it) {
///  case SET_TO_CLIENT_TIME4:
///          nfstime4       time;
///  default:
///          void;
/// };
///
///
/// /*
///  * File attribute definitions
///  */
///
/// /*
///  * FSID structure for major/minor
///  */
/// struct fsid4 {
///         uint64_t        major;
///         uint64_t        minor;
/// };
///
///
```

```
/// /*
///  * File system locations attribute for relocation/migration
///  */
/// struct fs_location4 {
///         utf8str_cis     server<>;
///         pathname4       rootpath;
/// };
///
/// struct fs_locations4 {
///         pathname4       fs_root;
///         fs_location4    locations<>;
/// };
///
///
/// /*
///  * Various Access Control Entry definitions
///  */
///
/// /*
///  * Mask that indicates which Access Control Entries
///  * are supported.  Values for the fattr4_aclsupport attribute.
///  */
/// const ACL4_SUPPORT_ALLOW_ACL    = 0x00000001;
/// const ACL4_SUPPORT_DENY_ACL     = 0x00000002;
/// const ACL4_SUPPORT_AUDIT_ACL    = 0x00000004;
/// const ACL4_SUPPORT_ALARM_ACL    = 0x00000008;
///
///
/// typedef uint32_t        acetype4;
///
///
/// /*
///  * acetype4 values; others can be added as needed.
///  */
/// const ACE4_ACCESS_ALLOWED_ACE_TYPE      = 0x00000000;
/// const ACE4_ACCESS_DENIED_ACE_TYPE       = 0x00000001;
/// const ACE4_SYSTEM_AUDIT_ACE_TYPE        = 0x00000002;
/// const ACE4_SYSTEM_ALARM_ACE_TYPE        = 0x00000003;
///
///
///
/// /*
///  * ACE flag
///  */
/// typedef uint32_t        aceflag4;
///
```

```
///
/// /*
///  * ACE flag values
///  */
/// const ACE4_FILE_INHERIT_ACE             = 0x00000001;
/// const ACE4_DIRECTORY_INHERIT_ACE        = 0x00000002;
/// const ACE4_NO_PROPAGATE_INHERIT_ACE     = 0x00000004;
/// const ACE4_INHERIT_ONLY_ACE             = 0x00000008;
/// const ACE4_SUCCESSFUL_ACCESS_ACE_FLAG   = 0x00000010;
/// const ACE4_FAILED_ACCESS_ACE_FLAG       = 0x00000020;
/// const ACE4_IDENTIFIER_GROUP             = 0x00000040;
///
///
///
/// /*
///  * ACE mask
///  */
/// typedef uint32_t        acemask4;
///
///
/// /*
///  * ACE mask values
///  */
/// const ACE4_READ_DATA            = 0x00000001;
/// const ACE4_LIST_DIRECTORY       = 0x00000001;
/// const ACE4_WRITE_DATA           = 0x00000002;
/// const ACE4_ADD_FILE             = 0x00000002;
/// const ACE4_APPEND_DATA          = 0x00000004;
/// const ACE4_ADD_SUBDIRECTORY     = 0x00000004;
/// const ACE4_READ_NAMED_ATTRS     = 0x00000008;
/// const ACE4_WRITE_NAMED_ATTRS    = 0x00000010;
/// const ACE4_EXECUTE              = 0x00000020;
/// const ACE4_DELETE_CHILD         = 0x00000040;
/// const ACE4_READ_ATTRIBUTES      = 0x00000080;
/// const ACE4_WRITE_ATTRIBUTES     = 0x00000100;
///
/// const ACE4_DELETE               = 0x00010000;
/// const ACE4_READ_ACL             = 0x00020000;
/// const ACE4_WRITE_ACL            = 0x00040000;
/// const ACE4_WRITE_OWNER          = 0x00080000;
/// const ACE4_SYNCHRONIZE          = 0x00100000;
///
///
```

```
/// /*
///  * ACE4_GENERIC_READ - defined as a combination of
///  *      ACE4_READ_ACL |
///  *      ACE4_READ_DATA |
///  *      ACE4_READ_ATTRIBUTES |
///  *      ACE4_SYNCHRONIZE
///  */
///
/// const ACE4_GENERIC_READ = 0x00120081;
///
/// /*
///  * ACE4_GENERIC_WRITE - defined as a combination of
///  *      ACE4_READ_ACL |
///  *      ACE4_WRITE_DATA |
///  *      ACE4_WRITE_ATTRIBUTES |
///  *      ACE4_WRITE_ACL |
///  *      ACE4_APPEND_DATA |
///  *      ACE4_SYNCHRONIZE
///  */
/// const ACE4_GENERIC_WRITE = 0x00160106;
///
///
/// /*
///  * ACE4_GENERIC_EXECUTE - defined as a combination of
///  *      ACE4_READ_ACL
///  *      ACE4_READ_ATTRIBUTES
///  *      ACE4_EXECUTE
///  *      ACE4_SYNCHRONIZE
///  */
/// const ACE4_GENERIC_EXECUTE = 0x001200A0;
///
///
/// /*
///  * Access Control Entry definition
///  */
/// struct nfsace4 {
///         acetype4                type;
///         aceflag4                flag;
///         acemask4                access_mask;
///         utf8str_mixed           who;
/// };
///
```

```
///
/// /*
///  * Field definitions for the fattr4_mode attribute
///  */
/// const MODE4_SUID = 0x800;  /* set user id on execution */
/// const MODE4_SGID = 0x400;  /* set group id on execution */
/// const MODE4_SVTX = 0x200;  /* save text even after use */
/// const MODE4_RUSR = 0x100;  /* read permission: owner */
/// const MODE4_WUSR = 0x080;  /* write permission: owner */
/// const MODE4_XUSR = 0x040;  /* execute permission: owner */
/// const MODE4_RGRP = 0x020;  /* read permission: group */
/// const MODE4_WGRP = 0x010;  /* write permission: group */
/// const MODE4_XGRP = 0x008;  /* execute permission: group */
/// const MODE4_ROTH = 0x004;  /* read permission: other */
/// const MODE4_WOTH = 0x002;  /* write permission: other */
/// const MODE4_XOTH = 0x001;  /* execute permission: other */
///
///
/// /*
///  * Special data/attribute associated with
///  * file types NF4BLK and NF4CHR.
///  */
/// struct specdata4 {
///  uint32_t specdata1; /* major device number */
///  uint32_t specdata2; /* minor device number */
/// };
///
///
/// /*
///  * Values for fattr4_fh_expire_type
///  */
/// const FH4_PERSISTENT          = 0x00000000;
/// const FH4_NOEXPIRE_WITH_OPEN  = 0x00000001;
/// const FH4_VOLATILE_ANY        = 0x00000002;
/// const FH4_VOL_MIGRATION       = 0x00000004;
/// const FH4_VOL_RENAME          = 0x00000008;
///
///
/// typedef bitmap4                 fattr4_supported_attrs;
/// typedef nfs_ftype4              fattr4_type;
/// typedef uint32_t                fattr4_fh_expire_type;
/// typedef changeid4               fattr4_change;
/// typedef uint64_t                fattr4_size;
/// typedef bool                    fattr4_link_support;
/// typedef bool                    fattr4_symlink_support;
/// typedef bool                    fattr4_named_attr;
/// typedef fsid4                   fattr4_fsid;
```

```
    /// typedef bool                 fattr4_unique_handles;
    /// typedef nfs_lease4           fattr4_lease_time;
    /// typedef nfsstat4             fattr4_rdattr_error;
    ///
    /// typedef nfsace4              fattr4_acl<>;
    /// typedef uint32_t             fattr4_aclsupport;
    /// typedef bool                 fattr4_archive;
    /// typedef bool                 fattr4_cansettime;
    /// typedef bool                 fattr4_case_insensitive;
    /// typedef bool                 fattr4_case_preserving;
    /// typedef bool                 fattr4_chown_restricted;
    /// typedef uint64_t             fattr4_fileid;
    /// typedef uint64_t             fattr4_files_avail;
    /// typedef nfs_fh4              fattr4_filehandle;
    /// typedef uint64_t             fattr4_files_free;
    /// typedef uint64_t             fattr4_files_total;
    /// typedef fs_locations4        fattr4_fs_locations;
    /// typedef bool                 fattr4_hidden;
    /// typedef bool                 fattr4_homogeneous;
    /// typedef uint64_t             fattr4_maxfilesize;
    /// typedef uint32_t             fattr4_maxlink;
    /// typedef uint32_t             fattr4_maxname;
    /// typedef uint64_t             fattr4_maxread;
    /// typedef uint64_t             fattr4_maxwrite;
    /// typedef ascii_REQUIRED4      fattr4_mimetype;
    /// typedef mode4                fattr4_mode;
    /// typedef uint64_t             fattr4_mounted_on_fileid;
    /// typedef bool                 fattr4_no_trunc;
    /// typedef uint32_t             fattr4_numlinks;
    /// typedef utf8str_mixed        fattr4_owner;
    /// typedef utf8str_mixed        fattr4_owner_group;
    /// typedef uint64_t             fattr4_quota_avail_hard;
    /// typedef uint64_t             fattr4_quota_avail_soft;
    /// typedef uint64_t             fattr4_quota_used;
    /// typedef specdata4            fattr4_rawdev;
    /// typedef uint64_t             fattr4_space_avail;
    /// typedef uint64_t             fattr4_space_free;
    /// typedef uint64_t             fattr4_space_total;
    /// typedef uint64_t             fattr4_space_used;
    /// typedef bool                 fattr4_system;
    /// typedef nfstime4             fattr4_time_access;
    /// typedef settime4             fattr4_time_access_set;
    /// typedef nfstime4             fattr4_time_backup;
    /// typedef nfstime4             fattr4_time_create;
    /// typedef nfstime4             fattr4_time_delta;
    /// typedef nfstime4             fattr4_time_metadata;
    /// typedef nfstime4             fattr4_time_modify;
    /// typedef settime4             fattr4_time_modify_set;
```

```
///
///
/// /*
///  * Mandatory attributes
///  */
/// const FATTR4_SUPPORTED_ATTRS    = 0;
/// const FATTR4_TYPE               = 1;
/// const FATTR4_FH_EXPIRE_TYPE     = 2;
/// const FATTR4_CHANGE             = 3;
/// const FATTR4_SIZE               = 4;
/// const FATTR4_LINK_SUPPORT       = 5;
/// const FATTR4_SYMLINK_SUPPORT    = 6;
/// const FATTR4_NAMED_ATTR         = 7;
/// const FATTR4_FSID               = 8;
/// const FATTR4_UNIQUE_HANDLES     = 9;
/// const FATTR4_LEASE_TIME         = 10;
/// const FATTR4_RDATTR_ERROR       = 11;
/// const FATTR4_FILEHANDLE         = 19;
///
/// /*
///  * Recommended attributes
///  */
/// const FATTR4_ACL                = 12;
/// const FATTR4_ACLSUPPORT         = 13;
/// const FATTR4_ARCHIVE            = 14;
/// const FATTR4_CANSETTIME         = 15;
/// const FATTR4_CASE_INSENSITIVE   = 16;
/// const FATTR4_CASE_PRESERVING    = 17;
/// const FATTR4_CHOWN_RESTRICTED   = 18;
/// const FATTR4_FILEID             = 20;
/// const FATTR4_FILES_AVAIL        = 21;
/// const FATTR4_FILES_FREE         = 22;
/// const FATTR4_FILES_TOTAL        = 23;
/// const FATTR4_FS_LOCATIONS       = 24;
/// const FATTR4_HIDDEN             = 25;
/// const FATTR4_HOMOGENEOUS        = 26;
/// const FATTR4_MAXFILESIZE        = 27;
/// const FATTR4_MAXLINK            = 28;
/// const FATTR4_MAXNAME            = 29;
/// const FATTR4_MAXREAD            = 30;
/// const FATTR4_MAXWRITE           = 31;
/// const FATTR4_MIMETYPE           = 32;
/// const FATTR4_MODE               = 33;
/// const FATTR4_NO_TRUNC           = 34;
/// const FATTR4_NUMLINKS           = 35;
/// const FATTR4_OWNER              = 36;
/// const FATTR4_OWNER_GROUP        = 37;
/// const FATTR4_QUOTA_AVAIL_HARD   = 38;
```

```
/// const FATTR4_QUOTA_AVAIL_SOFT   = 39;
/// const FATTR4_QUOTA_USED         = 40;
/// const FATTR4_RAWDEV             = 41;
/// const FATTR4_SPACE_AVAIL        = 42;
/// const FATTR4_SPACE_FREE         = 43;
/// const FATTR4_SPACE_TOTAL        = 44;
/// const FATTR4_SPACE_USED         = 45;
/// const FATTR4_SYSTEM             = 46;
/// const FATTR4_TIME_ACCESS        = 47;
/// const FATTR4_TIME_ACCESS_SET    = 48;
/// const FATTR4_TIME_BACKUP        = 49;
/// const FATTR4_TIME_CREATE        = 50;
/// const FATTR4_TIME_DELTA         = 51;
/// const FATTR4_TIME_METADATA      = 52;
/// const FATTR4_TIME_MODIFY        = 53;
/// const FATTR4_TIME_MODIFY_SET    = 54;
/// const FATTR4_MOUNTED_ON_FILEID  = 55;
///
/// /*
///  * File attribute container
///  */
/// struct fattr4 {
///         bitmap4         attrmask;
///         attrlist4       attr_vals;
/// };
///
///
/// /*
///  * Change info for the client
///  */
/// struct change_info4 {
///         bool            atomic;
///         changeid4       before;
///         changeid4       after;
/// };
///
///
/// struct clientaddr4 {
///         /* see struct rpcb in RFC 1833 */
///         string r_netid<>;       /* network id */
///         string r_addr<>;        /* universal address */
/// };
///
```

```
///
/// /*
///  * Callback program info as provided by the client
///  */
/// struct cb_client4 {
///         unsigned int    cb_program;
///         clientaddr4     cb_location;
/// };
///
///
/// /*
///  * Stateid
///  */
/// struct stateid4 {
///         uint32_t        seqid;
///         opaque          other[NFS4_OTHER_SIZE];
/// };
///
/// /*
///  * Client ID
///  */
/// struct nfs_client_id4 {
///         verifier4       verifier;
///         opaque          id<NFS4_OPAQUE_LIMIT>;
/// };
///
///
/// struct open_owner4 {
///         clientid4       clientid;
///         opaque          owner<NFS4_OPAQUE_LIMIT>;
/// };
///
///
/// struct lock_owner4 {
///         clientid4       clientid;
///         opaque          owner<NFS4_OPAQUE_LIMIT>;
/// };
///
///
/// enum nfs_lock_type4 {
///         READ_LT         = 1,
///         WRITE_LT        = 2,
///         READW_LT        = 3,    /* blocking read */
///         WRITEW_LT       = 4     /* blocking write */
/// };
///
```

```
///
/// const ACCESS4_READ      = 0x00000001;
/// const ACCESS4_LOOKUP    = 0x00000002;
/// const ACCESS4_MODIFY    = 0x00000004;
/// const ACCESS4_EXTEND    = 0x00000008;
/// const ACCESS4_DELETE    = 0x00000010;
/// const ACCESS4_EXECUTE   = 0x00000020;
///
/// struct ACCESS4args {
///         /* CURRENT_FH: object */
///         uint32_t        access;
/// };
///
/// struct ACCESS4resok {
///         uint32_t        supported;
///         uint32_t        access;
/// };
///
/// union ACCESS4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          ACCESS4resok   resok4;
///  default:
///          void;
/// };
///
/// struct CLOSE4args {
///         /* CURRENT_FH: object */
///         seqid4          seqid;
///         stateid4        open_stateid;
/// };
///
/// union CLOSE4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          stateid4       open_stateid;
///  default:
///          void;
/// };
///
/// struct COMMIT4args {
///         /* CURRENT_FH: file */
///         offset4         offset;
///         count4          count;
/// };
///
/// struct COMMIT4resok {
///         verifier4       writeverf;
/// };
///
```

```
/// union COMMIT4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          COMMIT4resok   resok4;
///  default:
///          void;
/// };
///
/// union createtype4 switch (nfs_ftype4 type) {
///  case NF4LNK:
///          linktext4 linkdata;
///  case NF4BLK:
///  case NF4CHR:
///          specdata4 devdata;
///  case NF4SOCK:
///  case NF4FIFO:
///  case NF4DIR:
///          void;
///  default:
///          void;  /* server should return NFS4ERR_BADTYPE */
/// };
///
/// struct CREATE4args {
///         /* CURRENT_FH: directory for creation */
///         createtype4     objtype;
///         component4      objname;
///         fattr4          createattrs;
/// };
///
/// struct CREATE4resok {
///         change_info4    cinfo;
///         bitmap4         attrset;        /* attributes set */
/// };
///
/// union CREATE4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          CREATE4resok resok4;
///  default:
///          void;
/// };
///
/// struct DELEGPURGE4args {
///         clientid4       clientid;
/// };
///
/// struct DELEGPURGE4res {
///         nfsstat4        status;
/// };
///
```

```
/// struct DELEGRETURN4args {
///         /* CURRENT_FH: delegated file */
///         stateid4        deleg_stateid;
/// };
///
/// struct DELEGRETURN4res {
///         nfsstat4        status;
/// };
///
/// struct GETATTR4args {
///         /* CURRENT_FH: directory or file */
///         bitmap4         attr_request;
/// };
///
/// struct GETATTR4resok {
///         fattr4          obj_attributes;
/// };
///
/// union GETATTR4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          GETATTR4resok  resok4;
///  default:
///          void;
/// };
///
/// struct GETFH4resok {
///         nfs_fh4         object;
/// };
///
/// union GETFH4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          GETFH4resok     resok4;
///  default:
///          void;
/// };
///
/// struct LINK4args {
///         /* SAVED_FH: source object */
///         /* CURRENT_FH: target directory */
///         component4      newname;
/// };
///
/// struct LINK4resok {
///         change_info4    cinfo;
/// };
///
```

```
/// union LINK4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          LINK4resok resok4;
///  default:
///          void;
/// };
///
/// /*
///  * For LOCK, transition from open_owner to new lock_owner
///  */
/// struct open_to_lock_owner4 {
///         seqid4          open_seqid;
///         stateid4        open_stateid;
///         seqid4          lock_seqid;
///         lock_owner4     lock_owner;
/// };
///
/// /*
///  * For LOCK, existing lock_owner continues to request file locks
///  */
/// struct exist_lock_owner4 {
///         stateid4        lock_stateid;
///         seqid4          lock_seqid;
/// };
///
/// union locker4 switch (bool new_lock_owner) {
///  case TRUE:
///         open_to_lock_owner4     open_owner;
///  case FALSE:
///         exist_lock_owner4       lock_owner;
/// };
///
/// /*
///  * LOCK/LOCKT/LOCKU: Record lock management
///  */
/// struct LOCK4args {
///         /* CURRENT_FH: file */
///         nfs_lock_type4  locktype;
///         bool            reclaim;
///         offset4         offset;
///         length4         length;
///         locker4         locker;
/// };
///
```

```
/// struct LOCK4denied {
///         offset4         offset;
///         length4         length;
///         nfs_lock_type4  locktype;
///         lock_owner4     owner;
/// };
///
/// struct LOCK4resok {
///         stateid4        lock_stateid;
/// };
///
/// union LOCK4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          LOCK4resok     resok4;
///  case NFS4ERR_DENIED:
///          LOCK4denied    denied;
///  default:
///          void;
/// };
///
/// struct LOCKT4args {
///         /* CURRENT_FH: file */
///         nfs_lock_type4  locktype;
///         offset4         offset;
///         length4         length;
///         lock_owner4     owner;
/// };
///
/// union LOCKT4res switch (nfsstat4 status) {
///  case NFS4ERR_DENIED:
///          LOCK4denied    denied;
///  case NFS4_OK:
///          void;
///  default:
///          void;
/// };
///
/// struct LOCKU4args {
///         /* CURRENT_FH: file */
///         nfs_lock_type4  locktype;
///         seqid4          seqid;
///         stateid4        lock_stateid;
///         offset4         offset;
///         length4         length;
/// };
///
```

```
/// union LOCKU4res switch (nfsstat4 status) {
///  case NFS4_OK:
///           stateid4       lock_stateid;
///  default:
///           void;
/// };
///
/// struct LOOKUP4args {
///         /* CURRENT_FH: directory */
///         component4      objname;
/// };
///
/// struct LOOKUP4res {
///         /* CURRENT_FH: object */
///         nfsstat4        status;
/// };
///
/// struct LOOKUPP4res {
///         /* CURRENT_FH: directory */
///         nfsstat4        status;
/// };
///
/// struct NVERIFY4args {
///         /* CURRENT_FH: object */
///         fattr4          obj_attributes;
/// };
///
/// struct NVERIFY4res {
///         nfsstat4        status;
/// };
///
/// const OPEN4_SHARE_ACCESS_READ   = 0x00000001;
/// const OPEN4_SHARE_ACCESS_WRITE  = 0x00000002;
/// const OPEN4_SHARE_ACCESS_BOTH   = 0x00000003;
///
/// const OPEN4_SHARE_DENY_NONE     = 0x00000000;
/// const OPEN4_SHARE_DENY_READ     = 0x00000001;
/// const OPEN4_SHARE_DENY_WRITE    = 0x00000002;
/// const OPEN4_SHARE_DENY_BOTH     = 0x00000003;
/// /*
///  * Various definitions for OPEN
///  */
/// enum createmode4 {
///         UNCHECKED4      = 0,
///         GUARDED4        = 1,
///         EXCLUSIVE4      = 2
/// };
///
```

```
/// union createhow4 switch (createmode4 mode) {
///  case UNCHECKED4:
///  case GUARDED4:
///          fattr4         createattrs;
///  case EXCLUSIVE4:
///          verifier4      createverf;
/// };
///
/// enum opentype4 {
///         OPEN4_NOCREATE  = 0,
///         OPEN4_CREATE    = 1
/// };
///
/// union openflag4 switch (opentype4 opentype) {
///  case OPEN4_CREATE:
///          createhow4     how;
///  default:
///          void;
/// };
///
/// /* Next definitions used for OPEN delegation */
/// enum limit_by4 {
///         NFS_LIMIT_SIZE          = 1,
///         NFS_LIMIT_BLOCKS        = 2
///         /* others as needed */
/// };
///
/// struct nfs_modified_limit4 {
///         uint32_t        num_blocks;
///         uint32_t        bytes_per_block;
/// };
///
/// union nfs_space_limit4 switch (limit_by4 limitby) {
///  /* limit specified as file size */
///  case NFS_LIMIT_SIZE:
///          uint64_t              filesize;
///  /* limit specified by number of blocks */
///  case NFS_LIMIT_BLOCKS:
///          nfs_modified_limit4    mod_blocks;
/// } ;
///
/// enum open_delegation_type4 {
///         OPEN_DELEGATE_NONE      = 0,
///         OPEN_DELEGATE_READ      = 1,
///         OPEN_DELEGATE_WRITE     = 2
/// };
///
```

```
/// enum open_claim_type4 {
///         CLAIM_NULL              = 0,
///         CLAIM_PREVIOUS          = 1,
///         CLAIM_DELEGATE_CUR      = 2,
///         CLAIM_DELEGATE_PREV     = 3
/// };
///
/// struct open_claim_delegate_cur4 {
///         stateid4        delegate_stateid;
///         component4      file;
/// };
///
/// union open_claim4 switch (open_claim_type4 claim) {
///  /*
///   * No special rights to file.
///   * Ordinary OPEN of the specified file.
///   */
///  case CLAIM_NULL:
///         /* CURRENT_FH: directory */
///         component4      file;
///  /*
///   * Right to the file established by an
///   * open previous to server reboot.  File
///   * identified by filehandle obtained at
///   * that time rather than by name.
///   */
///  case CLAIM_PREVIOUS:
///         /* CURRENT_FH: file being reclaimed */
///         open_delegation_type4   delegate_type;
///
///  /*
///   * Right to file based on a delegation
///   * granted by the server.  File is
///   * specified by name.
///   */
///  case CLAIM_DELEGATE_CUR:
///         /* CURRENT_FH: directory */
///         open_claim_delegate_cur4        delegate_cur_info;
///
///  /*
///   * Right to file based on a delegation
///   * granted to a previous boot instance
///   * of the client.  File is specified by name.
///   */
///  case CLAIM_DELEGATE_PREV:
///         /* CURRENT_FH: directory */
///         component4      file_delegate_prev;
/// };
```

```
///
/// /*
///  * OPEN: Open a file, potentially receiving an open delegation
///  */
/// struct OPEN4args {
///         seqid4          seqid;
///         uint32_t        share_access;
///         uint32_t        share_deny;
///         open_owner4     owner;
///         openflag4       openhow;
///         open_claim4     claim;
/// };
///
/// struct open_read_delegation4 {
///  stateid4 stateid;    /* Stateid for delegation */
///  bool     recall;     /* Pre-recalled flag for
///                          delegations obtained
///                          by reclaim (CLAIM_PREVIOUS). */
///
///  nfsace4 permissions; /* Defines users who don't
///                          need an ACCESS call to
///                          open for read. */
/// };
///
/// struct open_write_delegation4 {
///  stateid4 stateid;      /* Stateid for delegation */
///  bool     recall;       /* Pre-recalled flag for
///                            delegations obtained
///                            by reclaim
///                            (CLAIM_PREVIOUS). */
///
///  nfs_space_limit4
///            space_limit; /* Defines condition that
///                            the client must check to
///                            determine whether the
///                            file needs to be flushed
///                            to the server on close. */
///
///  nfsace4   permissions; /* Defines users who don't
///                            need an ACCESS call as
///                            part of a delegated
///                            open. */
/// };
///
```

```
/// union open_delegation4
/// switch (open_delegation_type4 delegation_type) {
///         case OPEN_DELEGATE_NONE:
///                 void;
///         case OPEN_DELEGATE_READ:
///                 open_read_delegation4 read;
///         case OPEN_DELEGATE_WRITE:
///                 open_write_delegation4 write;
/// };
///
/// /*
///  * Result flags
///  */
///
/// /* Client must confirm open */
/// const OPEN4_RESULT_CONFIRM      = 0x00000002;
/// /* Type of file locking behavior at the server */
/// const OPEN4_RESULT_LOCKTYPE_POSIX = 0x00000004;
///
/// struct OPEN4resok {
///  stateid4          stateid;     /* Stateid for open */
///  change_info4      cinfo;       /* Directory change info */
///  uint32_t          rflags;      /* Result flags */
///  bitmap4           attrset;     /* attribute set for create */
///  open_delegation4  delegation;  /* Info on any open
///                                    delegation */
/// };
///
/// union OPEN4res switch (nfsstat4 status) {
///  case NFS4_OK:
///           /* CURRENT_FH: opened file */
///           OPEN4resok      resok4;
///  default:
///           void;
/// };
///
/// struct OPENATTR4args {
///         /* CURRENT_FH: object */
///         bool    createdir;
/// };
///
/// struct OPENATTR4res {
///         /* CURRENT_FH: named attr directory */
///         nfsstat4        status;
/// };
///
```

```
/// struct OPEN_CONFIRM4args {
///         /* CURRENT_FH: opened file */
///         stateid4        open_stateid;
///         seqid4          seqid;
/// };
///
/// struct OPEN_CONFIRM4resok {
///         stateid4        open_stateid;
/// };
///
/// union OPEN_CONFIRM4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          OPEN_CONFIRM4resok     resok4;
///  default:
///          void;
/// };
///
/// struct OPEN_DOWNGRADE4args {
///         /* CURRENT_FH: opened file */
///         stateid4        open_stateid;
///         seqid4          seqid;
///         uint32_t        share_access;
///         uint32_t        share_deny;
/// };
///
/// struct OPEN_DOWNGRADE4resok {
///         stateid4        open_stateid;
/// };
///
/// union OPEN_DOWNGRADE4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          OPEN_DOWNGRADE4resok    resok4;
///  default:
///          void;
/// };
///
/// struct PUTFH4args {
///         nfs_fh4         object;
/// };
///
/// struct PUTFH4res {
///         /* CURRENT_FH: */
///         nfsstat4        status;
/// };
///
```

```
/// struct PUTPUBFH4res {
///         /* CURRENT_FH: public fh */
///         nfsstat4        status;
/// };
///
/// struct PUTROOTFH4res {
///         /* CURRENT_FH: root fh */
///         nfsstat4        status;
/// };
///
/// struct READ4args {
///         /* CURRENT_FH: file */
///         stateid4        stateid;
///         offset4         offset;
///         count4          count;
/// };
///
/// struct READ4resok {
///         bool            eof;
///         opaque          data<>;
/// };
///
/// union READ4res switch (nfsstat4 status) {
///  case NFS4_OK:
///           READ4resok     resok4;
///  default:
///           void;
/// };
///
/// struct READDIR4args {
///         /* CURRENT_FH: directory */
///         nfs_cookie4     cookie;
///         verifier4       cookieverf;
///         count4          dircount;
///         count4          maxcount;
///         bitmap4         attr_request;
/// };
///
/// struct entry4 {
///         nfs_cookie4     cookie;
///         component4      name;
///         fattr4          attrs;
///         entry4          *nextentry;
/// };
///
```

```
/// struct dirlist4 {
///         entry4          *entries;
///         bool            eof;
/// };
///
/// struct READDIR4resok {
///         verifier4       cookieverf;
///         dirlist4        reply;
/// };
///
///
/// union READDIR4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          READDIR4resok  resok4;
///  default:
///          void;
/// };
///
///
/// struct READLINK4resok {
///         linktext4       link;
/// };
///
/// union READLINK4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          READLINK4resok resok4;
///  default:
///          void;
/// };
///
/// struct REMOVE4args {
///         /* CURRENT_FH: directory */
///         component4      target;
/// };
///
/// struct REMOVE4resok {
///         change_info4    cinfo;
/// };
///
/// union REMOVE4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          REMOVE4resok   resok4;
///  default:
///          void;
/// };
///
```

```
/// struct RENAME4args {
///         /* SAVED_FH: source directory */
///         component4      oldname;
///         /* CURRENT_FH: target directory */
///         component4      newname;
/// };
///
/// struct RENAME4resok {
///         change_info4    source_cinfo;
///         change_info4    target_cinfo;
/// };
///
/// union RENAME4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          RENAME4resok    resok4;
///  default:
///          void;
/// };
///
/// struct RENEW4args {
///         clientid4       clientid;
/// };
///
/// struct RENEW4res {
///         nfsstat4        status;
/// };
///
/// struct RESTOREFH4res {
///         /* CURRENT_FH: value of saved fh */
///         nfsstat4        status;
/// };
///
/// struct SAVEFH4res {
///         /* SAVED_FH: value of current fh */
///         nfsstat4        status;
/// };
///
/// struct SECINFO4args {
///         /* CURRENT_FH: directory */
///         component4      name;
/// };
///
```

```
/// /*
///  * From RFC 2203
///  */
/// enum rpc_gss_svc_t {
///         RPC_GSS_SVC_NONE        = 1,
///         RPC_GSS_SVC_INTEGRITY   = 2,
///         RPC_GSS_SVC_PRIVACY     = 3
/// };
///
/// struct rpcsec_gss_info {
///         sec_oid4        oid;
///         qop4            qop;
///         rpc_gss_svc_t   service;
/// };
///
/// /* RPCSEC_GSS has a value of '6'.  See RFC 2203 */
/// union secinfo4 switch (uint32_t flavor) {
///  case RPCSEC_GSS:
///          rpcsec_gss_info        flavor_info;
///  default:
///          void;
/// };
///
/// typedef secinfo4 SECINFO4resok<>;
///
/// union SECINFO4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          SECINFO4resok resok4;
///  default:
///          void;
/// };
///
/// struct SETATTR4args {
///         /* CURRENT_FH: target object */
///         stateid4        stateid;
///         fattr4          obj_attributes;
/// };
///
/// struct SETATTR4res {
///         nfsstat4        status;
///         bitmap4         attrsset;
/// };
///
/// struct SETCLIENTID4args {
///         nfs_client_id4  client;
///         cb_client4      callback;
///         uint32_t        callback_ident;
/// };
```

```
///
/// struct SETCLIENTID4resok {
///         clientid4       clientid;
///         verifier4       setclientid_confirm;
/// };
///
/// union SETCLIENTID4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          SETCLIENTID4resok      resok4;
///  case NFS4ERR_CLID_INUSE:
///          clientaddr4    client_using;
///  default:
///          void;
/// };
///
/// struct SETCLIENTID_CONFIRM4args {
///         clientid4       clientid;
///         verifier4       setclientid_confirm;
/// };
///
/// struct SETCLIENTID_CONFIRM4res {
///         nfsstat4        status;
/// };
///
/// struct VERIFY4args {
///         /* CURRENT_FH: object */
///         fattr4          obj_attributes;
/// };
///
/// struct VERIFY4res {
///         nfsstat4        status;
/// };
///
/// enum stable_how4 {
///         UNSTABLE4       = 0,
///         DATA_SYNC4      = 1,
///         FILE_SYNC4      = 2
/// };
///
/// struct WRITE4args {
///         /* CURRENT_FH: file */
///         stateid4        stateid;
///         offset4         offset;
///         stable_how4     stable;
///         opaque          data<>;
/// };
///
```

```
/// struct WRITE4resok {
///         count4          count;
///         stable_how4     committed;
///         verifier4       writeverf;
/// };
///
/// union WRITE4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          WRITE4resok    resok4;
///  default:
///          void;
/// };
///
/// struct RELEASE_LOCKOWNER4args {
///         lock_owner4     lock_owner;
/// };
///
/// struct RELEASE_LOCKOWNER4res {
///         nfsstat4        status;
/// };
///
/// struct ILLEGAL4res {
///         nfsstat4        status;
/// };
///
/// /*
///  * Operation arrays
///  */
///
/// enum nfs_opnum4 {
///  OP_ACCESS              = 3,
///  OP_CLOSE               = 4,
///  OP_COMMIT              = 5,
///  OP_CREATE              = 6,
///  OP_DELEGPURGE          = 7,
///  OP_DELEGRETURN         = 8,
///  OP_GETATTR             = 9,
///  OP_GETFH               = 10,
///  OP_LINK                = 11,
///  OP_LOCK                = 12,
///  OP_LOCKT               = 13,
///  OP_LOCKU               = 14,
///  OP_LOOKUP              = 15,
///  OP_LOOKUPP             = 16,
///  OP_NVERIFY             = 17,
///  OP_OPEN                = 18,
///  OP_OPENATTR            = 19,
///  OP_OPEN_CONFIRM        = 20,
```

```
///  OP_OPEN_DOWNGRADE      = 21,
///  OP_PUTFH               = 22,
///  OP_PUTPUBFH            = 23,
///  OP_PUTROOTFH           = 24,
///  OP_READ                = 25,
///  OP_READDIR             = 26,
///  OP_READLINK            = 27,
///  OP_REMOVE              = 28,
///  OP_RENAME              = 29,
///  OP_RENEW               = 30,
///  OP_RESTOREFH           = 31,
///  OP_SAVEFH              = 32,
///  OP_SECINFO             = 33,
///  OP_SETATTR             = 34,
///  OP_SETCLIENTID         = 35,
///  OP_SETCLIENTID_CONFIRM = 36,
///  OP_VERIFY              = 37,
///  OP_WRITE               = 38,
///  OP_RELEASE_LOCKOWNER   = 39,
///  OP_ILLEGAL             = 10044
/// };
///
/// union nfs_argop4 switch (nfs_opnum4 argop) {
///  case OP_ACCESS:        ACCESS4args opaccess;
///  case OP_CLOSE:         CLOSE4args opclose;
///  case OP_COMMIT:        COMMIT4args opcommit;
///  case OP_CREATE:        CREATE4args opcreate;
///  case OP_DELEGPURGE:    DELEGPURGE4args opdelegpurge;
///  case OP_DELEGRETURN:   DELEGRETURN4args opdelegreturn;
///  case OP_GETATTR:       GETATTR4args opgetattr;
///  case OP_GETFH:         void;
///  case OP_LINK:          LINK4args oplink;
///  case OP_LOCK:          LOCK4args oplock;
///  case OP_LOCKT:         LOCKT4args oplockt;
///  case OP_LOCKU:         LOCKU4args oplocku;
///  case OP_LOOKUP:        LOOKUP4args oplookup;
///  case OP_LOOKUPP:       void;
///  case OP_NVERIFY:       NVERIFY4args opnverify;
///  case OP_OPEN:          OPEN4args opopen;
///  case OP_OPENATTR:      OPENATTR4args opopenattr;
///  case OP_OPEN_CONFIRM:  OPEN_CONFIRM4args opopen_confirm;
///  case OP_OPEN_DOWNGRADE:
///                         OPEN_DOWNGRADE4args opopen_downgrade;
///  case OP_PUTFH:         PUTFH4args opputfh;
///  case OP_PUTPUBFH:      void;
///  case OP_PUTROOTFH:     void;
///  case OP_READ:          READ4args opread;
///  case OP_READDIR:       READDIR4args opreaddir;
```

```
///  case OP_READLINK:      void;
///  case OP_REMOVE:        REMOVE4args opremove;
///  case OP_RENAME:        RENAME4args oprename;
///  case OP_RENEW:         RENEW4args oprenew;
///  case OP_RESTOREFH:     void;
///  case OP_SAVEFH:        void;
///  case OP_SECINFO:       SECINFO4args opsecinfo;
///  case OP_SETATTR:       SETATTR4args opsetattr;
///  case OP_SETCLIENTID:   SETCLIENTID4args opsetclientid;
///  case OP_SETCLIENTID_CONFIRM: SETCLIENTID_CONFIRM4args
///                                 opsetclientid_confirm;
///  case OP_VERIFY:        VERIFY4args opverify;
///  case OP_WRITE:         WRITE4args opwrite;
///  case OP_RELEASE_LOCKOWNER:
///                         RELEASE_LOCKOWNER4args
///                         oprelease_lockowner;
///  case OP_ILLEGAL:       void;
/// };
///
/// union nfs_resop4 switch (nfs_opnum4 resop) {
///  case OP_ACCESS:        ACCESS4res opaccess;
///  case OP_CLOSE:         CLOSE4res opclose;
///  case OP_COMMIT:        COMMIT4res opcommit;
///  case OP_CREATE:        CREATE4res opcreate;
///  case OP_DELEGPURGE:    DELEGPURGE4res opdelegpurge;
///  case OP_DELEGRETURN:   DELEGRETURN4res opdelegreturn;
///  case OP_GETATTR:       GETATTR4res opgetattr;
///  case OP_GETFH:         GETFH4res opgetfh;
///  case OP_LINK:          LINK4res oplink;
///  case OP_LOCK:          LOCK4res oplock;
///  case OP_LOCKT:         LOCKT4res oplockt;
///  case OP_LOCKU:         LOCKU4res oplocku;
///  case OP_LOOKUP:        LOOKUP4res oplookup;
///  case OP_LOOKUPP:       LOOKUPP4res oplookupp;
///  case OP_NVERIFY:       NVERIFY4res opnverify;
///  case OP_OPEN:          OPEN4res opopen;
///  case OP_OPENATTR:      OPENATTR4res opopenattr;
///  case OP_OPEN_CONFIRM:  OPEN_CONFIRM4res opopen_confirm;
///  case OP_OPEN_DOWNGRADE:
///                         OPEN_DOWNGRADE4res
///                                 opopen_downgrade;
///  case OP_PUTFH:         PUTFH4res opputfh;
///  case OP_PUTPUBFH:      PUTPUBFH4res opputpubfh;
///  case OP_PUTROOTFH:     PUTROOTFH4res opputrootfh;
///  case OP_READ:          READ4res opread;
///  case OP_READDIR:       READDIR4res opreaddir;
///  case OP_READLINK:      READLINK4res opreadlink;
///  case OP_REMOVE:        REMOVE4res opremove;
```

```
///  case OP_RENAME:        RENAME4res oprename;
///  case OP_RENEW:         RENEW4res oprenew;
///  case OP_RESTOREFH:     RESTOREFH4res oprestorefh;
///  case OP_SAVEFH:        SAVEFH4res opsavefh;
///  case OP_SECINFO:       SECINFO4res opsecinfo;
///  case OP_SETATTR:       SETATTR4res opsetattr;
///  case OP_SETCLIENTID:   SETCLIENTID4res opsetclientid;
///  case OP_SETCLIENTID_CONFIRM:
///                         SETCLIENTID_CONFIRM4res
///                                 opsetclientid_confirm;
///  case OP_VERIFY:        VERIFY4res opverify;
///  case OP_WRITE:         WRITE4res opwrite;
///  case OP_RELEASE_LOCKOWNER:
///                         RELEASE_LOCKOWNER4res
///                                 oprelease_lockowner;
///  case OP_ILLEGAL:       ILLEGAL4res opillegal;
/// };
///
/// struct COMPOUND4args {
///         utf8str_cs      tag;
///         uint32_t        minorversion;
///         nfs_argop4      argarray<>;
/// };
///
/// struct COMPOUND4res {
///         nfsstat4        status;
///         utf8str_cs      tag;
///         nfs_resop4      resarray<>;
/// };
///
///
/// /*
///  * Remote file service routines
///  */
/// program NFS4_PROGRAM {
///         version NFS_V4 {
///                 void
///                         NFSPROC4_NULL(void) = 0;
///
///                 COMPOUND4res
///                         NFSPROC4_COMPOUND(COMPOUND4args) = 1;
///
///         } = 4;
/// } = 100003;
///
```

```
/// /*
///  * NFS4 callback procedure definitions and program
///  */
/// struct CB_GETATTR4args {
///         nfs_fh4 fh;
///         bitmap4 attr_request;
/// };
///
/// struct CB_GETATTR4resok {
///         fattr4  obj_attributes;
/// };
///
/// union CB_GETATTR4res switch (nfsstat4 status) {
///  case NFS4_OK:
///          CB_GETATTR4resok       resok4;
///  default:
///          void;
/// };
///
/// struct CB_RECALL4args {
///         stateid4        stateid;
///         bool            truncate;
///         nfs_fh4         fh;
/// };
///
/// struct CB_RECALL4res {
///         nfsstat4        status;
/// };
///
/// /*
///  * CB_ILLEGAL: Response for illegal operation numbers
///  */
/// struct CB_ILLEGAL4res {
///         nfsstat4        status;
/// };
///
/// /*
///  * Various definitions for CB_COMPOUND
///  */
/// enum nfs_cb_opnum4 {
///         OP_CB_GETATTR           = 3,
///         OP_CB_RECALL            = 4,
///         OP_CB_ILLEGAL           = 10044
/// };
///
```

```
/// union nfs_cb_argop4 switch (unsigned argop) {
///  case OP_CB_GETATTR:    CB_GETATTR4args opcbgetattr;
///  case OP_CB_RECALL:     CB_RECALL4args opcbrecall;
///  case OP_CB_ILLEGAL:    void;
/// };
///
/// union nfs_cb_resop4 switch (unsigned resop) {
///  case OP_CB_GETATTR:    CB_GETATTR4res opcbgetattr;
///  case OP_CB_RECALL:     CB_RECALL4res opcbrecall;
///  case OP_CB_ILLEGAL:    CB_ILLEGAL4res opcbillegal;
/// };
///
///
/// struct CB_COMPOUND4args {
///         utf8str_cs      tag;
///         uint32_t        minorversion;
///         uint32_t        callback_ident;
///         nfs_cb_argop4   argarray<>;
/// };
///
/// struct CB_COMPOUND4res {
///         nfsstat4        status;
///         utf8str_cs      tag;
///         nfs_cb_resop4   resarray<>;
/// };
///
///
///
/// /*
///  * Program number is in the transient range, since the client
///  * will assign the exact transient program number and provide
///  * that to the server via the SETCLIENTID operation.
///  */
/// program NFS4_CALLBACK {
///         version NFS_CB {
///                 void
///                         CB_NULL(void) = 0;
///                 CB_COMPOUND4res
///                         CB_COMPOUND(CB_COMPOUND4args) = 1;
///         } = 1;
/// } = 0x40000000;
```

3. Security Considerations

See the Security Considerations section of [RFC7530].

4. Normative References

[RFC4506] Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, RFC 4506, May 2006, <http://www.rfc-editor.org/info/rfc4506>.

[RFC7530] Haynes, T., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, March 2015, <http://www.rfc-editor.org/info/rfc7530>.

Acknowledgments

Tom Haynes would like to thank NetApp, Inc. for its funding of his time on this project.

David Quigley tested the extraction of the .x file from this document and corrected the two resulting errors.

Authors' Addresses

Thomas Haynes (editor)
Primary Data, Inc.
4300 El Camino Real Ste 100
Los Altos, CA 94022
United States

Phone: +1 408 215 1519
EMail: thomas.haynes@primarydata.com

David Noveck (editor)
Dell
300 Innovative Way
Nashua, NH 03062
United States

Phone: +1 781 572 8038
EMail: dave_noveck@dell.com