NFSv4 T. Haynes
Internet-Draft Editors
Intended status: Standards Track April 04, 2011
Expires: October 6, 2011

NFSv4 Minor Version 2 Protocol External Data Representation Standard (XDR) Description
draft-haynes-nfsv4-minorversion2-dot-x-02.txt

Abstract

This Internet-Draft provides the XDR description for NFSv4 minor version two.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 6, 2011.

Copyright Notice

Copyright (c) 2011 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. XDR Description of NFSv4.2 4
2. Security Considerations 78
3. IANA Considerations 78
4. Normative References 78
Author's Address 78

1. XDR Description of NFSv4.2

This document contains the XDR ([2]) description of NFSv4.2 protocol ([3]). In order to facilitate implementations that support all of NFSv4.0, NFSv4.1, and NFSv4.2, the description includes operations, and other features of NFSv4.0 and NFSv4.1 that do not apply to NFSv4.2.

The XDR description is provided in this document in a way that makes it simple for the reader to extract into ready to compile form. The reader can feed this document in the following shell script to produce the machine readable XDR description of NFSv4.2:

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

I.e. 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
///  *  draft-haynes-nfsv4-minorversion2-01
///  * Last updated Mon Apr  4 19:04:25 CDT 2011
///  */
/// /*
///  *  Copyright (C) The IETF Trust (2007-2008)
///  *  All Rights Reserved.
///  *
///  *  Copyright (C) The Internet Society (1998-2006).
///  *  All Rights Reserved.
///  */
///
/// /*
///  *      nfs4_prot.x
///  */
///
/// %#ifndef _AUTH_SYS_DEFINE_FOR_NFSv42
/// %#define _AUTH_SYS_DEFINE_FOR_NFSv42
/// %#include <rpc/auth_sys.h>
/// %typedef struct authsys_parms authsys_parms;
```

```
/// %#endif /* _AUTH_SYS_DEFINE_FOR_NFSv42 */
///
/// /*
///  * 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_OPAQUE_LIMIT         = 1024;
/// const NFS4_SESSIONID_SIZE       = 16;
///
/// const NFS4_INT64_MAX            = 0x7fffffffffffffff;
/// const NFS4_UINT64_MAX           = 0xffffffffffffffff;
/// const NFS4_INT32_MAX            = 0x7fffffff;
/// const NFS4_UINT32_MAX           = 0xffffffff;
///
/// const NFS4_MAXFILELEN           = 0xffffffffffffffff;
/// const NFS4_MAXFILEOFF           = 0xfffffffffffffffe;
///
///
/// /*
///  * 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 filesystems   */
///
/// /*
///  * Please do not allocate value 19; it was used in NFSv3
///  * and we do not want a value in NFSv3 to have a different
///  * meaning in NFSv4.x.
///  */
///
///  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 filesystem  */
///  NFS4ERR_ROFS           = 30,   /* read-only filesystem    */
///  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 is not a valid error in NFSv4.1 */
///  NFS4ERR_RESOURCE       = 10018,/* resource exhaustion     */
```

```
///
///  NFS4ERR_MOVED          = 10019,/* filesystem 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,/* overlapping lock range  */
///  NFS4ERR_SYMLINK        = 10029,/* should be file/directory*/
///  NFS4ERR_RESTOREFH      = 10030,/* no saved filehandle     */
///  NFS4ERR_LEASE_MOVED    = 10031,/* some filesystem 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,/* lockowner state revoked */
///  NFS4ERR_CB_PATH_DOWN   = 10048,/* callback path down      */
///
///  /* NFSv4.1 errors start here. */
///
///  NFS4ERR_BADIOMODE      = 10049,
///  NFS4ERR_BADLAYOUT      = 10050,
///  NFS4ERR_BAD_SESSION_DIGEST = 10051,
///  NFS4ERR_BADSESSION     = 10052,
///  NFS4ERR_BADSLOT        = 10053,
///  NFS4ERR_COMPLETE_ALREADY = 10054,
///  NFS4ERR_CONN_NOT_BOUND_TO_SESSION = 10055,
///  NFS4ERR_DELEG_ALREADY_WANTED = 10056,
///  NFS4ERR_BACK_CHAN_BUSY = 10057,/*backchan reqs outstanding*/
///  NFS4ERR_LAYOUTTRYLATER = 10058,
///  NFS4ERR_LAYOUTUNAVAILABLE = 10059,
///  NFS4ERR_NOMATCHING_LAYOUT = 10060,
///  NFS4ERR_RECALLCONFLICT = 10061,
///  NFS4ERR_UNKNOWN_LAYOUTTYPE = 10062,
```

```
///  NFS4ERR_SEQ_MISORDERED = 10063,/* unexpected seq.ID in req*/
///  NFS4ERR_SEQUENCE_POS   = 10064,/* [CB_]SEQ. op not 1st op */
///  NFS4ERR_REQ_TOO_BIG    = 10065,/* request too big         */
///  NFS4ERR_REP_TOO_BIG    = 10066,/* reply too big           */
///  NFS4ERR_REP_TOO_BIG_TO_CACHE =10067,/* rep. not all cached*/
///  NFS4ERR_RETRY_UNCACHED_REP =10068,/* retry & rep. uncached*/
///  NFS4ERR_UNSAFE_COMPOUND =10069,/* retry/recovery too hard */
///  NFS4ERR_TOO_MANY_OPS   = 10070,/*too many ops in [CB_]COMP*/
///  NFS4ERR_OP_NOT_IN_SESSION =10071,/* op needs [CB_]SEQ. op */
///  NFS4ERR_HASH_ALG_UNSUPP = 10072, /* hash alg. not supp.   */
///                                   /* Error 10073 is unused.  */
///  NFS4ERR_CLIENTID_BUSY  = 10074,/* clientid has state      */
///  NFS4ERR_PNFS_IO_HOLE   = 10075,/* IO to _SPARSE file hole */
///  NFS4ERR_SEQ_FALSE_RETRY= 10076,/* Retry != original req.  */
///  NFS4ERR_BAD_HIGH_SLOT  = 10077,/* req has bad highest_slot*/
///  NFS4ERR_DEADSESSION    = 10078,/*new req sent to dead sess*/
///  NFS4ERR_ENCR_ALG_UNSUPP= 10079,/* encr alg. not supp.     */
///  NFS4ERR_PNFS_NO_LAYOUT = 10080,/* I/O without a layout    */
///  NFS4ERR_NOT_ONLY_OP    = 10081,/* addl ops not allowed    */
///  NFS4ERR_WRONG_CRED     = 10082,/* op done by wrong cred   */
///  NFS4ERR_WRONG_TYPE     = 10083,/* op on wrong type object */
///  NFS4ERR_DIRDELEG_UNAVAIL=10084,/* delegation not avail.   */
///  NFS4ERR_REJECT_DELEG   = 10085,/* cb rejected delegation  */
///  NFS4ERR_RETURNCONFLICT = 10086,/* layout get before return*/
///  NFS4ERR_DELEG_REVOKED  = 10087,/* deleg./layout revoked   */
///
///  /* NFSv4.2 errors start here. */
///
///  NFS4ERR_PARTNER_NOTSUPP= 10088,/* s2s not supported       */
///  NFS4ERR_PARTNER_NO_AUTH= 10089,/* s2s not authorized      */
///  NFS4ERR_METADATA_NOTSUPP=10090 /* dest metadata diff sourc*/
/// };
///
/// /*
///  * 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 uint64_t        offset4;
/// typedef uint32_t        qop4;
/// typedef opaque  sec_oid4<>;
```

```
/// typedef uint32_t        sequenceid4;
/// typedef uint32_t        seqid4;
/// typedef opaque  sessionid4[NFS4_SESSIONID_SIZE];
/// typedef uint32_t        slotid4;
/// typedef opaque  utf8string<>;
/// typedef utf8string      utf8str_cis;
/// typedef utf8string      utf8str_cs;
/// typedef utf8string      utf8str_mixed;
/// typedef utf8str_cs      component4;
/// typedef utf8str_cs      linktext4;
/// typedef component4      pathname4<>;
/// typedef opaque  verifier4[NFS4_VERIFIER_SIZE];
/// typedef string  secret4<>;
///
/// /*
///  * 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;
/// };
///
///
/// typedef uint32_t nfs_lease4;
///
/// /*
///  * File attribute definitions
///  */
///
/// /*
///  * FSID structure for major/minor
///  */
/// struct fsid4 {
///         uint64_t        major;
///         uint64_t        minor;
/// };
```

```
///
/// /*
///  * Filesystem locations attribute
///  * for relocation/migration and
///  * related attributes.
///  */
/// struct change_policy4 {
///         uint64_t        cp_major;
///         uint64_t        cp_minor;
/// };
///
/// 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;
/// const ACE4_INHERITED_ACE                = 0x00000080;
///
///
///
/// /*
///  * 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_WRITE_RETENTION      = 0x00000200;
/// const ACE4_WRITE_RETENTION_HOLD = 0x00000400;
///
/// 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 combination of
///  *      ACE4_READ_ACL |
///  *      ACE4_READ_DATA |
///  *      ACE4_READ_ATTRIBUTES |
///  *      ACE4_SYNCHRONIZE
///  */
///
/// const ACE4_GENERIC_READ = 0x00120081;
///
/// /*
///  * ACE4_GENERIC_WRITE -- defined as 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 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;
/// };
///
///
/// /*
```

```
///  * ACL flag
///  */
///
/// typedef uint32_t aclflag4;
///
/// /*
///  * ACL flag values
///  */
/// const ACL4_AUTO_INHERIT         = 0x00000001;
/// const ACL4_PROTECTED            = 0x00000002;
/// const ACL4_DEFAULTED            = 0x00000004;
///
///
/// /*
///  * Version 4.1 Access Control List definition
///  */
/// struct nfsacl41 {
///         aclflag4        na41_flag;
///         nfsace4         na41_aces<>;
/// };
///
///
/// /*
///  * Field definitions for the fattr4_mode
///  * and fattr4_mode_set_masked attributes.
///  */
/// 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 */
///
///
/// /*
///  * Masked mode for the mode_set_masked attribute.
///  */
/// struct mode_masked4 {
///  mode4  mm_value_to_set; /* Values of bits
///                             to set or reset
///                             in mode. */
///
```

```
///  mode4  mm_mask_bits;    /* Mask of bits to
///                             set or reset
///                             in mode. */
/// };
///
/// /*
///  * 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;
///
///
/// struct netaddr4 {
///         /* see struct rpcb in RFC 1833 */
///         string na_r_netid<>;    /* network id */
///         string na_r_addr<>;     /* universal address */
/// };
///
///
/// /*
///  * data structures new to NFSv4.1
///  */
///
/// struct nfs_impl_id4 {
///         utf8str_cis   nii_domain;
///         utf8str_cs    nii_name;
///         nfstime4      nii_date;
/// };
///
///
/// /*
///  * Stateid
///  */
/// struct stateid4 {
///         uint32_t        seqid;
///         opaque          other[12];
```

```
/// };
///
/// enum layouttype4 {
///         LAYOUT4_NFSV4_1_FILES   = 0x1,
///         LAYOUT4_OSD2_OBJECTS    = 0x2,
///         LAYOUT4_BLOCK_VOLUME    = 0x3
/// };
///
/// struct layout_content4 {
///         layouttype4 loc_type;
///         opaque      loc_body<>;
/// };
///
///
/// %/*
/// % * LAYOUT4_OSD2_OBJECTS loc_body description
/// % * is in a separate .x file
/// % */
/// %
/// %/*
/// % * LAYOUT4_BLOCK_VOLUME loc_body description
/// % * is in a separate .x file
/// % */
///
/// struct layouthint4 {
///         layouttype4             loh_type;
///         opaque                  loh_body<>;
/// };
///
/// enum layoutiomode4 {
///         LAYOUTIOMODE4_READ      = 1,
///         LAYOUTIOMODE4_RW        = 2,
///         LAYOUTIOMODE4_ANY       = 3
/// };
///
/// struct layout4 {
///         offset4                 lo_offset;
///         length4                 lo_length;
///         layoutiomode4           lo_iomode;
///         layout_content4         lo_content;
/// };
///
/// const NFS4_DEVICEID4_SIZE = 16;
///
/// typedef opaque  deviceid4[NFS4_DEVICEID4_SIZE];
///
/// struct device_addr4 {
///         layouttype4             da_layout_type;
```

```
///         opaque                  da_addr_body<>;
/// };
///
///
/// struct layoutupdate4 {
///         layouttype4             lou_type;
///         opaque                  lou_body<>;
/// };
///
/// %
/// /* Constants used for LAYOUTRETURN and CB_LAYOUTRECALL */
/// const LAYOUT4_RET_REC_FILE              = 1;
/// const LAYOUT4_RET_REC_FSID              = 2;
/// const LAYOUT4_RET_REC_ALL               = 3;
/// const LAYOUT4_RET_REC_FILE_NO_ACCESS    = 4;
/// const LAYOUT4_RET_REC_FSID_NO_ACCESS    = 5;
/// const LAYOUT4_RET_REC_ALL_NO_ACCESS     = 6;
/// %
/// enum layoutreturn_type4 {
///         LAYOUTRETURN4_FILE = LAYOUT4_RET_REC_FILE,
///         LAYOUTRETURN4_FSID = LAYOUT4_RET_REC_FSID,
///         LAYOUTRETURN4_ALL  = LAYOUT4_RET_REC_ALL,
///         LAYOUTRETURN4_FILE_NO_ACCESS =
///                 LAYOUT4_RET_REC_FILE_NO_ACCESS,
///         LAYOUTRETURN4_FSID_NO_ACCESS =
///                 LAYOUT4_RET_REC_FSID_NO_ACCESS,
///         LAYOUTRETURN4_ALL_NO_ACCESS  =
///                 LAYOUT4_RET_REC_ALL_NO_ACCESS
/// };
///
/// struct layoutreturn_file4 {
///         offset4         lrf_offset;
///         length4         lrf_length;
///         stateid4        lrf_stateid;
/// %       /* layouttype4 specific data */
///         opaque          lrf_body<>;
/// };
///
/// struct layoutreturn_device_no_access4 {
///         deviceid4       lrdna_deviceid;
///         nfsstat4        lrdna_status;
/// };
///
/// struct layoutreturn_file_no_access4 {
///         offset4         lrfna_offset;
///         length4         lrfna_length;
///         stateid4        lrfna_stateid;
///         deviceid4       lrfna_deviceid;
```

```
///         nfsstat4        lrfna_status;
/// /* layouttype4 specific data */
///         opaque          lrfna_body<>;
/// };
///
/// union layoutreturn4 switch(layoutreturn_type4 lr_returntype) {
///         case LAYOUTRETURN4_FILE:
///                 layoutreturn_file4      lr_layout;
///         case LAYOUTRETURN4_FILE_NO_ACCESS:
///                 layoutreturn_file_no_access4    lr_layout_na;
///         case LAYOUTRETURN4_FSID_NO_ACCESS:
///         case LAYOUTRETURN4_ALL_NO_ACCESS:
///                 layoutreturn_device_no_access4  lr_device<>;
///         default:
///                 void;
/// };
/// %
///
/// enum fs4_status_type {
///         STATUS4_FIXED = 1,
///         STATUS4_UPDATED = 2,
///         STATUS4_VERSIONED = 3,
///         STATUS4_WRITABLE = 4,
///         STATUS4_REFERRAL = 5
/// };
///
/// struct fs4_status {
///         bool            fss_absent;
///         fs4_status_type fss_type;
///         utf8str_cs      fss_source;
///         utf8str_cs      fss_current;
///         int32_t         fss_age;
///         nfstime4        fss_version;
/// };
///
///
/// const TH4_READ_SIZE     = 0;
/// const TH4_WRITE_SIZE    = 1;
/// const TH4_READ_IOSIZE   = 2;
/// const TH4_WRITE_IOSIZE  = 3;
///
/// typedef length4 threshold4_read_size;
/// typedef length4 threshold4_write_size;
/// typedef length4 threshold4_read_iosize;
/// typedef length4 threshold4_write_iosize;
///
/// struct threshold_item4 {
///         layouttype4     thi_layout_type;
```

```
///         bitmap4         thi_hintset;
///         opaque          thi_hintlist<>;
/// };
///
/// struct mdsthreshold4 {
///         threshold_item4 mth_hints<>;
/// };
///
/// const RET4_DURATION_INFINITE    = 0xffffffffffffffff;
/// struct retention_get4 {
///         uint64_t        rg_duration;
///         nfstime4        rg_begin_time<1>;
/// };
///
/// struct retention_set4 {
///         bool            rs_enable;
///         uint64_t        rs_duration<1>;
/// };
///
/// const FSCHARSET_CAP4_CONTAINS_NON_UTF8  = 0x1;
/// const FSCHARSET_CAP4_ALLOWS_ONLY_UTF8   = 0x2;
///
/// typedef uint32_t        fs_charset_cap4;
///
///
/// /*
///  * data structures new to NFSv4.2
///  */
///
/// enum netloc_type4 {
///         NL4_NAME        = 0,
///         NL4_URL         = 1,
///         NL4_NETADDR     = 2
/// };
/// union netloc4 switch (netloc_type4 nl_type) {
///         case NL4_NAME:          utf8str_cis nl_name;
///         case NL4_URL:           utf8str_cis nl_url;
///         case NL4_NETADDR:       netaddr4    nl_addr;
/// };
///
/// enum change_attr_typeinfo {
///            NFS4_CHANGE_TYPE_IS_MONOTONIC_INCR         = 0,
///            NFS4_CHANGE_TYPE_IS_VERSION_COUNTER        = 1,
///            NFS4_CHANGE_TYPE_IS_VERSION_COUNTER_NOPNFS = 2,
///            NFS4_CHANGE_TYPE_IS_TIME_METADATA          = 3,
///            NFS4_CHANGE_TYPE_IS_UNDEFINED              = 4
/// };
///
```

```
///
/// struct copy_from_auth_priv {
///         secret4             cfap_shared_secret;
///         netloc4             cfap_destination;
///         /* the NFSv4 user name that the user principal maps to */
///         utf8str_mixed       cfap_username;
///         /* equal to seq_num of rpc_gss_cred_vers_3_t */
///         unsigned int        cfap_seq_num;
/// };
///
/// struct copy_to_auth_priv {
///         /* equal to cfap_shared_secret */
///         secret4              ctap_shared_secret;
///         netloc4              ctap_source;
///         /* the NFSv4 user name that the user principal maps to */
///         utf8str_mixed        ctap_username;
///         /* equal to seq_num of rpc_gss_cred_vers_3_t */
///         unsigned int         ctap_seq_num;
/// };
///
/// struct copy_confirm_auth_priv {
///         /* equal to GSS_GetMIC() of cfap_shared_secret */
///         opaque              ccap_shared_secret_mic<>;
///         /* the NFSv4 user name that the user principal maps to */
///         utf8str_mixed       ccap_username;
///         /* equal to seq_num of rpc_gss_cred_vers_3_t */
///         unsigned int        ccap_seq_num;
/// };
///
///
/// /*
///  * NFSv4.1 attributes
///  */
/// 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 utf8str_cs      fattr4_mimetype;
/// typedef mode4           fattr4_mode;
/// typedef mode_masked4    fattr4_mode_set_masked;
/// 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;
/// /*
///  * attributes new to NFSv4.1
///  */
/// typedef bitmap4         fattr4_suppattr_exclcreat;
/// typedef nfstime4        fattr4_dir_notif_delay;
/// typedef nfstime4        fattr4_dirent_notif_delay;
```

```
/// typedef layouttype4     fattr4_fs_layout_types<>;
/// typedef fs4_status      fattr4_fs_status;
/// typedef fs_charset_cap4 fattr4_fs_charset_cap;
/// typedef uint32_t        fattr4_layout_alignment;
/// typedef uint32_t        fattr4_layout_blksize;
/// typedef layouthint4     fattr4_layout_hint;
/// typedef layouttype4     fattr4_layout_types<>;
/// typedef mdsthreshold4   fattr4_mdsthreshold;
/// typedef retention_get4  fattr4_retention_get;
/// typedef retention_set4  fattr4_retention_set;
/// typedef retention_get4  fattr4_retentevt_get;
/// typedef retention_set4  fattr4_retentevt_set;
/// typedef uint64_t        fattr4_retention_hold;
/// typedef nfsacl41        fattr4_dacl;
/// typedef nfsacl41        fattr4_sacl;
/// typedef change_policy4  fattr4_change_policy;
///
/// %/*
/// % * REQUIRED 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;
///
/// %/*
/// % * new to NFSV4.1
/// % */
/// const FATTR4_SUPPATTR_EXCLCREAT = 75;
///
/// %/*
/// % * 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;
///
/// %/*
/// % * new to NFSV4.1
/// % */
/// const FATTR4_DIR_NOTIF_DELAY    = 56;
/// const FATTR4_DIRENT_NOTIF_DELAY = 57;
/// const FATTR4_DACL               = 58;
/// const FATTR4_SACL               = 59;
/// const FATTR4_CHANGE_POLICY      = 60;
///
/// %/*
```

```
/// % * new to NFSV4.2
/// % */
/// const FATTR4_FS_STATUS          = 61;
/// const FATTR4_FS_LAYOUT_TYPES    = 62;
/// const FATTR4_LAYOUT_HINT        = 63;
/// const FATTR4_LAYOUT_TYPES       = 64;
/// const FATTR4_LAYOUT_BLKSIZE     = 65;
/// const FATTR4_LAYOUT_ALIGNMENT   = 66;
/// const FATTR4_FS_LOCATIONS_INFO  = 67;
/// const FATTR4_MDSTHRESHOLD       = 68;
/// const FATTR4_RETENTION_GET      = 69;
/// const FATTR4_RETENTION_SET      = 70;
/// const FATTR4_RETENTEVT_GET      = 71;
/// const FATTR4_RETENTEVT_SET      = 72;
/// const FATTR4_RETENTION_HOLD     = 73;
/// const FATTR4_MODE_SET_MASKED    = 74;
/// const FATTR4_FS_CHARSET_CAP     = 76;
/// const FATTR4_SPACE_RESERVED     = 77;
/// const FATTR4_SPACE_FREED        = 78;
/// const FATTR4_MAX_HOLE_PUNCHED   = 79;
/// const FATTR4_CHANGE_ATTR_TYPE   = 80;
///
/// /*
///  * File attribute container
///  */
/// struct fattr4 {
///         bitmap4         attrmask;
///         attrlist4       attr_vals;
/// };
///
/// /*
///  * Change info for the client
///  */
/// struct change_info4 {
///         bool            atomic;
///         changeid4       before;
///         changeid4       after;
/// };
///
/// typedef netaddr4 clientaddr4;
///
/// /*
///  * Callback program info as provided by the client
///  */
/// struct cb_client4 {
///         uint32_t        cb_program;
///         netaddr4        cb_location;
/// };
```

```
///
/// /*
///  * NFSv4.0 Long Hand Client ID
///  */
/// struct nfs_client_id4 {
///         verifier4       verifier;
///         opaque          id<NFS4_OPAQUE_LIMIT>;
/// };
///
/// /*
///  * NFSv4.1 Client Owner (aka long hand client ID)
///  */
/// struct client_owner4 {
///         verifier4       co_verifier;
///         opaque          co_ownerid<NFS4_OPAQUE_LIMIT>;
/// };
///
///
/// /*
///  * NFSv4.1 server Owner
///  */
/// struct server_owner4 {
///  uint64_t       so_minor_id;
///  opaque         so_major_id<NFS4_OPAQUE_LIMIT>;
/// };
///
///
/// struct state_owner4 {
///         clientid4       clientid;
///         opaque          owner<NFS4_OPAQUE_LIMIT>;
/// };
///
/// typedef state_owner4 open_owner4;
/// typedef state_owner4 lock_owner4;
///
///
/// enum nfs_lock_type4 {
///         READ_LT         = 1,
///         WRITE_LT        = 2,
///         READW_LT        = 3,    /* blocking read */
///         WRITEW_LT       = 4     /* blocking write */
/// };
///
///
/// %
/// %/* Input for computing subkeys */
/// enum ssv_subkey4 {
///         SSV4_SUBKEY_MIC_I2T     = 1,
```

```
///         SSV4_SUBKEY_MIC_T2I     = 2,
///         SSV4_SUBKEY_SEAL_I2T    = 3,
///         SSV4_SUBKEY_SEAL_T2I    = 4
/// };
/// %
///
/// %
/// %/* Input for computing smt_hmac */
/// struct ssv_mic_plain_tkn4 {
///   uint32_t        smpt_ssv_seq;
///   opaque          smpt_orig_plain<>;
/// };
/// %
///
/// %
/// %/* SSV GSS PerMsgToken token */
/// struct ssv_mic_tkn4 {
///   uint32_t        smt_ssv_seq;
///   opaque          smt_hmac<>;
/// };
/// %
///
/// %
/// %/* Input for computing ssct_encr_data and ssct_hmac */
/// struct ssv_seal_plain_tkn4 {
///   opaque          sspt_confounder<>;
///   uint32_t        sspt_ssv_seq;
///   opaque          sspt_orig_plain<>;
///   opaque          sspt_pad<>;
/// };
/// %
///
/// %
/// %/* SSV GSS SealedMessage token */
/// struct ssv_seal_cipher_tkn4 {
///   uint32_t      ssct_ssv_seq;
///   opaque        ssct_iv<>;
///   opaque        ssct_encr_data<>;
///   opaque        ssct_hmac<>;
/// };
/// %
///
/// /*
///  * Defines an individual server replica
///  */
/// struct  fs_locations_server4 {
///         int32_t         fls_currency;
///         opaque          fls_info<>;
```

```
///         utf8str_cis     fls_server;
/// };
///
/// /*
///  * Byte indices of items within
///  * fls_info: flag fields, class numbers,
///  * bytes indicating ranks and orders.
///  */
/// const FSLI4BX_GFLAGS            = 0;
/// const FSLI4BX_TFLAGS            = 1;
///
/// const FSLI4BX_CLSIMUL           = 2;
/// const FSLI4BX_CLHANDLE          = 3;
/// const FSLI4BX_CLFILEID          = 4;
/// const FSLI4BX_CLWRITEVER        = 5;
/// const FSLI4BX_CLCHANGE          = 6;
/// const FSLI4BX_CLREADDIR         = 7;
///
/// const FSLI4BX_READRANK          = 8;
/// const FSLI4BX_WRITERANK         = 9;
/// const FSLI4BX_READORDER         = 10;
/// const FSLI4BX_WRITEORDER        = 11;
///
/// /*
///  * Bits defined within the general flag byte.
///  */
/// const FSLI4GF_WRITABLE          = 0x01;
/// const FSLI4GF_CUR_REQ           = 0x02;
/// const FSLI4GF_ABSENT            = 0x04;
/// const FSLI4GF_GOING             = 0x08;
/// const FSLI4GF_SPLIT             = 0x10;
///
/// /*
///  * Bits defined within the transport flag byte.
///  */
/// const FSLI4TF_RDMA              = 0x01;
///
/// /*
///  * Defines a set of replicas sharing
///  * a common value of the root path
///  * with in the corresponding
///  * single-server namespaces.
///  */
/// struct  fs_locations_item4 {
///         fs_locations_server4    fli_entries<>;
///         pathname4               fli_rootpath;
/// };
///
```

```
/// /*
///  * Defines the overall structure of
///  * the fs_locations_info attribute.
///  */
/// struct  fs_locations_info4 {
///         uint32_t                fli_flags;
///         int32_t                 fli_valid_for;
///         pathname4               fli_fs_root;
///         fs_locations_item4      fli_items<>;
/// };
///
/// /*
///  * Flag bits in fli_flags.
///  */
/// const FSLI4IF_VAR_SUB           = 0x00000001;
///
/// typedef fs_locations_info4 fattr4_fs_locations_info;
///
/// const NFL4_UFLG_MASK            = 0x0000003F;
/// const NFL4_UFLG_DENSE           = 0x00000001;
/// const NFL4_UFLG_COMMIT_THRU_MDS = 0x00000002;
/// const NFL4_UFLG_STRIPE_UNIT_SIZE_MASK
///                                 = 0xFFFFFFC0;
///
/// typedef uint32_t nfl_util4;
///
/// %
///
/// enum filelayout_hint_care4 {
///         NFLH4_CARE_DENSE        = NFL4_UFLG_DENSE,
///
///         NFLH4_CARE_COMMIT_THRU_MDS
///                                 = NFL4_UFLG_COMMIT_THRU_MDS,
///
///         NFLH4_CARE_STRIPE_UNIT_SIZE
///                                 = 0x00000040,
///
///         NFLH4_CARE_STRIPE_COUNT = 0x00000080
/// };
/// %
/// %/*
/// % * Encoded in the loh_body field of data type layouthint4:
/// %*/
/// %
/// struct nfsv4_1_file_layouthint4 {
///         uint32_t        nflh_care;
///         nfl_util4       nflh_util;
///         count4          nflh_stripe_count;
```

```
/// };
///
/// %
///
/// %
/// typedef netaddr4 multipath_list4<>;
/// %
/// %/*
/// % * Encoded in the da_addr_body field of data type device_addr4:
/// %*/
/// struct nfsv4_1_file_layout_ds_addr4 {
///         uint32_t        nflda_stripe_indices<>;
///         multipath_list4 nflda_multipath_ds_list<>;
/// };
///
/// %
///
/// %
/// %/*
/// % * Encoded in the loc_body field of data type layout_content4:
/// %*/
/// struct nfsv4_1_file_layout4 {
///          deviceid4      nfl_deviceid;
///          nfl_util4      nfl_util;
///          uint32_t       nfl_first_stripe_index;
///          offset4        nfl_pattern_offset;
///          nfs_fh4        nfl_fh_list<>;
/// };
///
/// %
///
/// %/*
/// % * Encoded in the lou_body field of data type layoutupdate4:
/// % *      Nothing. lou_body is a zero length array of bytes.
/// % */
/// %
///
/// %/*
/// % * Encoded in the lrf_body field of
/// % * data type layoutreturn_file4:
/// % *      Nothing. lrf_body is a zero length array of bytes.
/// % */
/// %
///
///
/// 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:
///          /* new CURRENTFH: created object */
///          CREATE4resok resok4;
///  default:
///          void;
/// };
///
/// struct DELEGPURGE4args {
///         clientid4       clientid;
/// };
///
/// struct DELEGPURGE4res {
///         nfsstat4        status;
/// };
///
/// struct DELEGRETURN4args {
///         /* CURRENT_FH: delegated object */
///         stateid4        deleg_stateid;
```

```
/// };
///
/// struct DELEGRETURN4res {
///         nfsstat4        status;
/// };
///
/// struct GETATTR4args {
///         /* CURRENT_FH: object */
///         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_stateid and lock_owner
///  * to a lock stateid.
///  */
/// struct open_to_lock_owner4 {
///         seqid4          open_seqid;
///         stateid4        open_stateid;
///         seqid4          lock_seqid;
///         lock_owner4     lock_owner;
/// };
///
/// /*
///  * For LOCK, existing lock stateid  continues to request new
///  * file lock for the same lock_owner and open_stateid.
///  */
/// 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 {
///         /* New CURRENT_FH: object */
///         nfsstat4        status;
/// };
///
/// struct LOOKUPP4res {
///         /* new CURRENT_FH: parent directory */
///         nfsstat4        status;
/// };
///
/// struct NVERIFY4args {
///         /* CURRENT_FH: object */
///         fattr4          obj_attributes;
/// };
///
/// struct NVERIFY4res {
///         nfsstat4        status;
/// };
///
/// /*
///  * Various definitions for OPEN
///  */
/// enum createmode4 {
///         UNCHECKED4      = 0,
///         GUARDED4        = 1,
///         /* Deprecated in NFSv4.1. */
///         EXCLUSIVE4      = 2,
///         /*
///          * New to NFSv4.1. If session is persistent,
///          * GUARDED4 MUST be used. Otherwise, use
///          * EXCLUSIVE4_1 instead of EXCLUSIVE4.
///          */
///         EXCLUSIVE4_1    = 3
/// };
///
/// struct creatverfattr {
///          verifier4      cva_verf;
///          fattr4         cva_attrs;
/// };
///
/// union createhow4 switch (createmode4 mode) {
///  case UNCHECKED4:
///  case GUARDED4:
///          fattr4         createattrs;
///  case EXCLUSIVE4:
```

```
///          verifier4      createverf;
///  case EXCLUSIVE4_1:
///          creatverfattr  ch_createboth;
/// };
///
/// 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;
/// } ;
///
/// /*
///  * Share Access and Deny constants for open argument
///  */
/// 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;
///
///
/// /* new flags for share_access field of OPEN4args */
/// const OPEN4_SHARE_ACCESS_WANT_DELEG_MASK        = 0xFF00;
/// const OPEN4_SHARE_ACCESS_WANT_NO_PREFERENCE     = 0x0000;
/// const OPEN4_SHARE_ACCESS_WANT_READ_DELEG        = 0x0100;
/// const OPEN4_SHARE_ACCESS_WANT_WRITE_DELEG       = 0x0200;
/// const OPEN4_SHARE_ACCESS_WANT_ANY_DELEG         = 0x0300;
/// const OPEN4_SHARE_ACCESS_WANT_NO_DELEG          = 0x0400;
/// const OPEN4_SHARE_ACCESS_WANT_CANCEL            = 0x0500;
///
/// const
///  OPEN4_SHARE_ACCESS_WANT_SIGNAL_DELEG_WHEN_RESRC_AVAIL
///  = 0x10000;
///
/// const
///  OPEN4_SHARE_ACCESS_WANT_PUSH_DELEG_WHEN_UNCONTENDED
///  = 0x20000;
///
/// enum open_delegation_type4 {
///         OPEN_DELEGATE_NONE      = 0,
///         OPEN_DELEGATE_READ      = 1,
///         OPEN_DELEGATE_WRITE     = 2,
///         OPEN_DELEGATE_NONE_EXT  = 3 /* new to v4.1 */
/// };
///
/// enum open_claim_type4 {
///         /*
///          * Not a reclaim.
///          */
///         CLAIM_NULL              = 0,
///
///         CLAIM_PREVIOUS          = 1,
///         CLAIM_DELEGATE_CUR      = 2,
///         CLAIM_DELEGATE_PREV     = 3,
///
///         /*
///          * Not a reclaim.
///          *
///          * Like CLAIM_NULL, but object identified
///          * by the current filehandle.
///          */
///         CLAIM_FH                = 4, /* new to v4.1 */
///
///         /*
///          * Like CLAIM_DELEGATE_CUR, but object identified
///          * by current filehandle.
```

```
///          */
///         CLAIM_DELEG_CUR_FH      = 5, /* new to v4.1 */
///
///         /*
///          * Like CLAIM_DELEGATE_PREV, but object identified
///          * by current filehandle.
///          */
///         CLAIM_DELEG_PREV_FH     = 6 /* new to v4.1 */
/// };
///
/// 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;
///
///  /*
///   * Like CLAIM_NULL.  No special rights
///   * to file.  Ordinary OPEN of the
///   * specified file by current filehandle.
///   */
///  case CLAIM_FH: /* new to v4.1 */
///         /* CURRENT_FH: regular file to open */
///         void;
///
///  /*
///   * Like CLAIM_DELEGATE_PREV.  Right to file based on a
///   * delegation granted to a previous boot
///   * instance of the client.  File is identified by
///   * by filehandle.
///   */
///  case CLAIM_DELEG_PREV_FH: /* new to v4.1 */
///         /* CURRENT_FH: file being opened */
///         void;
///
///  /*
///   * Like CLAIM_DELEGATE_CUR.  Right to file based on
///   * a delegation granted by the server.
///   * File is identified by filehandle.
///   */
///  case CLAIM_DELEG_CUR_FH: /* new to v4.1 */
///          /* CURRENT_FH: file being opened */
///          stateid4       oc_delegate_stateid;
///
/// };
///
/// /*
///  * 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. */
/// };
///
///
/// enum why_no_delegation4 { /* new to v4.1 */
///         WND4_NOT_WANTED         = 0,
///         WND4_CONTENTION         = 1,
///         WND4_RESOURCE           = 2,
///         WND4_NOT_SUPP_FTYPE     = 3,
///         WND4_WRITE_DELEG_NOT_SUPP_FTYPE = 4,
///         WND4_NOT_SUPP_UPGRADE   = 5,
///         WND4_NOT_SUPP_DOWNGRADE = 6,
///         WND4_CANCELLED          = 7,
///         WND4_IS_DIR             = 8
/// };
///
/// union open_none_delegation4 /* new to v4.1 */
/// switch (why_no_delegation4 ond_why) {
///         case WND4_CONTENTION:
///                 bool ond_server_will_push_deleg;
///         case WND4_RESOURCE:
///                 bool ond_server_will_signal_avail;
///         default:
```

```
///                 void;
/// };
///
/// 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;
///         case OPEN_DELEGATE_NONE_EXT: /* new to v4.1 */
///                 open_none_delegation4 od_whynone;
/// };
///
/// /*
///  * 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;
/// /* Server will preserve file if removed while open */
/// const OPEN4_RESULT_PRESERVE_UNLINKED = 0x00000008;
///
/// /*
///  * Server may use CB_NOTIFY_LOCK on locks
///  * derived from this open
///  */
/// const OPEN4_RESULT_MAY_NOTIFY_LOCK = 0x00000020;
///
/// 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:
///         /* New CURRENT_FH: opened file */
///         OPEN4resok      resok4;
///  default:
///         void;
/// };
```

```
///
/// struct OPENATTR4args {
///         /* CURRENT_FH: object */
///         bool    createdir;
/// };
///
/// struct OPENATTR4res {
///         /*
///          * If status is NFS4_OK,
///          *   new CURRENT_FH: named attribute
///          *                   directory
///          */
///         nfsstat4        status;
/// };
///
/// /* obsolete in NFSv4.1 */
/// 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 {
///         /*
///          * If status is NFS4_OK,
///          *    new CURRENT_FH: argument to PUTFH
///          */
///         nfsstat4        status;
/// };
///
/// struct PUTPUBFH4res {
///         /*
///          * If status is NFS4_OK,
///          *   new CURRENT_FH: public fh
///          */
///         nfsstat4        status;
/// };
///
/// struct PUTROOTFH4res {
///         /*
///          * If status is NFS4_OK,
///          *   new 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;
/// };
///
/// /* Obsolete in NFSv4.1 */
/// struct RENEW4args {
///         clientid4       clientid;
/// };
///
/// struct RENEW4res {
///         nfsstat4        status;
/// };
///
/// struct RESTOREFH4res {
///         /*
///          * If status is NFS4_OK,
```

```
///          *     new CURRENT_FH: value of saved fh
///          */
///         nfsstat4        status;
/// };
///
/// struct SAVEFH4res {
///         /*
///          * If status is NFS4_OK,
///          *    new 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:
///         /* CURRENTFH: consumed */
///          SECINFO4resok resok4;
///  default:
```

```
///          void;
/// };
///
/// struct SETATTR4args {
///         /* CURRENT_FH: target object */
///         stateid4        stateid;
///         fattr4          obj_attributes;
/// };
///
/// struct SETATTR4res {
///         nfsstat4        status;
///         bitmap4         attrsset;
/// };
///
/// /* Obsolete in NFSv4.1 */
/// 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;
/// };
///
/// /* Obsolete in NFSv4.1 */
/// 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;
/// };
///
/// /* Obsolete in NFSv4.1 */
/// struct RELEASE_LOCKOWNER4args {
///         lock_owner4     lock_owner;
/// };
///
/// struct RELEASE_LOCKOWNER4res {
///         nfsstat4        status;
/// };
///
/// struct ILLEGAL4res {
///         nfsstat4        status;
/// };
///
/// typedef opaque gsshandle4_t<>;
///
```

```
/// struct gss_cb_handles4 {
///         rpc_gss_svc_t           gcbp_service; /* RFC 2203 */
///         gsshandle4_t            gcbp_handle_from_server;
///         gsshandle4_t            gcbp_handle_from_client;
/// };
///
/// union callback_sec_parms4 switch (uint32_t cb_secflavor) {
/// case AUTH_NONE:
///         void;
/// case AUTH_SYS:
///         authsys_parms   cbsp_sys_cred; /* RFC 1831 */
/// case RPCSEC_GSS:
///         gss_cb_handles4 cbsp_gss_handles;
/// };
///
/// struct BACKCHANNEL_CTL4args {
///         uint32_t                bca_cb_program;
///         callback_sec_parms4     bca_sec_parms<>;
/// };
///
/// struct BACKCHANNEL_CTL4res {
///         nfsstat4                bcr_status;
/// };
///
/// enum channel_dir_from_client4 {
///  CDFC4_FORE             = 0x1,
///  CDFC4_BACK             = 0x2,
///  CDFC4_FORE_OR_BOTH     = 0x3,
///  CDFC4_BACK_OR_BOTH     = 0x7
/// };
///
/// struct BIND_CONN_TO_SESSION4args {
///  sessionid4     bctsa_sessid;
///
///  channel_dir_from_client4
///                 bctsa_dir;
///
///  bool           bctsa_use_conn_in_rdma_mode;
/// };
///
/// enum channel_dir_from_server4 {
///  CDFS4_FORE     = 0x1,
///  CDFS4_BACK     = 0x2,
///  CDFS4_BOTH     = 0x3
/// };
///
/// struct BIND_CONN_TO_SESSION4resok {
///  sessionid4     bctsr_sessid;
```

```
///
///  channel_dir_from_server4
///                 bctsr_dir;
///
///  bool           bctsr_use_conn_in_rdma_mode;
/// };
///
/// union BIND_CONN_TO_SESSION4res
///  switch (nfsstat4 bctsr_status) {
///
///  case NFS4_OK:
///   BIND_CONN_TO_SESSION4resok
///                 bctsr_resok4;
///
///  default:       void;
/// };
///
/// const EXCHGID4_FLAG_SUPP_MOVED_REFER    = 0x00000001;
/// const EXCHGID4_FLAG_SUPP_MOVED_MIGR     = 0x00000002;
///
/// const EXCHGID4_FLAG_BIND_PRINC_STATEID  = 0x00000100;
///
/// const EXCHGID4_FLAG_USE_NON_PNFS        = 0x00010000;
/// const EXCHGID4_FLAG_USE_PNFS_MDS        = 0x00020000;
/// const EXCHGID4_FLAG_USE_PNFS_DS         = 0x00040000;
///
/// const EXCHGID4_FLAG_MASK_PNFS           = 0x00070000;
///
/// const EXCHGID4_FLAG_UPD_CONFIRMED_REC_A = 0x40000000;
/// const EXCHGID4_FLAG_CONFIRMED_R         = 0x80000000;
///
/// struct state_protect_ops4 {
///         bitmap4 spo_must_enforce;
///         bitmap4 spo_must_allow;
/// };
///
/// struct ssv_sp_parms4 {
///         state_protect_ops4      ssp_ops;
///         sec_oid4                ssp_hash_algs<>;
///         sec_oid4                ssp_encr_algs<>;
///         uint32_t                ssp_window;
///         uint32_t                ssp_num_gss_handles;
/// };
///
/// enum state_protect_how4 {
///         SP4_NONE = 0,
///         SP4_MACH_CRED = 1,
///         SP4_SSV = 2
```

```
/// };
///
/// union state_protect4_a switch(state_protect_how4 spa_how) {
///         case SP4_NONE:
///                 void;
///         case SP4_MACH_CRED:
///                 state_protect_ops4      spa_mach_ops;
///         case SP4_SSV:
///                 ssv_sp_parms4           spa_ssv_parms;
/// };
///
/// struct EXCHANGE_ID4args {
///         client_owner4           eia_clientowner;
///         uint32_t                eia_flags;
///         state_protect4_a        eia_state_protect;
///         nfs_impl_id4            eia_client_impl_id<1>;
/// };
///
/// struct ssv_prot_info4 {
///  state_protect_ops4     spi_ops;
///  uint32_t               spi_hash_alg;
///  uint32_t               spi_encr_alg;
///  uint32_t               spi_ssv_len;
///  uint32_t               spi_window;
///  gsshandle4_t           spi_handles<>;
/// };
///
/// union state_protect4_r switch(state_protect_how4 spr_how) {
///  case SP4_NONE:
///          void;
///  case SP4_MACH_CRED:
///          state_protect_ops4     spr_mach_ops;
///  case SP4_SSV:
///          ssv_prot_info4         spr_ssv_info;
/// };
///
/// struct EXCHANGE_ID4resok {
///  clientid4        eir_clientid;
///  sequenceid4      eir_sequenceid;
///  uint32_t         eir_flags;
///  state_protect4_r eir_state_protect;
///  server_owner4    eir_server_owner;
///  opaque           eir_server_scope<NFS4_OPAQUE_LIMIT>;
///  nfs_impl_id4     eir_server_impl_id<1>;
/// };
///
/// union EXCHANGE_ID4res switch (nfsstat4 eir_status) {
/// case NFS4_OK:
```

```
///  EXCHANGE_ID4resok      eir_resok4;
///
/// default:
///  void;
/// };
///
/// struct channel_attrs4 {
///         count4                  ca_headerpadsize;
///         count4                  ca_maxrequestsize;
///         count4                  ca_maxresponsesize;
///         count4                  ca_maxresponsesize_cached;
///         count4                  ca_maxoperations;
///         count4                  ca_maxrequests;
///         uint32_t                ca_rdma_ird<1>;
/// };
///
/// const CREATE_SESSION4_FLAG_PERSIST              = 0x00000001;
/// const CREATE_SESSION4_FLAG_CONN_BACK_CHAN       = 0x00000002;
/// const CREATE_SESSION4_FLAG_CONN_RDMA            = 0x00000004;
///
/// struct CREATE_SESSION4args {
///         clientid4               csa_clientid;
///         sequenceid4             csa_sequence;
///
///         uint32_t                csa_flags;
///
///         channel_attrs4          csa_fore_chan_attrs;
///         channel_attrs4          csa_back_chan_attrs;
///
///         uint32_t                csa_cb_program;
///         callback_sec_parms4     csa_sec_parms<>;
/// };
///
/// struct CREATE_SESSION4resok {
///         sessionid4              csr_sessionid;
///         sequenceid4             csr_sequence;
///
///         uint32_t                csr_flags;
///
///         channel_attrs4          csr_fore_chan_attrs;
///         channel_attrs4          csr_back_chan_attrs;
/// };
///
/// union CREATE_SESSION4res switch (nfsstat4 csr_status) {
/// case NFS4_OK:
///         CREATE_SESSION4resok    csr_resok4;
/// default:
///         void;
```

```
/// };
///
/// struct DESTROY_SESSION4args {
///         sessionid4      dsa_sessionid;
/// };
///
/// struct DESTROY_SESSION4res {
///         nfsstat4        dsr_status;
/// };
///
/// struct FREE_STATEID4args {
///         stateid4        fsa_stateid;
/// };
///
/// struct FREE_STATEID4res {
///         nfsstat4        fsr_status;
/// };
///
///
/// typedef nfstime4 attr_notice4;
///
/// struct GET_DIR_DELEGATION4args {
///         /* CURRENT_FH: delegated directory */
///         bool            gdda_signal_deleg_avail;
///         bitmap4         gdda_notification_types;
///         attr_notice4    gdda_child_attr_delay;
///         attr_notice4    gdda_dir_attr_delay;
///         bitmap4         gdda_child_attributes;
///         bitmap4         gdda_dir_attributes;
/// };
/// struct GET_DIR_DELEGATION4resok {
///         verifier4       gddr_cookieverf;
///         /* Stateid for get_dir_delegation */
///         stateid4        gddr_stateid;
///         /* Which notifications can the server support */
///         bitmap4         gddr_notification;
///         bitmap4         gddr_child_attributes;
///         bitmap4         gddr_dir_attributes;
/// };
///
/// enum gddrnf4_status {
///         GDD4_OK         = 0,
///         GDD4_UNAVAIL    = 1
/// };
///
/// union GET_DIR_DELEGATION4res_non_fatal
///  switch (gddrnf4_status gddrnf_status) {
///  case GDD4_OK:
```

```
///   GET_DIR_DELEGATION4resok      gddrnf_resok4;
///  case GDD4_UNAVAIL:
///   bool                          gddrnf_will_signal_deleg_avail;
/// };
///
/// union GET_DIR_DELEGATION4res
///  switch (nfsstat4 gddr_status) {
///  case NFS4_OK:
///   GET_DIR_DELEGATION4res_non_fatal      gddr_res_non_fatal4;
///  default:
///   void;
/// };
///
/// struct GETDEVICEINFO4args {
///         deviceid4       gdia_device_id;
///         layouttype4     gdia_layout_type;
///         count4          gdia_maxcount;
///         bitmap4         gdia_notify_types;
/// };
///
/// struct GETDEVICEINFO4resok {
///         device_addr4    gdir_device_addr;
///         bitmap4         gdir_notification;
/// };
///
/// union GETDEVICEINFO4res switch (nfsstat4 gdir_status) {
/// case NFS4_OK:
///         GETDEVICEINFO4resok     gdir_resok4;
/// case NFS4ERR_TOOSMALL:
///         count4                  gdir_mincount;
/// default:
///         void;
/// };
///
/// struct GETDEVICELIST4args {
///         /* CURRENT_FH: object belonging to the file system */
///         layouttype4     gdla_layout_type;
///
///         /* number of deviceIDs to return */
///         count4          gdla_maxdevices;
///
///         nfs_cookie4     gdla_cookie;
///         verifier4       gdla_cookieverf;
/// };
///
/// struct GETDEVICELIST4resok {
///         nfs_cookie4             gdlr_cookie;
///         verifier4               gdlr_cookieverf;
```

```
///         deviceid4               gdlr_deviceid_list<>;
///         bool                    gdlr_eof;
/// };
///
/// union GETDEVICELIST4res switch (nfsstat4 gdlr_status) {
/// case NFS4_OK:
///         GETDEVICELIST4resok     gdlr_resok4;
/// default:
///         void;
/// };
///
/// union newtime4 switch (bool nt_timechanged) {
/// case TRUE:
///         nfstime4           nt_time;
/// case FALSE:
///         void;
/// };
///
/// union newoffset4 switch (bool no_newoffset) {
/// case TRUE:
///         offset4           no_offset;
/// case FALSE:
///         void;
/// };
///
/// struct LAYOUTCOMMIT4args {
///         /* CURRENT_FH: file */
///         offset4                 loca_offset;
///         length4                 loca_length;
///         bool                    loca_reclaim;
///         stateid4                loca_stateid;
///         newoffset4              loca_last_write_offset;
///         newtime4                loca_time_modify;
///         layoutupdate4           loca_layoutupdate;
/// };
/// union newsize4 switch (bool ns_sizechanged) {
/// case TRUE:
///         length4         ns_size;
/// case FALSE:
///         void;
/// };
///
/// struct LAYOUTCOMMIT4resok {
///         newsize4                locr_newsize;
/// };
///
/// union LAYOUTCOMMIT4res switch (nfsstat4 locr_status) {
/// case NFS4_OK:
```

```
///         LAYOUTCOMMIT4resok      locr_resok4;
/// default:
///         void;
/// };
///
/// struct LAYOUTGET4args {
///         /* CURRENT_FH: file */
///         bool                    loga_signal_layout_avail;
///         layouttype4             loga_layout_type;
///         layoutiomode4           loga_iomode;
///         offset4                 loga_offset;
///         length4                 loga_length;
///         length4                 loga_minlength;
///         stateid4                loga_stateid;
///         count4                  loga_maxcount;
/// };
/// struct LAYOUTGET4resok {
///         bool               logr_return_on_close;
///         stateid4           logr_stateid;
///         layout4            logr_layout<>;
/// };
///
/// union LAYOUTGET4res switch (nfsstat4 logr_status) {
/// case NFS4_OK:
///         LAYOUTGET4resok     logr_resok4;
/// case NFS4ERR_LAYOUTTRYLATER:
///         bool                logr_will_signal_layout_avail;
/// default:
///         void;
/// };
///
///
/// struct LAYOUTRETURN4args {
///         /* CURRENT_FH: file */
///         bool                    lora_reclaim;
///         layouttype4             lora_layout_type;
///         layoutiomode4           lora_iomode;
///         layoutreturn4           lora_layoutreturn;
/// };
///
///
/// union layoutreturn_stateid switch (bool lrs_present) {
/// case TRUE:
///         stateid4                lrs_stateid;
/// case FALSE:
///         void;
/// };
///
```

```
/// union LAYOUTRETURN4res switch (nfsstat4 lorr_status) {
/// case NFS4_OK:
///         layoutreturn_stateid    lorr_stateid;
/// default:
///         void;
/// };
///
/// enum secinfo_style4 {
///         SECINFO_STYLE4_CURRENT_FH       = 0,
///         SECINFO_STYLE4_PARENT           = 1
/// };
///
/// /* CURRENT_FH: object or child directory */
/// typedef secinfo_style4 SECINFO_NO_NAME4args;
///
/// /* CURRENTFH: consumed if status is NFS4_OK */
/// typedef SECINFO4res SECINFO_NO_NAME4res;
///
/// struct SEQUENCE4args {
///         sessionid4     sa_sessionid;
///         sequenceid4    sa_sequenceid;
///         slotid4        sa_slotid;
///         slotid4        sa_highest_slotid;
///         bool           sa_cachethis;
/// };
///
/// const SEQ4_STATUS_CB_PATH_DOWN                  = 0x00000001;
/// const SEQ4_STATUS_CB_GSS_CONTEXTS_EXPIRING      = 0x00000002;
/// const SEQ4_STATUS_CB_GSS_CONTEXTS_EXPIRED       = 0x00000004;
/// const SEQ4_STATUS_EXPIRED_ALL_STATE_REVOKED     = 0x00000008;
/// const SEQ4_STATUS_EXPIRED_SOME_STATE_REVOKED    = 0x00000010;
/// const SEQ4_STATUS_ADMIN_STATE_REVOKED           = 0x00000020;
/// const SEQ4_STATUS_RECALLABLE_STATE_REVOKED      = 0x00000040;
/// const SEQ4_STATUS_LEASE_MOVED                   = 0x00000080;
/// const SEQ4_STATUS_RESTART_RECLAIM_NEEDED        = 0x00000100;
/// const SEQ4_STATUS_CB_PATH_DOWN_SESSION          = 0x00000200;
/// const SEQ4_STATUS_BACKCHANNEL_FAULT             = 0x00000400;
/// const SEQ4_STATUS_DEVID_CHANGED                 = 0x00000800;
/// const SEQ4_STATUS_DEVID_DELETED                 = 0x00001000;
///
/// struct SEQUENCE4resok {
///         sessionid4      sr_sessionid;
///         sequenceid4     sr_sequenceid;
///         slotid4         sr_slotid;
///         slotid4         sr_highest_slotid;
///         slotid4         sr_target_highest_slotid;
///         uint32_t        sr_status_flags;
/// };
```

```
///
/// union SEQUENCE4res switch (nfsstat4 sr_status) {
/// case NFS4_OK:
///         SEQUENCE4resok  sr_resok4;
/// default:
///         void;
/// };
///
/// struct ssa_digest_input4 {
///         SEQUENCE4args sdi_seqargs;
/// };
///
/// struct SET_SSV4args {
///         opaque          ssa_ssv<>;
///         opaque          ssa_digest<>;
/// };
///
/// struct ssr_digest_input4 {
///         SEQUENCE4res sdi_seqres;
/// };
///
/// struct SET_SSV4resok {
///         opaque          ssr_digest<>;
/// };
///
/// union SET_SSV4res switch (nfsstat4 ssr_status) {
/// case NFS4_OK:
///         SET_SSV4resok   ssr_resok4;
/// default:
///         void;
/// };
///
/// struct TEST_STATEID4args {
///         stateid4        ts_stateids<>;
/// };
///
/// struct TEST_STATEID4resok {
///         nfsstat4        tsr_status_codes<>;
/// };
///
/// union TEST_STATEID4res switch (nfsstat4 tsr_status) {
///     case NFS4_OK:
///         TEST_STATEID4resok tsr_resok4;
///     default:
///         void;
/// };
///
/// union deleg_claim4 switch (open_claim_type4 dc_claim) {
```

```
/// /*
///  * No special rights to object.  Ordinary delegation
///  * request of the specified object.  Object identified
///  * by filehandle.
///  */
/// case CLAIM_FH: /* new to v4.1 */
///         /* CURRENT_FH: object being delegated */
///         void;
///
/// /*
///  * Right to file based on a delegation granted
///  * to a previous boot instance of the client.
///  * File is specified by filehandle.
///  */
/// case CLAIM_DELEG_PREV_FH: /* new to v4.1 */
///         /* CURRENT_FH: object being delegated */
///         void;
///
/// /*
///  * Right to the file established by an open previous
///  * to server reboot.  File identified by filehandle.
///  * Used during server reclaim grace period.
///  */
/// case CLAIM_PREVIOUS:
///         /* CURRENT_FH: object being reclaimed */
///         open_delegation_type4   dc_delegate_type;
/// };
///
/// struct WANT_DELEGATION4args {
///         uint32_t        wda_want;
///         deleg_claim4    wda_claim;
/// };
///
/// union WANT_DELEGATION4res switch (nfsstat4 wdr_status) {
/// case NFS4_OK:
///         open_delegation4 wdr_resok4;
/// default:
///         void;
/// };
///
/// struct DESTROY_CLIENTID4args {
///         clientid4       dca_clientid;
/// };
///
/// struct DESTROY_CLIENTID4res {
///         nfsstat4        dcr_status;
/// };
///
```

```
/// struct RECLAIM_COMPLETE4args {
///         /*
///          * If rca_one_fs TRUE,
///          *
///          *    CURRENT_FH: object in
///          *    filesystem reclaim is
///          *    complete for.
///          */
///         bool            rca_one_fs;
/// };
///
/// struct RECLAIM_COMPLETE4res {
///         nfsstat4        rcr_status;
/// };
///
///
/// const COPY4_GUARDED     = 0x00000001;
/// const COPY4_METADATA    = 0x00000002;
///
/// struct COPY4args {
///         /* SAVED_FH: source file */
///         /* CURRENT_FH: destination file or */
///         /*             directory           */
///         offset4         ca_src_offset;
///         offset4         ca_dst_offset;
///         length4         ca_count;
///         uint32_t        ca_flags;
///         component4      ca_destination;
///         netloc4         ca_source_server<>;
/// };
///
/// union COPY4res switch (nfsstat4 cr_status) {
///         case NFS4_OK:
///                 stateid4        cr_callback_id<1>;
///         default:
///                 length4         cr_bytes_copied;
/// };
///
/// struct COPY_ABORT4args {
///         /* CURRENT_FH: desination file */
///         stateid4        caa_stateid;
/// };
///
/// struct COPY_ABORT4res {
///         nfsstat4        car_status;
/// };
///
/// struct COPY_NOTIFY4args {
```

```
///         /* CURRENT_FH: source file */
///         netloc4         cna_destination_server;
/// };
///
/// struct COPY_NOTIFY4resok {
///         nfstime4        cnr_lease_time;
///         netloc4         cnr_source_server<>;
/// };
///
/// union COPY_NOTIFY4res switch (nfsstat4 cnr_status) {
///         case NFS4_OK:
///                 COPY_NOTIFY4resok       resok4;
///         default:
///                 void;
/// };
///
/// struct COPY_REVOKE4args {
///         /* CURRENT_FH: source file */
///         netloc4         cra_destination_server;
/// };
///
/// struct COPY_REVOKE4res {
///         nfsstat4        crr_status;
/// };
///
///
/// struct COPY_STATUS4args {
///         /* CURRENT_FH: destination file */
///         stateid4        csa_stateid;
/// };
///
/// struct COPY_STATUS4resok {
///         length4         csr_bytes_copied;
///         nfsstat4        csr_complete<1>;
/// };
///
/// union COPY_STATUS4res switch (nfsstat4 csr_status) {
///         case NFS4_OK:
///                 COPY_STATUS4resok       resok4;
///         default:
///                 void;
/// };
///
///
/// struct HOLE_PUNCH4args {
///         /* CURRENT_FH: file */
///         offset4        hpa_offset;
///         length4        hpa_count;
```

```
/// };
///
///
/// struct HOLE_PUNCH4res {
///         nfsstat4        hpr_status;
/// };
///
/// struct READPLUS4args {
///         /* CURRENT_FH: file */
///         stateid4        rpa_stateid;
///         offset4         rpa_offset;
///         count4          rpa_count;
/// };
/// struct ressdplus_hole_info {
///         offset4         rphi_offset;
///         length4         rphi_length;
/// };
///
/// enum holeres4 {
///         HOLE_NOINFO = 0,
///         HOLE_INFO = 1
/// };
///
/// union readplus_hole switch (holeres4 resop) {
///         case HOLE_INFO:
///                 ressdplus_hole_info     rph_info;
///         case HOLE_NOINFO:
///                 void;
/// };
///
/// enum readplusrestype4 {
///         READ_OK = 0,
///         READ_HOLE = 1
/// };
///
/// union readplus_data switch (readplusrestype4 resop) {
///         case READ_OK:
///                 opaque          rpd_data<>;
///         case READ_HOLE:
///                 readplus_hole   rpd_hole4;
/// };
///
/// struct READPLUS4resok {
///         bool            rpr_eof;
///         readplus_data   rpr_data;
/// };
///
/// union READPLUS4res switch (nfsstat4 status) {
```

```
///         case NFS4_OK:
///                 READPLUS4resok  resok4;
///         default:
///                 void;
/// };
/// /*
///  * 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, /* Mandatory not-to-implement */
///  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, /* Mandatory not-to-implement */
///  OP_RESTOREFH           = 31,
///  OP_SAVEFH              = 32,
///  OP_SECINFO             = 33,
///  OP_SETATTR             = 34,
///  OP_SETCLIENTID         = 35, /* Mandatory not-to-implement */
///  OP_SETCLIENTID_CONFIRM = 36, /* Mandatory not-to-implement */
///  OP_VERIFY              = 37,
///  OP_WRITE               = 38,
///  OP_RELEASE_LOCKOWNER   = 39, /* Mandatory not-to-implement */
/// %
```

```
/// %/* new operations for NFSv4.1 */
/// %
///  OP_BACKCHANNEL_CTL     = 40,
///  OP_BIND_CONN_TO_SESSION = 41,
///  OP_EXCHANGE_ID         = 42,
///  OP_CREATE_SESSION      = 43,
///  OP_DESTROY_SESSION     = 44,
///  OP_FREE_STATEID        = 45,
///  OP_GET_DIR_DELEGATION  = 46,
///  OP_GETDEVICEINFO       = 47,
///  OP_GETDEVICELIST       = 48,
///  OP_LAYOUTCOMMIT        = 49,
///  OP_LAYOUTGET           = 50,
///  OP_LAYOUTRETURN        = 51,
///  OP_SECINFO_NO_NAME     = 52,
///  OP_SEQUENCE            = 53,
///  OP_SET_SSV             = 54,
///  OP_TEST_STATEID        = 55,
///  OP_WANT_DELEGATION     = 56,
///  OP_DESTROY_CLIENTID    = 57,
///  OP_RECLAIM_COMPLETE    = 58,
/// %
/// %/* new operations for NFSv4.2 */
/// %
///  OP_COPY                = 59,
///  OP_COPY_ABORT          = 60,
///  OP_COPY_NOTIFY         = 61,
///  OP_COPY_REVOKE         = 62,
///  OP_COPY_STATUS         = 63,
///  OP_HOLE_PUNCH          = 64,
///  OP_READPLUS            = 65,
///  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;
///
///  /* Not for NFSv4.1 */
///  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;
///
///  /* Not for NFSv4.1 */
///  case OP_RENEW:         RENEW4args oprenew;
///
///  case OP_RESTOREFH:     void;
///  case OP_SAVEFH:        void;
///  case OP_SECINFO:       SECINFO4args opsecinfo;
///  case OP_SETATTR:       SETATTR4args opsetattr;
///
///  /* Not for NFSv4.1 */
///  case OP_SETCLIENTID: SETCLIENTID4args opsetclientid;
///
///  /* Not for NFSv4.1 */
///  case OP_SETCLIENTID_CONFIRM: SETCLIENTID_CONFIRM4args
///                                 opsetclientid_confirm;
///  case OP_VERIFY:        VERIFY4args opverify;
///  case OP_WRITE:         WRITE4args opwrite;
///
///  /* Not for NFSv4.1 */
///  case OP_RELEASE_LOCKOWNER:
///                         RELEASE_LOCKOWNER4args
///                         oprelease_lockowner;
///
///  /* Operations new to NFSv4.1 */
///  case OP_BACKCHANNEL_CTL:
///                         BACKCHANNEL_CTL4args opbackchannel_ctl;
///
///  case OP_BIND_CONN_TO_SESSION:
///                         BIND_CONN_TO_SESSION4args
```

```
///                         opbind_conn_to_session;
///
///  case OP_EXCHANGE_ID:   EXCHANGE_ID4args opexchange_id;
///
///  case OP_CREATE_SESSION:
///                         CREATE_SESSION4args opcreate_session;
///
///  case OP_DESTROY_SESSION:
///                         DESTROY_SESSION4args opdestroy_session;
///
///  case OP_FREE_STATEID:  FREE_STATEID4args opfree_stateid;
///
///  case OP_GET_DIR_DELEGATION:
///                         GET_DIR_DELEGATION4args
///                                 opget_dir_delegation;
///
///  case OP_GETDEVICEINFO: GETDEVICEINFO4args opgetdeviceinfo;
///  case OP_GETDEVICELIST: GETDEVICELIST4args opgetdevicelist;
///  case OP_LAYOUTCOMMIT:  LAYOUTCOMMIT4args oplayoutcommit;
///  case OP_LAYOUTGET:     LAYOUTGET4args oplayoutget;
///  case OP_LAYOUTRETURN:  LAYOUTRETURN4args oplayoutreturn;
///
///  case OP_SECINFO_NO_NAME:
///                         SECINFO_NO_NAME4args opsecinfo_no_name;
///
///  case OP_SEQUENCE:      SEQUENCE4args opsequence;
///  case OP_SET_SSV:       SET_SSV4args opset_ssv;
///  case OP_TEST_STATEID:  TEST_STATEID4args optest_stateid;
///
///  case OP_WANT_DELEGATION:
///                         WANT_DELEGATION4args opwant_delegation;
///
///  case OP_DESTROY_CLIENTID:
///                         DESTROY_CLIENTID4args
///                                 opdestroy_clientid;
///
///  case OP_RECLAIM_COMPLETE:
///                         RECLAIM_COMPLETE4args
///                                 opreclaim_complete;
///
///  /* Operations new to NFSv4.2 */
///  case OP_COPY_NOTIFY:   COPY_NOTIFY4args opcopy_notify;
///  case OP_COPY_REVOKE:   COPY_REVOKE4args opcopy_revoke;
///  case OP_COPY:          COPY4args opcopy;
///  case OP_COPY_ABORT:    COPY_ABORT4args opcopy_abort;
///  case OP_COPY_STATUS:   COPY_STATUS4args opcopy_status;
///  case OP_HOLE_PUNCH:    HOLE_PUNCH4args ophole_punch;
///  case OP_READPLUS:      READPLUS4args opreadplus;
```

```
///
///  /* Operations not new to NFSv4.1 */
///  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;
///  /* Not for NFSv4.1 */
///  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;
///  /* Not for NFSv4.1 */
///  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;
///  /* Not for NFSv4.1 */
///  case OP_SETCLIENTID: SETCLIENTID4res opsetclientid;
///
///  /* Not for NFSv4.1 */
```

```
///  case OP_SETCLIENTID_CONFIRM:
///                         SETCLIENTID_CONFIRM4res
///                                 opsetclientid_confirm;
///  case OP_VERIFY:        VERIFY4res opverify;
///  case OP_WRITE:         WRITE4res opwrite;
///
///  /* Not for NFSv4.1 */
///  case OP_RELEASE_LOCKOWNER:
///                         RELEASE_LOCKOWNER4res
///                                 oprelease_lockowner;
///
///  /* Operations new to NFSv4.1 */
///  case OP_BACKCHANNEL_CTL:
///                         BACKCHANNEL_CTL4res
///                                 opbackchannel_ctl;
///
///  case OP_BIND_CONN_TO_SESSION:
///                         BIND_CONN_TO_SESSION4res
///                                  opbind_conn_to_session;
///
///  case OP_EXCHANGE_ID:   EXCHANGE_ID4res opexchange_id;
///
///  case OP_CREATE_SESSION:
///                         CREATE_SESSION4res
///                                 opcreate_session;
///
///  case OP_DESTROY_SESSION:
///                         DESTROY_SESSION4res
///                                 opdestroy_session;
///
///  case OP_FREE_STATEID:  FREE_STATEID4res
///                                 opfree_stateid;
///
///  case OP_GET_DIR_DELEGATION:
///                         GET_DIR_DELEGATION4res
///                                 opget_dir_delegation;
///
///  case OP_GETDEVICEINFO: GETDEVICEINFO4res
///                                 opgetdeviceinfo;
///
///  case OP_GETDEVICELIST: GETDEVICELIST4res
///                                 opgetdevicelist;
///
///  case OP_LAYOUTCOMMIT:  LAYOUTCOMMIT4res oplayoutcommit;
///  case OP_LAYOUTGET:     LAYOUTGET4res oplayoutget;
///  case OP_LAYOUTRETURN:  LAYOUTRETURN4res oplayoutreturn;
///
///  case OP_SECINFO_NO_NAME:
```

```
///                         SECINFO_NO_NAME4res
///                                 opsecinfo_no_name;
///
///  case OP_SEQUENCE:      SEQUENCE4res opsequence;
///  case OP_SET_SSV:       SET_SSV4res opset_ssv;
///  case OP_TEST_STATEID:  TEST_STATEID4res optest_stateid;
///
///  case OP_WANT_DELEGATION:
///                         WANT_DELEGATION4res
///                                 opwant_delegation;
///
///  case OP_DESTROY_CLIENTID:
///                         DESTROY_CLIENTID4res
///                                 opdestroy_clientid;
///
///  case OP_RECLAIM_COMPLETE:
///                         RECLAIM_COMPLETE4res
///                                 opreclaim_complete;
///
///  /* Operations new to NFSv4.1 */
///  case OP_COPY_NOTIFY:   COPY_NOTIFY4res opcopy_notify;
///  case OP_COPY_REVOKE:   COPY_REVOKE4res opcopy_revoke;
///  case OP_COPY:          COPY4res opcopy;
///  case OP_COPY_ABORT:    COPY_ABORT4res opcopy_abort;
///  case OP_COPY_STATUS:   COPY_STATUS4res opcopy_status;
///  case OP_HOLE_PUNCH:    HOLE_PUNCH4res ophole_punch;
///  case OP_READPLUS:      READPLUS4res opreadplus;
///
///  /* Operations not new to NFSv4.1 */
///  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;
/// };
///
/// /*
```

```
///  * NFSv4.1 callback arguments and results
///  */
///
/// enum layoutrecall_type4 {
///         LAYOUTRECALL4_FILE = LAYOUT4_RET_REC_FILE,
///         LAYOUTRECALL4_FSID = LAYOUT4_RET_REC_FSID,
///         LAYOUTRECALL4_ALL  = LAYOUT4_RET_REC_ALL
/// };
///
/// struct layoutrecall_file4 {
///         nfs_fh4         lor_fh;
///         offset4         lor_offset;
///         length4         lor_length;
///         stateid4        lor_stateid;
/// };
///
/// union layoutrecall4 switch(layoutrecall_type4 lor_recalltype) {
/// case LAYOUTRECALL4_FILE:
///         layoutrecall_file4 lor_layout;
/// case LAYOUTRECALL4_FSID:
///         fsid4              lor_fsid;
/// case LAYOUTRECALL4_ALL:
///         void;
/// };
///
/// struct CB_LAYOUTRECALL4args {
///         layouttype4             clora_type;
///         layoutiomode4           clora_iomode;
///         bool                    clora_changed;
///         layoutrecall4           clora_recall;
/// };
/// struct CB_LAYOUTRECALL4res {
///         nfsstat4        clorr_status;
/// };
///
/// /*
///  * Directory notification types.
///  */
/// enum notify_type4 {
///         NOTIFY4_CHANGE_CHILD_ATTRS = 0,
///         NOTIFY4_CHANGE_DIR_ATTRS = 1,
///         NOTIFY4_REMOVE_ENTRY = 2,
///         NOTIFY4_ADD_ENTRY = 3,
///         NOTIFY4_RENAME_ENTRY = 4,
///         NOTIFY4_CHANGE_COOKIE_VERIFIER = 5
/// };
///
/// /* Changed entry information.  */
```

```
/// struct notify_entry4 {
///         component4      ne_file;
///         fattr4          ne_attrs;
/// };
///
/// /* Previous entry information */
/// struct prev_entry4 {
///         notify_entry4   pe_prev_entry;
///         /* what READDIR returned for this entry */
///         nfs_cookie4     pe_prev_entry_cookie;
/// };
///
/// struct notify_remove4 {
///         notify_entry4   nrm_old_entry;
///         nfs_cookie4     nrm_old_entry_cookie;
/// };
///
/// struct notify_add4 {
///         /*
///          * Information on object
///          * possibly renamed over.
///          */
///         notify_remove4      nad_old_entry<1>;
///         notify_entry4       nad_new_entry;
///         /* what READDIR would have returned for this entry */
///         nfs_cookie4         nad_new_entry_cookie<1>;
///         prev_entry4         nad_prev_entry<1>;
///         bool                nad_last_entry;
/// };
///
/// struct notify_attr4 {
///         notify_entry4   na_changed_entry;
/// };
///
/// struct notify_rename4 {
///         notify_remove4  nrn_old_entry;
///         notify_add4     nrn_new_entry;
/// };
///
/// struct notify_verifier4 {
///         verifier4       nv_old_cookieverf;
///         verifier4       nv_new_cookieverf;
/// };
///
/// /*
///  * Objects of type notify_<>4 and
///  * notify_device_<>4 are encoded in this.
///  */
```

```
/// typedef opaque notifylist4<>;
///
/// struct notify4 {
///         /* composed from notify_type4 or notify_deviceid_type4 */
///         bitmap4         notify_mask;
///         notifylist4     notify_vals;
/// };
///
/// struct CB_NOTIFY4args {
///         stateid4    cna_stateid;
///         nfs_fh4     cna_fh;
///         notify4     cna_changes<>;
/// };
///
/// struct CB_NOTIFY4res {
///         nfsstat4    cnr_status;
/// };
///
/// struct CB_PUSH_DELEG4args {
///         nfs_fh4          cpda_fh;
///         open_delegation4 cpda_delegation;
///
/// };
///
/// struct CB_PUSH_DELEG4res {
///         nfsstat4 cpdr_status;
/// };
///
/// const RCA4_TYPE_MASK_RDATA_DLG          = 0;
/// const RCA4_TYPE_MASK_WDATA_DLG          = 1;
/// const RCA4_TYPE_MASK_DIR_DLG            = 2;
/// const RCA4_TYPE_MASK_FILE_LAYOUT        = 3;
/// const RCA4_TYPE_MASK_BLK_LAYOUT         = 4;
/// const RCA4_TYPE_MASK_OBJ_LAYOUT_MIN     = 8;
/// const RCA4_TYPE_MASK_OBJ_LAYOUT_MAX     = 9;
/// const RCA4_TYPE_MASK_OTHER_LAYOUT_MIN   = 12;
/// const RCA4_TYPE_MASK_OTHER_LAYOUT_MAX   = 15;
///
/// struct  CB_RECALL_ANY4args      {
///         uint32_t        craa_objects_to_keep;
///         bitmap4         craa_type_mask;
/// };
///
/// struct CB_RECALL_ANY4res {
///         nfsstat4        crar_status;
/// };
///
/// typedef CB_RECALL_ANY4args CB_RECALLABLE_OBJ_AVAIL4args;
```

```
///
/// struct CB_RECALLABLE_OBJ_AVAIL4res {
///         nfsstat4        croa_status;
/// };
///
/// struct CB_RECALL_SLOT4args {
///         slotid4       rsa_target_highest_slotid;
/// };
///
/// struct CB_RECALL_SLOT4res {
///         nfsstat4   rsr_status;
/// };
///
/// struct referring_call4 {
///         sequenceid4     rc_sequenceid;
///         slotid4         rc_slotid;
/// };
///
/// struct referring_call_list4 {
///         sessionid4      rcl_sessionid;
///         referring_call4 rcl_referring_calls<>;
/// };
///
/// struct CB_SEQUENCE4args {
///         sessionid4           csa_sessionid;
///         sequenceid4          csa_sequenceid;
///         slotid4              csa_slotid;
///         slotid4              csa_highest_slotid;
///         bool                 csa_cachethis;
///         referring_call_list4 csa_referring_call_lists<>;
/// };
///
/// struct CB_SEQUENCE4resok {
///         sessionid4         csr_sessionid;
///         sequenceid4        csr_sequenceid;
///         slotid4            csr_slotid;
///         slotid4            csr_highest_slotid;
///         slotid4            csr_target_highest_slotid;
/// };
///
/// union CB_SEQUENCE4res switch (nfsstat4 csr_status) {
/// case NFS4_OK:
///         CB_SEQUENCE4resok   csr_resok4;
/// default:
///         void;
/// };
///
/// struct CB_WANTS_CANCELLED4args {
```

```
///         bool cwca_contended_wants_cancelled;
///         bool cwca_resourced_wants_cancelled;
/// };
///
/// struct CB_WANTS_CANCELLED4res {
///         nfsstat4        cwcr_status;
/// };
///
/// struct CB_NOTIFY_LOCK4args {
///     nfs_fh4     cnla_fh;
///     lock_owner4 cnla_lock_owner;
/// };
///
/// struct CB_NOTIFY_LOCK4res {
///         nfsstat4        cnlr_status;
/// };
///
/// /*
///  * Device notification types.
///  */
/// enum notify_deviceid_type4 {
///         NOTIFY_DEVICEID4_CHANGE = 1,
///         NOTIFY_DEVICEID4_DELETE = 2
/// };
///
/// /* For NOTIFY4_DEVICEID4_DELETE */
/// struct notify_deviceid_delete4 {
///         layouttype4     ndd_layouttype;
///         deviceid4       ndd_deviceid;
/// };
///
/// /* For NOTIFY4_DEVICEID4_CHANGE */
/// struct notify_deviceid_change4 {
///         layouttype4     ndc_layouttype;
///         deviceid4       ndc_deviceid;
///         bool            ndc_immediate;
/// };
///
/// struct CB_NOTIFY_DEVICEID4args {
///         notify4 cnda_changes<>;
/// };
///
/// struct CB_NOTIFY_DEVICEID4res {
///         nfsstat4        cndr_status;
/// };
///
/// union copy_info4 switch (nfsstat4 cca_status) {
///         case NFS4_OK:
```

```
///                 void;
///         default:
///                 length4         cca_bytes_copied;
/// };
///
/// struct CB_COPY4args {
///         nfs_fh4         cca_fh;
///         stateid4        cca_stateid;
///         copy_info4      cca_copy_info;
/// };
/// struct CB_COPY4res {
///         nfsstat4        ccr_status;
/// };
/// /*
///  * Various definitions for CB_COMPOUND
///  */
/// %
/// enum nfs_cb_opnum4 {
///         OP_CB_GETATTR                   = 3,
///         OP_CB_RECALL                    = 4,
/// %/* Callback operations new to NFSv4.1 */
///         OP_CB_LAYOUTRECALL              = 5,
///         OP_CB_NOTIFY                    = 6,
///         OP_CB_PUSH_DELEG                = 7,
///         OP_CB_RECALL_ANY                = 8,
///         OP_CB_RECALLABLE_OBJ_AVAIL      = 9,
///         OP_CB_RECALL_SLOT               = 10,
///         OP_CB_SEQUENCE                  = 11,
///         OP_CB_WANTS_CANCELLED           = 12,
///         OP_CB_NOTIFY_LOCK               = 13,
///         OP_CB_NOTIFY_DEVICEID           = 14,
/// %/* Callback operations new to NFSv4.2 */
///         OP_CB_COPY                      = 15,
///
///         OP_CB_ILLEGAL                   = 10044
/// };
///
/// union nfs_cb_argop4 switch (unsigned argop) {
///  case OP_CB_GETATTR:
///       CB_GETATTR4args           opcbgetattr;
///
///  /* new NFSv4.1 operations */
///  case OP_CB_RECALL:
///       CB_RECALL4args            opcbrecall;
///  case OP_CB_LAYOUTRECALL:
///       CB_LAYOUTRECALL4args      opcblayoutrecall;
///  case OP_CB_NOTIFY:
///       CB_NOTIFY4args            opcbnotify;
```

```
///  case OP_CB_PUSH_DELEG:
///       CB_PUSH_DELEG4args        opcbpush_deleg;
///  case OP_CB_RECALL_ANY:
///       CB_RECALL_ANY4args        opcbrecall_any;
///  case OP_CB_RECALLABLE_OBJ_AVAIL:
///       CB_RECALLABLE_OBJ_AVAIL4args opcbrecallable_obj_avail;
///  case OP_CB_RECALL_SLOT:
///       CB_RECALL_SLOT4args       opcbrecall_slot;
///  case OP_CB_SEQUENCE:
///       CB_SEQUENCE4args          opcbsequence;
///  case OP_CB_WANTS_CANCELLED:
///       CB_WANTS_CANCELLED4args   opcbwants_cancelled;
///  case OP_CB_NOTIFY_LOCK:
///       CB_NOTIFY_LOCK4args       opcbnotify_lock;
///  case OP_CB_NOTIFY_DEVICEID:
///       CB_NOTIFY_DEVICEID4args   opcbnotify_deviceid;
///
///  /* new NFSv4.2 operations */
///  case OP_CB_COPY:
///       CB_COPY4args              opcbcopy;
///
///  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;
///
///  /* new NFSv4.1 operations */
///  case OP_CB_LAYOUTRECALL:
///                         CB_LAYOUTRECALL4res
///                                         opcblayoutrecall;
///
///  case OP_CB_NOTIFY:     CB_NOTIFY4res   opcbnotify;
///
///  case OP_CB_PUSH_DELEG: CB_PUSH_DELEG4res
///                                         opcbpush_deleg;
///
///  case OP_CB_RECALL_ANY: CB_RECALL_ANY4res
///                                         opcbrecall_any;
///
///  case OP_CB_RECALLABLE_OBJ_AVAIL:
///                         CB_RECALLABLE_OBJ_AVAIL4res
///                                 opcbrecallable_obj_avail;
///
///  case OP_CB_RECALL_SLOT:
///                         CB_RECALL_SLOT4res
///                                         opcbrecall_slot;
```

```
///
///  case OP_CB_SEQUENCE:   CB_SEQUENCE4res opcbsequence;
///
///  case OP_CB_WANTS_CANCELLED:
///                         CB_WANTS_CANCELLED4res
///                                 opcbwants_cancelled;
///
///  case OP_CB_NOTIFY_LOCK:
///                         CB_NOTIFY_LOCK4res
///                                        opcbnotify_lock;
///
///  case OP_CB_NOTIFY_DEVICEID:
///                         CB_NOTIFY_DEVICEID4res
///                                        opcbnotify_deviceid;
///
///  /* new NFSv4.2 operations */
///  case OP_CB_COPY:       CB_COPY4res     opcbcopy;
///
///  /* Not new operation */
///  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;
```

2. Security Considerations

See the Security Considerations section of [3].

3. IANA Considerations

See the IANA Considerations section of [3].

4. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

[2] Eisler, M., "XDR: External Data Representation Standard", STD 67, RFC 4506, May 2006.

[3] Haynes, T., "NFS Version 4 Minor Version 2", draft-haynes-nfsv4-minorversion2-00 (Work In Progress), March 2011.

Author's Address

Thomas Haynes
NetApp
9110 E 66th St
Tulsa, OK 74133
USA

Phone: +1-918-307-1415
Email: thomas@netapp.com