```
**********************************************************
   19771 Sat Mar 22 14:25:25 2014
new/usr/src/man/man1m/share_nfs.1m
4398 Extra spaces in man pages
Reviewed by: Marcel Telka <marcel@telka.sk>
**********************************************************
   1 '\" te
   2 .\" Copyright (C) 2008, Sun Microsystems, Inc. All Rights Reserved
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   6 .TH SHARE_NFS 1M "Mar 17, 2014"
   6 .TH SHARE_NFS 1M "May 6, 2009"
   7 .SH NAME
   8 share_nfs \- make local NFS file systems available for mounting by remote
   9 systems
  10 .SH SYNOPSIS
  11 .LP
  12 .nf
  13 \fBshare\fR [\fB-d\fR \fIdescription\fR] [\fB-F\fR nfs] [\fB-o\fR \fIspecific_op
  14 .fi

  16 .SH DESCRIPTION
  17 .sp
  18 .LP
  19 The \fBshare\fR utility makes local file systems available for mounting by
  20 remote systems. It starts the \fBnfsd\fR(1M) and \fBmountd\fR(1M) daemons if
  21 they are not already running.
  22 .sp
  23 .LP
  24 If no argument is specified, then \fBshare\fR displays all file systems
  25 currently shared, including \fBNFS\fR file systems and file systems shared
  26 through other distributed file system packages.
  27 .SH OPTIONS
  28 .sp
  29 .LP
  30 The following options are supported:
  31 .sp
  32 .ne 2
  33 .na
  34 \fB\fB-d\fR \fIdescription\fR\fR
  35 .ad
  36 .sp .6
  37 .RS 4n
  38 Provide a comment that describes the file system to be shared.
  39 .RE

  41 .sp
  42 .ne 2
  43 .na
  44 \fB\fB\fR\fB-F\fR \fBnfs\fR\fR
  45 .ad
  46 .sp .6
  47 .RS 4n
  48 Share \fBNFS\fR file system type.
  49 .RE

  51 .sp
  52 .ne 2
  53 .na
  54 \fB\fB-o\fR \fIspecific_options\fR\fR
  55 .ad
  56 .sp .6
  57 .RS 4n
  58 Specify \fIspecific_options\fR in a comma-separated list of keywords and
  59 attribute-value-assertions for interpretation by the file-system-type-specific
```



```
  60 command. If \fIspecific_options\fR is not specified, then by default sharing is
  61 read-write to all clients. \fIspecific_options\fR can be any combination of the
  62 following:
  63 .sp
  64 .ne 2
  65 .na
  66 \fB\fBaclok\fR\fR
  67 .ad
  68 .sp .6
  69 .RS 4n
  70 Allows the \fBNFS\fR server to do access control for \fBNFS\fR Version 2
  71 clients (running SunOS 2.4 or earlier). When \fBaclok\fR is set on the server,
  72 maximal access is given to all clients. For example, with \fBaclok\fR set, if
  73 anyone has read permissions, then everyone does. If \fBaclok\fR is not set,
  74 minimal access is given to all clients.
  75 .RE

  77 .sp
  78 .ne 2
  79 .na
  80 \fB\fBanon=\fR\fIuid\fR\fR
  81 .ad
  82 .sp .6
  83 .RS 4n
  84 Set \fIuid\fR to be the effective user \fBID\fR of unknown users. By default,
  85 unknown users are given the effective user \fBID\fR \fBUID_NOBODY\fR. If
  86 \fIuid\fR is set to \fB\(mi1\fR, access is denied.
  87 .RE

  89 .sp
  90 .ne 2
  91 .na
  92 \fB\fIcharset\fR=\fIaccess_list\fR\fR
  93 .ad
  94 .sp .6
  95 .RS 4n
  96 Where \fIcharset\fR is one of: \fBeuc-cn\fR, \fBeuc-jp\fR, \fBeuc-jpms\fR,
  97 \fBeuc-kr\fR, \fBeuc-tw\fR, \fBiso8859-1\fR, \fBiso8859-2\fR, \fBiso8859-5\fR,
  98 \fBiso8859-6\fR, \fBiso8859-7\fR, \fBiso8859-8\fR, \fBiso8859-9\fR,
  99 \fBiso8859-13\fR, \fBiso8859-15\fR, \fBkoi8-r\fR.
 100 .sp
 101 Clients that match the \fIaccess_list\fR for one of these properties will be
 102 assumed to be using that character set and file and path names will be
 103 converted to UTF-8 for the server.
 104 .RE

 106 .sp
 107 .ne 2
 108 .na
 109 \fB\fBindex=\fR\fBfile\fR\fR
 110 .ad
 111 .sp .6
 112 .RS 4n
 113 Load \fBfile\fR rather than a listing of the directory containing this file
 114 when the directory is referenced by an \fBNFS URL\fR.
 115 .RE

 117 .sp
 118 .ne 2
 119 .na
 120 \fB\fBlog=tag\fR\fR
 121 .ad
 122 .sp .6
 123 .RS 4n
 124 Enables \fBNFS\fR server logging for the specified file system. The optional
 125 tag determines the location of the related log files. The \fBtag\fR is defined
```

in \fBetc/nfs/nfslog.conf\fR. If no \fBtag\fR is specified, the default values associated with the \fBglobal\fR \fBtag\fR in \fBetc/nfs/nfslog.conf\fR is used. Support of NFS server logging is only available for NFS Version 2 and Version 3 requests.
.RE

.sp
.ne 2
.na
\fB\fBnone=\fR\fIaccess_list\fR\fR
.ad
.sp .6
.RS 4n
Access is not allowed to any client that matches the access list. The exception is when the access list is an asterisk (\fB*\fR), in which case \fBro\fR or \fBrw\fR can override \fBnone\fR.
.RE

.sp
.ne 2
.na
\fB\fBnosub\fR\fR
.ad
.sp .6
.RS 4n
Prevents clients from mounting subdirectories of shared directories. For example, if \fB/export\fR is shared with the \fBnosub\fR option on server \fIfooey\fR then a \fBNFS\fR client cannot do:
.sp
.in +2
.nf
mount -F nfs fooey:/export/home/mnt
.fi
.in -2
.sp

NFS Version 4 does not use the \fBMOUNT\fR protocol. The \fBnosub\fR option only applies to NFS Version 2 and Version 3 requests.
.RE

.sp
.ne 2
.na
\fB\fBnosuid\fR\fR
.ad
.sp .6
.RS 4n
By default, clients are allowed to create files on the shared file system with the setuid or setgid mode enabled. Specifying \fBnosuid\fR causes the server file system to silently ignore any attempt to enable the setuid or setgid mode bits.
.RE

.sp
.ne 2
.na
\fB\fBpublic\fR\fR
.ad
.sp .6
.RS 4n
Moves the location of the public file handle from \fBroot\fR (\fB/\fR) to the exported directory for Web\fBNFS\fR-enabled browsers and clients. This option does not enable Web\fBNFS\fR service; Web\fBNFS\fR is always on. Only one file system per server may use this option. Any other option, including the \fB-ro=list\fR and \fB-rw=list\fR options can be included with the \fBpublic\fR option.

.RE

.sp
.ne 2
.na
\fB\fBro\fR\fR
.ad
.sp .6
.RS 4n
Sharing is read-only to all clients.
.RE

.sp
.ne 2
.na
\fB\fBro=\fR\fIaccess_list\fR\fR
.ad
.sp .6
.RS 4n
Sharing is read-only to the clients listed in \fIaccess_list\fR; overrides the \fBrw\fR suboption for the clients specified. See \fIaccess_list\fR below.
.RE

.sp
.ne 2
.na
\fB\fBroot=\fR\fIaccess_list\fR\fR
.ad
.sp .6
.RS 4n
Only root users from the hosts specified in \fIaccess_list\fR have root access. See \fIaccess_list\fR below. By default, no host has root access, so root users are mapped to an anonymous user \fBID\fR (see the \fBanon=\fR\fIuid\fR option described above). Netgroups can be used if the file system shared is using UNIX
authentication (\fBAUTH_SYS\fR).
authentication (\fBAUTH_SYS\fR).
.RE

.sp
.ne 2
.na
\fB\fBroot_mapping=\fIuid\fR\fR\fR
.ad
.sp .6
.RS 4n
For a client that is allowed root access, map the root UID to the specified user id.
.RE

.sp
.ne 2
.na
\fB\fBrw\fR\fR
.ad
.sp .6
.RS 4n
Sharing is read-write to all clients.
.RE

.sp
.ne 2
.na
\fB\fBrw=\fR\fIaccess_list\fR\fR
.ad
.sp .6
.RS 4n

```
257 Sharing is read-write to the clients listed in \fIaccess_list\fR; overrides the
258 \fBro\fR suboption for the clients specified. See \fIaccess_list\fR below.
259 .RE

261 .sp
262 .ne 2
263 .na
264 \fB\fBsec=\fR\fImode\fR[\fB:\fR\fImode\fR].\|.\|.\fR
265 .ad
266 .sp .6
267 .RS 4n
268 Sharing uses one or more of the specified security modes. The \fImode\fR in the
269 \fBsec=\fR\fImode\fR option must be a node name supported on the client. If the
270 \fBsec=\fR option is not specified, the default security mode used is
271 \fBAUTH_SYS.\fR Multiple \fBsec=\fR options can be specified on the command
272 line, although each mode can appear only once. The security modes are defined
273 in \fBnfssec\fR(5).
274 .sp
275 Each \fBsec=\fR option specifies modes that apply to any subsequent \fBwindow=,
276 rw, ro, rw=, ro=\fR and \fBroot=\fR options that are provided before another
277 \fBsec=\fRoption. Each additional \fBsec=\fR resets the security mode context,
278 so that more \fBwindow=,\fR \fBrw,\fR \fBro,\fR \fBrw=,\fR \fBro=\fR and
279 \fBroot=\fR options can be supplied for additional modes.
280 .RE

282 .sp
283 .ne 2
284 .na
285 \fB\fBsec=\fR\fInone\fR\fR
286 .ad
287 .sp .6
288 .RS 4n
289 If the option \fBsec=\fR\fInone\fR is specified when the client uses
290 \fBAUTH_NONE,\fR or if the client uses a security mode that is not one that the
291 file system is shared with, then the credential of each \fBNFS\fR request is
292 treated as unauthenticated. See the \fBanon=\fR\fIuid\fR option for a
293 description of how unauthenticated requests are handled.
294 .RE

296 .sp
297 .ne 2
298 .na
299 \fB\fBsecure\fR\fR
300 .ad
301 .sp .6
302 .RS 4n
303 This option has been deprecated in favor of the \fBsec=\fR\fIdh\fR option.
304 .RE

306 .sp
307 .ne 2
308 .na
309 \fB\fBwindow=\fR\fIvalue\fR\fR
310 .ad
311 .sp .6
312 .RS 4n
313 When sharing with \fBsec=\fR\fIdh\fR, set the maximum life time (in seconds) of
314 the \fBRPC\fR request's credential (in the authentication header) that the
315 \fBNFS\fR server allows. If a credential arrives with a life time larger than
316 what is allowed, the \fBNFS\fR server rejects the request. The default value is
317 30000 seconds (8.3 hours).
318 .RE

320 .RE

322 .SS "\fIaccess_list\fR"
```



```
323 .sp
324 .LP
325 The \fIaccess_list\fR argument is a colon-separated list whose components may
326 be any number of the following:
327 .sp
328 .ne 2
329 .na
330 \fBhostname\fR
331 .ad
332 .sp .6
333 .RS 4n
334 The name of a host. With a server configured for \fBDNS\fR or \fBLDAP\fR naming
335 in the \fBnsswitch\fR "hosts" entry, any hostname must be represented as a
336 fully qualified \fBDNS\fR or \fBLDAP\fR name.
337 .RE

339 .sp
340 .ne 2
341 .na
342 \fBnetgroup\fR
343 .ad
344 .sp .6
345 .RS 4n
346 A netgroup contains a number of hostnames. With a server configured for
347 \fBDNS\fR or \fBLDAP\fR naming in the \fBnsswitch\fR "hosts" entry, any
348 hostname in a netgroup must be represented as a fully qualified \fBDNS\fR or
349 \fBLDAP\fR name.
350 .RE

352 .sp
353 .ne 2
354 .na
355 \fBdomain name suffix\fR
356 .ad
357 .sp .6
358 .RS 4n
359 To use domain membership the server must use \fBDNS\fR or \fBLDAP\fR to resolve
360 hostnames to \fBIP\fR addresses; that is, the "hosts" entry in the
361 \fB/etc/nsswitch.conf\fR must specify "dns" or "ldap" ahead of "nis" or
362 "nisplus", since only \fBDNS\fR and \fBLDAP\fR return the full domain name of
363 the host. Other name services like \fBNIS\fR or \fBNIS+\fR cannot be used to
364 resolve hostnames on the server because when mapping an \fBIP\fR address to a
365 hostname they do not return domain information. For example,
366 .sp
367 .in +2
368 .nf
369 NIS or NIS+   172.16.45.9 --> "myhost"
370 .fi
371 .in -2
372 .sp

374 and
375 .sp
376 .in +2
377 .nf
378 DNS or LDAP   172.16.45.9 -->
379      "myhost.mydomain.mycompany.com"
380 .fi
381 .in -2
382 .sp

384 The domain name suffix is distinguished from hostnames and netgroups by a
385 prefixed dot. For example,
386 .sp
387 \fBrw=.mydomain.mycompany.com\fR
388 .sp
```

```
A single dot can be used to match a hostname with no suffix. For example,
.sp
\fBrw=.\fR
.sp
matches "mydomain" but not "mydomain.mycompany.com". This feature can be used
to match hosts resolved through \fBNIS\fR and \fBNIS+\fR rather than \fBDNS\fR
and \fBLDAP\fR.
.RE

.sp
.ne 2
.na
\fBnetwork\fR
.ad
.sp .6
.RS 4n
The network or subnet component is preceded by an at-sign (\fB@\fR). It can be
either a name or a dotted address. If a name, it is converted to a dotted
address by \fBgetnetbyname\fR(3SOCKET). For example,
.sp
\fB=@mynet\fR
.sp
would be equivalent to:
.sp
\fB=@172.16\fR or \fB=@172.16.0.0\fR
.sp
The network prefix assumes an octet-aligned netmask determined from the zeroth
octet in the low-order part of the address up to and including the high-order
octet, if you want to specify a single IP address (see below). In the case
where network prefixes are not byte-aligned, the syntax allows a mask length to
be specified explicitly following a slash (\fB/\fR) delimiter. For example,
.sp
\fB=@theothernet/17\fR or \fB=@172.16.132/22\fR
.sp
\&...where the mask is the number of leftmost contiguous significant bits in
the corresponding IP address.
.sp
When specifying individual IP addresses, use the same \fB@\fR notation
described above, without a netmask specification. For example:
.sp
.in +2
.nf
=@172.16.132.14
.fi
.in -2
.sp

Multiple, individual IP addresses would be specified, for example, as:
.sp
.in +2
.nf
root=@172.16.132.20:@172.16.134.20
.fi
.in -2
.sp

.RE

.sp
.LP
A prefixed minus sign (\fB\(mi\fR) denies access to that component of
\fIaccess_list\fR. The list is searched sequentially until a match is found
that either grants or denies access, or until the end of the list is reached.
For example, if host "terra" is in the "engineering" netgroup, then
.sp
.in +2
```



```
.nf
rw=-terra:engineering
.fi
.in -2
.sp

.sp
.LP
denies access to \fBterra\fR but
.sp
.in +2
.nf
rw=engineering:-terra
.fi
.in -2
.sp

.sp
.LP
grants access to \fBterra\fR.
.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fIpathname\fR\fR
.ad
.sp .6
.RS 4n
The pathname of the file system to be shared.
.RE

.SH EXAMPLES
.LP
\fBExample 1 \fRSharing A File System With Logging Enabled
.sp
.LP
The following example shows the \fB/export\fR file system shared with logging
enabled:

.sp
.in +2
.nf
example% \fBshare -o log /export\fR
.fi
.in -2
.sp

.sp
.LP
The default global logging parameters are used since no tag identifier is
specified. The location of the log file, as well as the necessary logging work
files, is specified by the global entry in \fB/etc/nfs/nfslog.conf\fR. The
\fBnfslogd\fR(1M) daemon runs only if at least one file system entry in
\fB/etc/dfs/dfstab\fR is shared with logging enabled upon starting or rebooting
the system. Simply sharing a file system with logging enabled from the command
line does not start the \fBnfslogd\fR(1M).

.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
.sp
.ne 2
```

```
.na
\fB\fB0\fR\fR
.ad
.sp .6
.RS 4n
Successful completion.
.RE

.sp
.ne 2
.na
\fB\fB>0\fR\fR
.ad
.sp .6
.RS 4n
An error occurred.
.RE

.SH FILES
.sp
.ne 2
.na
\fB\fB/etc/dfs/fstypes\fR\fR
.ad
.sp .6
.RS 4n
list of system types, \fBNFS\fR by default
.RE

.sp
.ne 2
.na
\fB\fB/etc/dfs/sharetab\fR\fR
.ad
.sp .6
.RS 4n
system record of shared file systems
.RE

.sp
.ne 2
.na
\fB\fB/etc/nfs/nfslogtab\fR\fR
.ad
.sp .6
.RS 4n
system record of logged file systems
.RE

.sp
.ne 2
.na
\fB\fB/etc/nfs/nfslog.conf\fR\fR
.ad
.sp .6
.RS 4n
logging configuration file
.RE

.SH SEE ALSO
.sp
.LP
\fBmount\fR(1M), \fBmountd\fR(1M), \fBnfsd\fR(1M), \fBnfslogd\fR(1M),
\fBshare\fR(1M), \fBunshare\fR(1M), \fBgetnetbyname\fR(3SOCKET),
\fBnfslog.conf\fR(4), \fBnetgroup\fR(4), \fBattributes\fR(5), \fBnfssec\fR(5)
.SH NOTES
```



```
.sp
.LP
If the \fBsec=\fR option is presented at least once, all uses of the
\fBwindow=,\fR \fBrw,\fR \fBro,\fR \fBrw=,\fR \fBro=\fR and \fBroot=\fR options
must come \fBafter\fR the first \fBsec=\fR option. If the \fBsec=\fR option is
not presented, then \fBsec=\fR\fIsys\fR is implied.
.sp
.LP
If one or more explicit \fBsec=\fR options are presented, \fIsys\fR must appear
in one of the options mode lists for accessing using the \fBAUTH_SYS\fR
security mode to be allowed. For example:
.sp
.in +2
.nf
\fBshare\fR \fB-F\fR \fBnfs /var\fR
\fBshare\fR \fB-F\fR \fBnfs\fR \fB-o\fR \fBsec=sys /var\fR
.fi
.in -2
.sp

.sp
.LP
grants read-write access to any host using \fBAUTH_SYS,\fR but
.sp
.in +2
.nf
\fBshare\fR \fB-F\fR \fBnfs\fR \fB-o\fR \fBsec=dh /var\fR
.fi
.in -2
.sp

.sp
.LP
grants no access to clients that use \fBAUTH_SYS.\fR
.sp
.LP
Unlike previous implementations of \fBshare_nfs\fR, access checking for the
\fBwindow=, rw, ro, rw=,\fR and \fBro=\fR options is done per \fBNFS\fR
request, instead of per mount request.
.sp
.LP
Combining multiple security modes can be a security hole in situations where
the \fBro=\fR and \fBrw=\fR options are used to control access to weaker
security modes. In this example,
.sp
.in +2
.nf
\fBshare\fR \fB-F\fR \fBnfs\fR \fB-o\fR \fBsec=dh,rw,sec=sys,rw=hosta /var\fR
.fi
.in -2
.sp

.sp
.LP
an intruder can forge the IP address for \fBhosta\fR (albeit on each \fBNFS\fR
request) to side-step the stronger controls of \fBAUTH_DES.\fR Something like:
.sp
.in +2
.nf
\fBshare\fR \fB-F\fR \fBnfs\fR \fB-o\fR \fBsec=dh,rw,sec=sys,ro /var\fR
.fi
.in -2
.sp

.sp
.LP
```

is safer, because any client (intruder or legitimate) that avoids \fBAUTH_DES\fR only gets read-only access. In general, multiple security modes per \fBshare\fR command should only be used in situations where the clients using more secure modes get stronger access than clients using less secure modes.
.sp
.LP
If \fBrw=,\fR and \fBro=\fR options are specified in the same \fBsec=\fR clause, and a client is in both lists, the order of the two options determines the access the client gets. If client \fBhosta\fR is in two netgroups - \fBgroup1\fR and \fBgroup2\fR - in this example, the client would get read-only access:
.sp
.in +2
.nf
\fBshare\fR \fB-F\fR \fBnfs\fR \fB-o\fR \fBro=group1,rw=group2 /var\fR
.fi
.in -2
.sp

.sp
.LP
In this example \fBhosta\fR would get read-write access:
.sp
.in +2
.nf
\fBshare\fR \fB-F\fR \fBnfs\fR \fB-o\fR \fBrw=group2,ro=group1 /var\fR
.fi
.in -2
.sp

.sp
.LP
If within a \fBsec=\fR clause, both the \fBro\fR and \fBrw=\fR options are specified, for compatibility, the order of the options rule is not enforced. All hosts would get read-only access, with the exception to those in the read-write list. Likewise, if the \fBro=\fR and \fBrw\fR options are specified, all hosts get read-write access with the exceptions of those in the read-only list.
.sp
.LP
The \fBro=\fR and \fBrw=\fR options are guaranteed to work over \fBUDP\fR and \fBTCP\fR but may not work over other transport providers.
.sp
.LP
The \fBroot=\fR option with \fBAUTH_SYS\fR is guaranteed to work over \fBUDP\fR and \fBTCP\fR but may not work over other transport providers.
.sp
.LP
The \fBroot=\fR option with \fBAUTH_DES\fR is guaranteed to work over any transport provider.
.sp
.LP
There are no interactions between the \fBroot=\fR option and the \fBrw, ro, rw=,\fR and \fBro=\fR options. Putting a host in the \fBroot\fR list does not override the semantics of the other options. The access the host gets is the same as when the \fBroot=\fR options is absent. For example, the following \fBshare\fR command denies access to \fBhostb:\fR
.sp
.in +2
.nf
\fBshare\fR \fB-F\fR \fBnfs\fR \fB-o\fR \fBro=hosta,root=hostb /var\fR
.fi
.in -2
.sp

.sp
.LP
The following gives read-only permissions to \fBhostb:\fR
.sp
.in +2
.nf
\fBshare\fR \fB-F\fR \fBnfs\fR \fB-o\fR \fBro=hostb,root=hostb /var\fR
.fi
.in -2
.sp

.sp
.LP
The following gives read-write permissions to \fBhostb:\fR
.sp
.in +2
.nf
\fBshare\fR \fB-F\fR \fBnfs\fR \fB-o\fR \fBro=hosta,rw=hostb,root=hostb /var\fR
.fi
.in -2
.sp

.sp
.LP
If the file system being shared is a symbolic link to a valid pathname, the canonical path (the path which the symbolic link follows) are shared. For example, if \fB/export/foo\fR is a symbolic link to \fB/export/bar\fR (\fB/export/foo -> /export/bar\fR), the following \fBshare\fR command results in \fB/export/bar\fR as the shared pathname (and not \fB/export/foo\fR).
.sp
.in +2
.nf
\fBexample# share\fR \fB-F\fR \fBnfs /export/foo\fR
.fi
.in -2
.sp

.sp
.LP
An \fBNFS\fR mount of \fBserver:/export/foo\fR results in \fBserver:/export/bar\fR really being mounted.
.sp
.LP
This line in the \fB/etc/dfs/dfstab\fR file shares the \fB/disk\fR file system read-only at boot time:
.sp
.in +2
.nf
\fBshare\fR \fB-F\fR \fBnfs\fR \fB-o\fR \fBro /disk\fR
.fi
.in -2
.sp

.sp
.LP
The same command entered from the command line does not share the \fB/disk\fR file system unless there is at least one file system entry in the \fB/etc/dfs/dfstab\fR file. The \fBmountd\fR(1M) and \fBnfsd\fR(1M) daemons only run if there is a file system entry in \fB/etc/dfs/dfstab\fR when starting or rebooting the system.
.sp
.LP
The \fBmountd\fR(1M) process allows the processing of a path name the contains a symbolic link. This allows the processing of paths that are not themselves explicitly shared with \fBshare_nfs\fR. For example, \fB/export/foo\fR might be a symbolic link that refers to \fB/export/bar\fR which has been specifically

shared. When the client mounts \fB/export/foo\fR the \fBmountd\fR processing follows the symbolic link and responds with the \fB/export/bar\fR. The NFS Version 4 protocol does not use the \fBmountd\fR processing and the client's use of \fB/export/foo\fR does not work as it does with NFS Version 2 and Version 3 and the client receives an error when attempting to mount \fB/export/foo\fR.

```
**********************************************************
    6810 Sat Mar 22 14:25:25 2014
new/usr/src/man/man3nsl/xdr_admin.3nsl
4398 Extra spaces in man pages
Reviewed by: Marcel Telka <marcel@telka.sk>
**********************************************************
```

```
'\" te
.\"  Copyright 1989 AT&T  Copyright (c) 1997, Sun Microsystems, Inc.  All Rights
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH XDR_ADMIN 3NSL "Mar 17, 2014"
.TH XDR_ADMIN 3NSL "Dec 30, 1996"
.SH NAME
xdr_admin, xdr_control, xdr_getpos, xdr_inline, xdrrec_endofrecord, xdrrec_eof,
xdrrec_readbytes, xdrrec_skiprecord, xdr_setpos, xdr_sizeof \- library routines
for external data representation
.SH DESCRIPTION
.sp
.LP
\fBXDR\fR library routines allow C programmers to describe arbitrary data
structures in a machine-independent fashion. Protocols such as remote procedure
calls (RPC) use these routines to describe the format of the data.
.sp
.LP
These routines deal specifically with the management of the \fBXDR\fR stream.
.SS "Routines"
.sp
.LP
See \fBrpc\fR(3NSL) for the definition of the \fBXDR\fR data structure. Note
that any buffers passed to the \fBXDR\fR routines must be properly aligned.  It
is suggested either that \fBmalloc\fR(3C) be used to allocate these buffers, or
that the programmer insure  that the buffer address is divisible evenly by
four.
.sp
.LP
\fB#include <rpc/xdr.h>\fR
.sp
.ne 2
.na
\fB\fBbool_t xdr_control(XDR *\fR\fIxdrs\fR\fB, int\fR\fI req\fR\fB, void
\fB\fBbool_t xdr_control( XDR *\fR\fIxdrs\fR\fB, int\fR\fI req\fR\fB, void
*\fR\fIinfo\fR\fB);\fR\fR
.ad
.sp .6
.RS 4n
A function macro to change or retrieve various information about an  \fBXDR\fR
stream. \fIreq\fR indicates the type of operation and \fIinfo\fR is a pointer
to the information. The supported values of \fIreq\fR is
\fBXDR_GET_BYTES_AVAIL\fR and its argument type is \fBxdr_bytesrec *\fR. They
return the number of bytes left unconsumed in the stream and a flag indicating
whether or not this is the last fragment.
.RE

.sp
.ne 2
.na
\fB\fBuint_t xdr_getpos(const XDR *\fR\fIxdrs\fR\fB);\fR\fR
.ad
.sp .6
.RS 4n
A macro that invokes the get-position routine associated with the \fBXDR\fR
stream, \fIxdrs\fR. The routine returns an unsigned integer, which indicates
the position of the \fBXDR\fR byte stream. A desirable feature of \fBXDR\fR
streams is that simple arithmetic works with this number, although the
\fBXDR\fR stream instances need not guarantee this. Therefore, applications
```



```
written for portability should not depend on this feature.
.RE

.sp
.ne 2
.na
\fB\fBlong *xdr_inline(XDR *\fR\fIxdrs\fR\fB, const int \fR\fIlen\fR\fB);\fR\fR
.ad
.sp .6
.RS 4n
A macro that invokes the in-line routine associated with the \fBXDR\fR stream,
\fIxdrs\fR. The routine returns a pointer to a contiguous piece of the stream's
buffer; \fIlen\fR is the byte length of the desired buffer. Note: pointer is
cast to \fBlong *\fR.
.sp
Warning: \fBxdr_inline()\fR may return \fINULL\fR (\fB0\fR) if it cannot
allocate a contiguous piece of a buffer. Therefore the behavior may vary among
stream instances; it exists for the sake of efficiency, and applications
written for portability should not depend on this feature.
.RE

.sp
.ne 2
.na
\fB\fBbool_t xdrrec_endofrecord(XDR *xdrs, int \fR\fIsendnow\fR\fB);\fR\fR
.ad
.sp .6
.RS 4n
This routine can be invoked only on streams created by \fBxdrrec_create()\fR.
See \fBxdr_create\fR(3NSL). The data in the output buffer is marked as a
completed record, and the output buffer is optionally written out if
\fIsendnow\fR is non-zero. This routine returns \fBTRUE\fR if it succeeds,
\fBFALSE\fR otherwise.
.RE

.sp
.ne 2
.na
\fB\fBbool_t xdrrec_eof(XDR *\fR\fIxdrs\fR\fB);\fR\fR
.ad
.sp .6
.RS 4n
This routine can be invoked only on streams created by \fBxdrrec_create()\fR.
After consuming the rest of the current record in the stream, this routine
returns \fBTRUE\fR if there is no more data in the stream's input buffer. It
returns  \fBFALSE\fR if there is additional data in the stream's input buffer.
.RE

.sp
.ne 2
.na
\fB\fBint xdrrec_readbytes(XDR *\fR\fIxdrs\fR\fB, caddr_t\fR\fI addr\fR\fB,
uint_t\fR\fI nbytes\fR\fB);\fR\fR
.ad
.sp .6
.RS 4n
This routine can be invoked only on streams created by \fBxdrrec_create()\fR.
It attempts to read \fInbytes\fR bytes from the \fBXDR\fR stream into the
buffer pointed to by \fIaddr\fR. Upon success this routine returns the number
of bytes read. Upon  failure, it returns \fB\(mi1\fR\&. A return value of
\fB0\fR indicates an end of record.
.RE

.sp
.ne 2
.na
```

```
\fB\fBbool_t xdrrec_skiprecord(XDR *\fR\fIxdrs\fR\fB);\fR\fR
.ad
.sp .6
.RS 4n
This routine can be invoked only on streams created by \fBxdrrec_create()\fR.
See \fBxdr_create\fR(3NSL). It tells the   \fBXDR\fR implementation that the
rest of the current record in the stream's input buffer should be discarded.
This routine returns \fBTRUE\fR if it succeeds,   \fBFALSE\fR otherwise.
.RE

.sp
.ne 2
.na
\fB\fBbool_t xdr_setpos(XDR *\fR\fIxdrs\fR\fB, const uint_t
\fR\fIpos\fR\fB);\fR\fR
.ad
.sp .6
.RS 4n
A macro that invokes the set position routine associated with the \fBXDR\fR
stream \fIxdrs\fR. The parameter \fIpos\fR is a position value obtained from
\fBxdr_getpos()\fR. This routine returns \fBTRUE\fR if the \fBXDR\fR stream was
repositioned, and \fBFALSE\fR otherwise.
.sp
Warning: it is difficult to reposition some types of \fBXDR\fR streams, so this
routine may fail with one type of stream and succeed with another. Therefore,
applications written for portability should not depend on this feature.
.RE

.sp
.ne 2
.na
\fB\fBunsigned long xdr_sizeof(xdrproc_t \fR\fIfunc\fR\fB, void
*\fR\fIdata\fR\fB);\fR\fR
.ad
.sp .6
.RS 4n
This routine returns the number of bytes required to encode \fIdata\fR using
the \fBXDR\fR filter function \fIfunc\fR, excluding potential overhead such as
\fBRPC\fR headers or record markers. \fB0\fR is returned on error. This
information might be used to select between transport protocols, or to
determine the buffer size for various lower levels of \fBRPC\fR client and
server creation routines, or to allocate storage when  \fBXDR\fR is used
outside of the \fBRPC\fR subsystem.
.RE

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
MT-Level        Safe
.TE

.SH SEE ALSO
.sp
.LP
\fBmalloc\fR(3C), \fBrpc\fR(3NSL), \fBxdr_complex\fR(3NSL),
\fBxdr_create\fR(3NSL), \fBxdr_simple\fR(3NSL), \fBattributes\fR(5)
```

```
**********************************************************
   19590 Sat Mar 22 14:25:26 2014
new/usr/src/man/man7p/tcp.7p
4398 Extra spaces in man pages
Reviewed by: Marcel Telka <marcel@telka.sk>
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright (c) 2006, Sun Microsystems, Inc. All Rights Reserved.
   3 .\" Copyright (c) 2011 Nexenta Systems, Inc. All rights reserved.
   4 .\" Copyright 1989 AT&T
   5 .\" The contents of this file are subject to the terms of the Common Development
   6 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   7 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   8 .TH TCP 7P "Mar 17, 2014"
   8 .TH TCP 7P "Jun 30, 2006"
   9 .SH NAME
  10 tcp, TCP \- Internet Transmission Control Protocol
  11 .SH SYNOPSIS
  12 .LP
  13 .nf
  14 \fB#include <sys/socket.h>\fR
  15 .fi

  17 .LP
  18 .nf
  19 \fB#include <netinet/in.h>\fR
  20 .fi

  22 .LP
  23 .nf
  24 \fBs = socket(AF_INET, SOCK_STREAM, 0);\fR
  25 .fi

  27 .LP
  28 .nf
  29 \fBs = socket(AF_INET6, SOCK_STREAM, 0);\fR
  30 .fi

  32 .LP
  33 .nf
  34 \fBt = t_open("/dev/tcp", O_RDWR);\fR
  35 .fi

  37 .LP
  38 .nf
  39 \fBt = t_open("/dev/tcp6", O_RDWR);\fR
  40 .fi

  42 .SH DESCRIPTION
  43 .sp
  44 .LP
  45 \fBTCP\fR is the virtual circuit protocol of the Internet protocol family. It
  46 provides reliable, flow-controlled, in order, two-way transmission of data. It
  47 is a byte-stream protocol layered above the Internet Protocol (\fBIP\fR), or
  48 the Internet Protocol Version 6 (\fBIPv6\fR), the Internet protocol family's
  49 internetwork datagram delivery protocol.
  50 .sp
  51 .LP
  52 Programs can access \fBTCP\fR using the socket interface as a \fBSOCK_STREAM\fR
  53 socket type, or using the Transport Level Interface (\fBTLI\fR) where it
  54 supports the connection-oriented (\fBT_COTS_ORD\fR) service type.
  55 .sp
  56 .LP
  57 \fBTCP\fR uses \fBIP\fR's host-level addressing and adds its own per-host
  58 collection of "port addresses." The endpoints of a \fBTCP\fR connection are
  59 identified by the combination of an \fBIP\fR or IPv6 address and a \fBTCP\fR
```



```
  60 port number. Although other protocols, such as the User Datagram Protocol
  61 (UDP), may use the same host and port address format, the port space of these
  62 protocols is distinct. See \fBinet\fR(7P) and \fBinet6\fR(7P) for details on
  63 the common aspects of addressing in the Internet protocol family.
  64 .sp
  65 .LP
  66 Sockets utilizing \fBTCP\fR are either "active" or "passive." Active sockets
  67 initiate connections to passive sockets. Both types of sockets must have their
  68 local \fBIP\fR or IPv6 address and \fBTCP\fR port number bound with the
  69 \fBbind\fR(3SOCKET) system call after the socket is created. By default,
  70 \fBTCP\fR sockets are active. A passive socket is created by calling the
  71 \fBlisten\fR(3SOCKET) system call after binding the socket with \fBbind()\fR.
  72 This establishes a queueing parameter for the passive socket. After this,
  73 connections to the passive socket can be received with the
  74 \fBaccept\fR(3SOCKET) system call. Active sockets use the
  75 \fBconnect\fR(3SOCKET) call after binding to initiate connections.
  76 .sp
  77 .LP
  78 By using the special value \fBINADDR_ANY\fR with \fBIP\fR, or the unspecified
  79 address (all zeroes) with IPv6, the local \fBIP\fR address can be left
  80 unspecified in the \fBbind()\fR call by either active or passive \fBTCP\fR
  81 sockets. This feature is usually used if the local address is either unknown or
  82 irrelevant. If left unspecified, the local \fBIP\fR or IPv6 address will be
  83 bound at connection time to the address of the network interface used to
  84 service the connection.
  85 .sp
  86 .LP
  87 Note that no two TCP sockets can be bound to the same port unless the bound IP
  88 addresses are different.  IPv4 \fBINADDR_ANY\fR and IPv6 unspecified addresses
  89 compare as equal to any IPv4 or IPv6 address. For example, if a socket is bound
  90 to \fBINADDR_ANY\fR or unspecified address and port X, no other socket can bind
  91 to port X, regardless of the binding address. This special consideration of
  92 \fBINADDR_ANY\fR and unspecified address can be changed using the socket option
  93 \fBSO_REUSEADDR\fR. If \fBSO_REUSEADDR\fR is set on a socket doing a bind, IPv4
  94 \fBINADDR_ANY\fR and IPv6 unspecified address do not compare as equal to any IP
  95 address. This means that as long as the two sockets are not both bound to
  96 \fBINADDR_ANY\fR/unspecified address or the same IP address, the two sockets
  97 can be bound to the same port.
  98 .sp
  99 .LP
 100 If an application does not want to allow another socket  using the
 100  If an application does not want to allow another socket  using the
 101 \fBSO_REUSEADDR\fR option to bind to a port its socket is bound to, the
 102 application can set the socket level option \fBSO_EXCLBIND\fR on a socket. The
 103 option values of 0 and 1 mean enabling and disabling the option respectively.
 104 Once this option is enabled on a socket, no other socket can be bound to the
 105 same port.
 106 .sp
 107 .LP
 108 Once a connection has been established, data can be exchanged using the
 109 \fBread\fR(2) and \fBwrite\fR(2) system calls.
 110 .sp
 111 .LP
 112 Under most circumstances, \fBTCP\fR sends data when it is presented. When
 113 outstanding data has not yet been acknowledged, \fBTCP\fR gathers small amounts
 114 of output to be sent in a single packet once an acknowledgement has been
 115 received. For a small number of clients, such as window systems that send a
 116 stream of mouse events which receive no replies, this packetization may cause
 117 significant delays. To circumvent this problem, \fBTCP\fR provides a
 118 socket-level boolean option, \fBTCP_NODELAY.\fR \fBTCP_NODELAY\fR is defined in
 119 \fB<netinet/tcp.h>\fR, and is set with \fBsetsockopt\fR(3SOCKET) and tested
 120 with \fBgetsockopt\fR(3SOCKET). The option level for the \fBsetsockopt()\fR
 121 call is the protocol number for \fBTCP,\fR available from
 122 \fBgetprotobyname\fR(3SOCKET).
 123 .sp
 124 .LP
```

For some applications, it may be desirable for TCP not to send out data unless a full TCP segment can be sent. To enable this behavior, an application can use the \fBTCP_CORK\fR socket option. When \fBTCP_CORK\fR is set with a non-zero value, TCP sends out a full TCP segment only. When \fBTCP_CORK\fR is set to zero after it has been enabled, all buffered data is sent out (as permitted by the peer's receive window and the current congestion window). \fBTCP_CORK\fR is defined in <\fBnetinet/tcp.h\fR>, and is set with \fBsetsockopt\fR(3SOCKET) and tested with \fBgetsockopt\fR(3SOCKET). The option level for the \fBsetsockopt()\fR call is the protocol number for TCP, available from \fBgetprotobyname\fR(3SOCKET).

.sp
.LP
The SO_RCVBUF socket level option can be used to control the window that TCP advertises to the peer. IP level options may also be used with TCP. See \fBip\fR(7P) and \fBip6\fR(7P).

.sp
.LP
Another socket level option, \fBSO_RCVBUF,\fR can be used to control the window that \fBTCP\fR advertises to the peer. \fBIP\fR level options may also be used with \fBTCP.\fR See \fBip\fR(7P) and \fBip6\fR(7P).

.sp
.LP
\fBTCP\fR provides an urgent data mechanism, which may be invoked using the out-of-band provisions of \fBsend\fR(3SOCKET). The caller may mark one byte as "urgent" with the \fBMSG_OOB\fR flag to \fBsend\fR(3SOCKET). This sets an "urgent pointer" pointing to this byte in the \fBTCP\fR stream. The receiver on the other side of the stream is notified of the urgent data by a \fBSIGURG\fR signal. The \fBSIOCATMARK\fR \fBioctl\fR(2) request returns a value indicating whether the stream is at the urgent mark. Because the system never returns data across the urgent mark in a single \fBread\fR(2) call, it is possible to advance to the urgent data in a simple loop which reads data, testing the socket with the \fBSIOCATMARK\fR \fBioctl()\fR request, until it reaches the mark.

.sp
.LP
Incoming connection requests that include an \fBIP\fR source route option are noted, and the reverse source route is used in responding.

.sp
.LP
A checksum over all data helps \fBTCP\fR implement reliability. Using a window-based flow control mechanism that makes use of positive acknowledgements, sequence numbers, and a retransmission strategy, \fBTCP\fR can usually recover when datagrams are damaged, delayed, duplicated or delivered out of order by the underlying communication medium.

.sp
.LP
If the local \fBTCP\fR receives no acknowledgements from its peer for a period of time, (for example, if the remote machine crashes), the connection is closed and an error is returned.

.sp
.LP
TCP follows the congestion control algorithm described in \fIRFC 2581\fR, and also supports the initial congestion window (cwnd) changes in \fIRFC 3390\fR. The initial cwnd calculation can be overridden by the socket option TCP_INIT_CWND. An application can use this option to set the initial cwnd to a specified number of TCP segments. This applies to the cases when the connection first starts and restarts after an idle period. The process must have the PRIV_SYS_NET_CONFIG privilege if it wants to specify a number greater than that calculated by \fIRFC 3390\fR.

.sp
.LP
SunOS supports \fBTCP\fR Extensions for High Performance (\fIRFC 1323\fR) which includes the window scale and time stamp options, and Protection Against Wrap Around Sequence Numbers (PAWS). SunOS also supports Selective Acknowledgment (SACK) capabilities (RFC 2018) and Explicit Congestion Notification (ECN) mechanism (\fIRFC 3168\fR).

.sp
.LP
Turn on the window scale option in one of the following ways:

.RS +4
.TP
.ie t \(bu
.el o
An application can set \fBSO_SNDBUF\fR or \fBSO_RCVBUF\fR size in the \fBsetsockopt()\fR option to be larger than 64K. This must be done \fIbefore\fR the program calls \fBlisten()\fR or \fBconnect()\fR, because the window scale option is negotiated when the connection is established. Once the connection has been made, it is too late to increase the send or receive window beyond the default \fBTCP\fR limit of 64K.
.RE

.RS +4
.TP
.ie t \(bu
.el o
For all applications, use \fBndd\fR(1M) to modify the configuration parameter \fBtcp_wscale_always\fR. If \fBtcp_wscale_always\fR is set to \fB1\fR, the window scale option will always be set when connecting to a remote system. If \fBtcp_wscale_always\fR is \fB0,\fR the window scale option will be set only if the user has requested a send or receive window larger than 64K. The default value of \fBtcp_wscale_always\fR is \fB1\fR.
.RE

.RS +4
.TP
.ie t \(bu
.el o
Regardless of the value of \fBtcp_wscale_always\fR, the window scale option will always be included in a connect acknowledgement if the connecting system has used the option.
.RE

.sp
.LP
Turn on \fBSACK\fR capabilities in the following way:

.RS +4
.TP
.ie t \(bu
.el o
Use \fBndd\fR to modify the configuration parameter \fBtcp_sack_permitted\fR. If \fBtcp_sack_permitted\fR is set to \fB0\fR, \fBTCP\fR will not accept \fBSACK\fR or send out \fBSACK\fR information. If \fBtcp_sack_permitted\fR is set to \fB1\fR, \fBTCP\fR will not initiate a connection with \fBSACK\fR permitted option in the \fBSYN\fR segment, but will respond with \fBSACK\fR permitted option in the \fBSYN|ACK\fR segment if an incoming connection request has the \fBSACK \fR permitted option. This means that \fBTCP\fR will only accept \fBSACK\fR information if the other side of the connection also accepts \fBSACK\fR information. If \fBtcp_sack_permitted\fR is set to \fB2\fR, it will both initiate and accept connections with \fBSACK\fR information. The default for \fBtcp_sack_permitted\fR is \fB2\fR (active enabled).
.RE

.sp
.LP
Turn on \fBTCP ECN\fR mechanism in the following way:

.RS +4
.TP
.ie t \(bu
.el o
Use \fBndd\fR to modify the configuration parameter \fBtcp_ecn_permitted\fR. If \fBtcp_ecn_permitted\fR is set to \fB0\fR, \fBTCP\fR will not negotiate with a peer that supports \fBECN\fR mechanism. If \fBtcp_ecn_permitted\fR is set to \fB1\fR when initiating a connection, TCP will not tell a peer that it supports ECN mechanism. However, it will tell a peer that it supports \fBECN\fR mechanism when accepting a new incoming connection request if the peer indicates that it supports \fBECN\fR mechanism in the SYN segment. If

tcp_ecn_permitted is set to 2, in addition to negotiating with a peer on ECN mechanism when accepting connections, TCP will indicate in the outgoing SYN segment that it supports \fBECN\fR mechanism when \fBTCP\fR makes active outgoing connections. The default for \fBtcp_ecn_permitted\fR is 1.
.RE
.sp
.LP
Turn on the time stamp option in the following way:
.RS +4
.TP
.ie t \(bu
.el o
Use \fBndd\fR to modify the configuration parameter \fBtcp_tstamp_always\fR. If \fBtcp_tstamp_always\fR is \fB1\fR, the time stamp option will always be set when connecting to a remote machine. If \fBtcp_tstamp_always\fR is \fB0\fR, the timestamp option will not be set when connecting to a remote system. The default for \fBtcp_tstamp_always\fR is \fB0\fR.
.RE
.RS +4
.TP
.ie t \(bu
.el o
Regardless of the value of \fBtcp_tstamp_always\fR, the time stamp option will always be included in a connect acknowledgement (and all succeeding packets) if the connecting system has used the time stamp option.
.RE
.sp
.LP
Use the following procedure to turn on the time stamp option only when the window scale option is in effect:
.RS +4
.TP
.ie t \(bu
.el o
Use \fBndd\fR to modify the configuration parameter \fBtcp_tstamp_if_wscale\fR. Setting \fBtcp_tstamp_if_wscale\fR to \fB1\fR will cause the time stamp option to be set when connecting to a remote system, if the window scale option has been set. If \fBtcp_tstamp_if_wscale\fR is \fB0\fR, the time stamp option will not be set when connecting to a remote system. The default for \fBtcp_tstamp_if_wscale\fR is \fB1\fR.
.RE
.sp
.LP
Protection Against Wrap Around Sequence Numbers (PAWS) is always used when the time stamp option is set.
.sp
.LP
SunOS also supports multiple methods of generating initial sequence numbers. One of these methods is the improved technique suggested in \fBRFC\fR 1948. We \fBHIGHLY\fR recommend that you set sequence number generation parameters as close to boot time as possible. This prevents sequence number problems on connections that use the same connection-ID as ones that used a different sequence number generation. The \fBsvc:/network/initial:default\fR service configures the initial sequence number generation. The service reads the value contained in the configuration file \fB/etc/default/inetinit\fR to determine which method to use.
.sp
.LP
The \fB/etc/default/inetinit\fR file is an unstable interface, and may change in future releases.
.sp
.LP
\fBTCP\fR may be configured to report some information on connections that terminate by means of an \fBRST\fR packet. By default, no logging is done. If the \fBndd\fR(1M) parameter \fIcp_trace\fR is set to 1, then trace data is collected for all new connections established after that time.

.sp
.LP
The trace data consists of the \fBTCP\fR headers and \fBIP\fR source and destination addresses of the last few packets sent in each direction before RST occurred. Those packets are logged in a series of \fBstrlog\fR(9F) calls. This trace facility has a very low overhead, and so is superior to such utilities as \fBsnoop\fR(1M) for non-intrusive debugging for connections terminating by means of an \fBRST\fR.
.sp
.LP
SunOS supports the keep-alive mechanism described in \fIRFC 1122\fR. It is enabled using the socket option SO_KEEPALIVE. When enabled, the first keep-alive probe is sent out after a TCP is idle for two hours If the peer does not respond to the probe within eight minutes, the TCP connection is aborted. You can alter the interval for sending out the first probe using the socket option TCP_KEEPALIVE_THRESHOLD. The option value is an unsigned integer in milliseconds. The system default is controlled by the TCP ndd parameter tcp_keepalive_interval. The minimum value is ten seconds. The maximum is ten days, while the default is two hours. If you receive no response to the probe, you can use the TCP_KEEPALIVE_ABORT_THRESHOLD socket option to change the time threshold for aborting a TCP connection. The option value is an unsigned integer in milliseconds. The value zero indicates that TCP should never time out and abort the connection when probing. The system default is controlled by the TCP ndd parameter tcp_keepalive_abort_interval. The default is eight minutes.
.sp
.LP
socket options TCP_KEEPIDLE, TCP_KEEPCNT and TCP_KEEPINTVL are also supported for compatibility with other Unix Flavors. TCP_KEEPIDLE option specifies the interval in seconds for sending out the first keep-alive probe. TCP_KEEPCNT specifies the number of keep-alive probes to be sent before aborting the connection in the event of no response from peer. TCP_KEEPINTVL specifies the interval in seconds between successive keep-alive probes.
.SH SEE ALSO
.sp
.LP
\fBsvcs\fR(1), \fBndd\fR(1M), \fBioctl\fR(2), \fBread\fR(2), \fBsvcadm\fR(1M), \fBwrite\fR(2), \fBaccept\fR(3SOCKET), \fBbind\fR(3SOCKET), \fBconnect\fR(3SOCKET), \fBgetprotobyname\fR(3SOCKET), \fBgetsockopt\fR(3SOCKET), \fBlisten\fR(3SOCKET), \fBsend\fR(3SOCKET), \fBsmf\fR(5), \fBinet\fR(7P), \fBinet6\fR(7P), \fBip\fR(7P), \fBip6\fR(7P)
.sp
.LP
Ramakrishnan, K., Floyd, S., Black, D., RFC 3168, \fIThe Addition of Explicit Congestion Notification (ECN) to IP\fR, September 2001.
.sp
.LP
Mathias, M. and Hahdavi, J. Pittsburgh Supercomputing Center; Ford, S. Lawrence Berkeley National Laboratory; Romanow, A. Sun Microsystems, Inc. \fIRFC 2018, TCP Selective Acknowledgement Options\fR, October 1996.
.sp
.LP
Bellovin, S., \fIRFC 1948, Defending Against Sequence Number Attacks\fR, May 1996.
.sp
.LP
Jacobson, V., Braden, R., and Borman, D., \fIRFC 1323, TCP Extensions for High Performance\fR, May 1992.
.sp
.LP
Postel, Jon, \fIRFC 793, Transmission Control Protocol - DARPA Internet Program Protocol Specification\fR, Network Information Center, SRI International, Menlo Park, CA., September 1981.
.SH DIAGNOSTICS
.sp
.LP

```
389 A socket operation may fail if:
390 .sp
391 .ne 2
392 .na
393 \fB\fBEISCONN\fR\fR
394 .ad
395 .RS 17n
396 A \fBconnect()\fR operation was attempted on a socket on which a
397 \fBconnect()\fR operation had already been performed.
398 .RE

400 .sp
401 .ne 2
402 .na
403 \fB\fBETIMEDOUT\fR\fR
404 .ad
405 .RS 17n
406 A connection was dropped due to excessive retransmissions.
407 .RE

409 .sp
410 .ne 2
411 .na
412 \fB\fBECONNRESET\fR\fR
413 .ad
414 .RS 17n
415 The remote peer forced the connection to be closed (usually because the remote
416 machine has lost state information about the connection due to a crash).
417 .RE

419 .sp
420 .ne 2
421 .na
422 \fB\fBECONNREFUSED\fR\fR
423 .ad
424 .RS 17n
425 The remote peer actively refused connection establishment (usually because no
426 process is listening to the port).
427 .RE

429 .sp
430 .ne 2
431 .na
432 \fB\fBEADDRINUSE\fR\fR
433 .ad
434 .RS 17n
435 A \fBbind()\fR operation was attempted on a socket with a network address/port
436 pair that has already been bound to another socket.
437 .RE

439 .sp
440 .ne 2
441 .na
442 \fB\fBEADDRNOTAVAIL\fR\fR
443 .ad
444 .RS 17n
445 A \fBbind()\fR operation was attempted on a socket with a network address for
446 which no network interface exists.
447 .RE

449 .sp
450 .ne 2
451 .na
452 \fB\fBEACCES\fR\fR
453 .ad
454 .RS 17n
```



```
455 A \fBbind()\fR operation was attempted with a "reserved" port number and the
456 effective user \fBID\fR of the process was not the privileged user.
457 .RE

459 .sp
460 .ne 2
461 .na
462 \fB\fBENOBUFS\fR\fR
463 .ad
464 .RS 17n
465 The system ran out of memory for internal data structures.
466 .RE

468 .SH NOTES
469 .sp
470 .LP
471 The \fBtcp\fR service is managed by the service management facility,
472 \fBsmf\fR(5), under the service identifier:
473 .sp
474 .in +2
475 .nf
476 svc:/network/initial:default
477 .fi
478 .in -2
479 .sp

481 .sp
482 .LP
483 Administrative actions on this service, such as enabling, disabling, or
484 requesting restart, can be performed using \fBsvcadm\fR(1M). The service's
485 status can be queried using the \fBsvcs\fR(1) command.
```