```
**********************************************************
   13902 Fri Dec 27 22:57:52 2013
new/usr/src/man/man3nsl/rpc_svc_create.3nsl
4344 Minor typos in the 3nsl man pages
**********************************************************
'\" te
.\"  Copyright 1989 AT&T
.\" Copyright (C) 2005, Sun Microsystems, Inc. All Rights Reserved.
.\" 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 RPC_SVC_CREATE 3NSL "Dec 27, 2013"
.TH RPC_SVC_CREATE 3NSL "Mar 22, 2005"
.SH NAME
rpc_svc_create, svc_control, svc_create, svc_destroy, svc_dg_create,
svc_fd_create, svc_raw_create, svc_tli_create, svc_tp_create, svc_vc_create,
svc_door_create \- server handle creation routines
.SH SYNOPSIS
.LP
.nf
#include <rpc/rpc.h>

\fBbool_t\fR \fBsvc_control\fR(\fBSVCXPRT *\fR\fIsvc\fR, \fBconst uint_t\fR \fIr
.fi

.LP
.nf
\fBint\fR \fBsvc_create\fR(\fBconst void (*\fR\fIdispatch\fR)(const struct svc_r
     const SVCXPRT *), \fBconst rpcprog_t\fR \fIprognum\fR, \fBconst rpcvers_t\f
\fBint\fR \fBsvc_create\fR(\fBconst void (*\fR\fIdispatch\fR)const struct svc_re
     const SVCXPRT *, \fBconst rpcprog_t\fR \fIprognum\fR, \fBconst rpcvers_t\fR
     \fBconst char *\fR\fInettype\fR);
.fi

.LP
.nf
\fBvoid\fR \fBsvc_destroy\fR(\fBSVCXPRT *\fR\fIxprt\fR);
.fi

.LP
.nf
\fBSVCXPRT *\fR\fBsvc_dg_create\fR(\fBconst int\fR \fIfildes\fR, \fBconst uint_t
     \fBconst uint_t\fR \fIrecvsz\fR);
.fi

.LP
.nf
\fBSVCXPRT *\fR\fBsvc_fd_create\fR(\fBconst int\fR \fIfildes\fR, \fBconst uint_t
     \fBconst uint_t\fR \fIrecvsz\fR);
.fi

.LP
.nf
\fBSVCXPRT *\fR\fBsvc_raw_create\fR(void)
.fi

.LP
.nf
\fBSVCXPRT *\fR\fBsvc_tli_create\fR(\fBconst int\fR \fIfildes\fR, \fBconst struc
     \fBconst struct t_bind *\fR\fIbind_addr\fR, \fBconst uint_t\fR \fIsendsz\fR
     \fBconst uint_t\fR \fIrecvsz\fR);
.fi

.LP
.nf
\fBSVCXPRT *\fR\fBsvc_tp_create\fR(\fBconst void (*\fR\fIdispatch\fR)
     (const struct svc_req *, const SVCXPRT *), \fBconst rpcprog_t\fR \fIprognum
     const struct svc_req *, const SVCXPRT *), \fBconst rpcprog_t\fR \fIprognum\
     \fBconst rpcvers_t\fR \fIversnum\fR, \fBconst struct netconfig *\fR\fInetco
.fi

.LP
.nf
\fBSVCXPRT *\fR\fBsvc_vc_create\fR(\fBconst int\fR \fIfildes\fR, \fBconst uint_t
     \fBconst uint_t\fR \fIrecvsz\fR);
.fi

.LP
.nf
\fBSVCXPRT *\fR\fBsvc_door_create\fR(\fBvoid (*\fR\fIdispatch\fR)(struct svc_req
     \fBconst rpcprog_t\fR \fIprognum\fR, \fBconst rpcvers_t\fR \fIversnum\fR,
     \fBconst uint_t\fR \fIsendsz\fR);
.fi

.SH DESCRIPTION
.sp
.LP
These routines are part of the \fBRPC\fR library which allows C language
programs to make procedure calls on servers across the network. These routines
deal with the creation of service handles. Once the handle is created, the
server can be invoked by calling \fBsvc_run()\fR.
.SS "Routines"
.sp
.LP
See \fBrpc\fR(3NSL) for the definition of the \fBSVCXPRT\fR data structure.
.sp
.ne 2
.na
\fB\fBsvc_control()\fR\fR
.ad
.RS 21n
A function to change or retrieve information about a service object. \fIreq\fR
indicates the type of operation and \fIinfo\fR is a pointer to the information.
The supported values of \fIreq\fR,  their argument types, and what they do are:
.sp
.ne 2
.na
\fB\fBSVCGET_VERSQUIET\fR\fR
.ad
.RS 25n
If a request is received for a program number served by this server but the
version number is outside the range registered with the server, an
\fBRPC_PROGVERSMISMATCH\fR error will normally be returned.  \fIinfo\fR should
be a pointer to an integer. Upon successful completion of the
\fBSVCGET_VERSQUIET\fR request,  *\fIinfo\fR contains an integer which
describes the server's current behavior:  \fB0\fR indicates normal server
behavior, that is, an  \fBRPC_PROGVERSMISMATCH\fR error will be returned.
\fB1\fR indicates that the out of range request will be silently ignored.
.RE

.sp
.ne 2
.na
\fB\fBSVCSET_VERSQUIET\fR\fR
.ad
.RS 25n
If a request is received for a program number served by this server but the
version number is outside the range registered with the server, an
\fBRPC_PROGVERSMISMATCH\fR error will normally be returned.  It is sometimes
desirable to change this behavior. \fIinfo\fR should be a pointer to an integer
which is either  \fB0\fR, indicating normal server behavior and an
\fBRPC_PROGVERSMISMATCH\fR error will be returned, or  \fB1\fR, indicating that
```

```
the out of range request should be silently ignored.
.RE

.sp
.ne 2
.na
\fB\fBSVCGET_XID\fR\fR
.ad
.RS 25n
Returns the transaction  \fBID\fR of connection\(mioriented and connectionless
transport service calls. The transaction  \fBID\fR assists in uniquely
identifying client requests for a given \fBRPC\fR version, program number,
procedure, and client. The transaction  \fBID\fR is extracted from the service
transport handle  \fIsvc\fR. \fIinfo\fR must be a pointer  to an unsigned long.
Upon successful completion of the  \fBSVCGET_XID\fR request,  *\fIinfo\fR
contains the transaction  \fBID\fR. Note that rendezvous and raw service
handles do not define a transaction  \fBID\fR. Thus, if the service handle is
of rendezvous or raw type, and the request is of type \fBSVCGET_XID,\fR
\fBsvc_control()\fR will return \fBFALSE\fR. Note also that the transaction
\fBID\fR read by the server can be set by the client through the suboption
\fBCLSET_XID\fR in  \fBclnt_control()\fR. See \fBclnt_create\fR(3NSL)
.RE

.sp
.ne 2
.na
\fB\fBSVCSET_RECVERRHANDLER\fR\fR
.ad
.RS 25n
Attaches or detaches a disconnection handler to the service handle, \fIsvc\fR,
that will be called when a transport error arrives during the reception of a
request or when the server is waiting for a request and the connection shuts
down. This handler is only useful for a connection oriented service handle.
.sp
\fI*info\fR contains the address of the error handler to attach, or \fINULL\fR
to detach a previously defined one. The error handler has two arguments. It has
a pointer to the erroneous service handle. It also has an integer that
indicates if the full service is closed (when equal to zero), or that only one
connection on this service is closed (when not equal to zero).
.sp
.in +2
.nf
void handler (const SVCXPRT *svc, const bool_t isAConnection);
.fi
.in -2

With the service handle address, \fIsvc\fR, the error handler is able to detect
which connection has failed and to begin an error recovery process. The error
handler can be called by multiple threads and should be implemented in an
MT-safe way.
.RE

.sp
.ne 2
.na
\fB\fBSVCGET_RECVERRHANDLER\fR\fR
.ad
.RS 25n
Upon successful completion of the \fBSVCGET_RECVERRHANDLER\fR request,
\fI*info\fR contains the address of the handler for receiving errors. Upon
failure, \fI*info\fR contains \fINULL\fR.
.RE

.sp
.ne 2
.na
```

```
\fB\fBSVCSET_CONNMAXREC\fR\fR
.ad
.RS 25n
Set the maximum record size (in bytes) and enable non-blocking mode for this
service handle. Value can be set and read for both connection and
non-connection oriented transports, but is silently ignored for the
non-connection oriented case. The \fIinfo\fR argument should be a pointer to an
\fBint\fR.
.RE

.sp
.ne 2
.na
\fB\fBSVCGET_CONNMAXREC\fR\fR
.ad
.RS 25n
Get the maximum record size for this service handle. Zero means no maximum in
effect and the connection is in blocking mode. The result is not significant
for non-connection oriented transports. The \fIinfo\fR argument should be a
pointer to an \fBint\fR.
.RE

This routine returns TRUE if the operation was successful. Otherwise, it
returns false.
.RE

.sp
.ne 2
.na
\fB\fBsvc_create()\fR\fR
.ad
.RS 21n
\fBsvc_create()\fR creates server handles for all the transports belonging to
the class \fInettype\fR.
.sp
\fInettype\fR defines a class of transports which can be used for a particular
application. The transports are tried in left to right order in \fBNETPATH\fR
variable or in top to bottom order in the netconfig database. If \fInettype\fR
is \fINULL,\fR it defaults to \fBnetpath\fR.
.sp
\fBsvc_create()\fR registers itself with the \fBrpcbind\fR service (see
\fBrpcbind\fR(1M)). \fIdispatch\fR is called when there is a remote procedure
call for the given \fIprognum\fR and \fIversnum\fR; this requires calling
\fBsvc_run()\fR (see \fBsvc_run()\fR in \fBrpc_svc_reg\fR(3NSL)). If
\fBsvc_create()\fR succeeds, it returns the number of server handles it
created, otherwise it returns \fB0\fR and an error message is logged.
.RE

.sp
.ne 2
.na
\fB\fBsvc_destroy()\fR\fR
.ad
.RS 21n
A function macro that destroys the \fBRPC\fR service handle \fIxprt\fR.
Destruction usually involves deallocation of private data structures, including
\fIxprt\fR itself.  Use of \fIxprt\fR is undefined after calling this routine.
.RE

.sp
.ne 2
.na
\fB\fBsvc_dg_create()\fR\fR
.ad
.RS 21n
This routine creates a connectionless \fBRPC\fR service handle, and returns a
```

```
pointer to it. This routine returns \fINULL\fR if it fails, and an error
message is logged. \fIsendsz\fR and \fIrecvsz\fR are parameters used to specify
the size of the buffers. If they are \fB0\fR, suitable defaults are chosen. The
file descriptor \fIfildes\fR should be open and bound. The server is not
registered with \fBrpcbind\fR(1M).
.sp
Warning: since connectionless-based \fBRPC\fR messages can only hold limited
amount of encoded data, this transport cannot be used for procedures that take
large arguments or return huge results.
.RE

.sp
.ne 2
.na
\fB\fBsvc_fd_create()\fR\fR
.ad
.RS 21n
This routine creates a service on top of an open and bound file descriptor, and
returns the handle to it. Typically, this descriptor is a connected file
descriptor for a connection-oriented transport. \fIsendsz\fR and \fIrecvsz\fR
indicate sizes for the send and receive buffers. If they are \fB0\fR,
reasonable defaults are chosen. This routine returns \fINULL\fR if it fails,
and an error message is logged.
.RE

.sp
.ne 2
.na
\fB\fBsvc_raw_create()\fR\fR
.ad
.RS 21n
This routine creates an \fBRPC\fR service handle and returns a pointer to it.
The transport is really a buffer within the process’s address space, so the
corresponding \fBRPC\fR client should live in the same address space; (see
\fBclnt_raw_create()\fR in \fBrpc_clnt_create\fR(3NSL)). This routine allows
simulation of \fBRPC\fR and acquisition of \fBRPC\fR overheads (such as round
trip times), without any kernel and networking interference. This routine
returns \fINULL\fR if it fails, and an error message is logged.
.sp
Note: \fBsvc_run()\fR should not be called when the raw interface is being
used.
.RE

.sp
.ne 2
.na
\fB\fBsvc_tli_create()\fR\fR
.ad
.RS 21n
This routine creates an \fBRPC\fR server handle, and returns a pointer to it.
\fIfildes\fR is the file descriptor on which the service is listening.  If
\fIfildes\fR is \fBRPC_ANYFD\fR, it opens a file descriptor on the transport
specified by \fInetconf\fR. If the file descriptor is unbound and
\fIbindaddr\fR is non-null \fIfildes\fR is bound to the address specified by
\fIbindaddr\fR, otherwise \fIfildes\fR is bound to a default address chosen by
the transport. In the case where the default address is chosen, the number of
outstanding connect requests is set to 8 for connection-oriented transports.
The user may specify the size of the send and receive buffers with the
parameters \fIsendsz\fR and \fIrecvsz\fR \fI;\fR values of \fB0\fR choose
suitable defaults. This routine returns \fINULL\fR if it fails, and an error
message is logged. The server is not registered with the \fBrpcbind\fR(1M)
service.
.RE

.sp
.ne 2
```

```
.na
\fB\fBsvc_tp_create()\fR\fR
.ad
.RS 21n
\fBsvc_tp_create()\fR creates a server handle for the network specified by
\fInetconf\fR, and registers itself with the \fBrpcbind\fR service.
\fIdispatch\fR is called when there is a remote procedure call for the given
\fIprognum\fR and \fIversnum\fR; this requires calling \fBsvc_run()\fR.
\fBsvc_tp_create()\fR returns the service handle if it succeeds, otherwise a
\fINULL\fR is returned and an error message is logged.
.RE

.sp
.ne 2
.na
\fB\fBsvc_vc_create()\fR\fR
.ad
.RS 21n
This routine creates a connection-oriented \fBRPC\fR service and returns a
pointer to it. This routine returns \fINULL\fR if it fails, and an error
message is logged. The users may specify the size of the send and receive
buffers with the parameters \fIsendsz\fR and \fIrecvsz\fR; values of \fB0\fR
choose suitable defaults. The file descriptor \fIfildes\fR should be open and
bound. The server is not registered with the \fBrpcbind\fR(1M) service.
.RE

.sp
.ne 2
.na
\fB\fBsvc_door_create()\fR\fR
.ad
.RS 21n
This routine creates an RPC server handle over doors and returns a pointer to
it. Doors is a transport mechanism that facilitates fast data transfer between
processes on the same machine. for the given program The user may set the size
of the send buffer with the parameter \fIsendsz\fR. If \fIsendsz\fR is 0, the
corresponding default buffer size is 16 Kbyte. If successful, the
\fBsvc_door_create()\fR routine returns the service handle. Otherwise it
returns \fINULL\fR and sets a value for \fBrpc_createerr\fR. The server is not
registered with \fBrpcbind\fR(1M). The \fBSVCSET_CONNMAXREC\fR and
\fBSVCGET_CONNMAXREC\fR \fBsvc_control()\fR requests can be used to set and
change the maximum allowed request size for the doors transport.
.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
_
Architecture    All
_
Interface Stability     Evolving
_
MT-Level        MT-Safe
.TE

.SH SEE ALSO
.sp
```

```
.LP
\fBrpcbind\fR(1M), \fBrpc\fR(3NSL), \fBrpc_clnt_create\fR(3NSL),
\fBrpc_svc_calls\fR(3NSL), \fBrpc_svc_err\fR(3NSL), \fBrpc_svc_reg\fR(3NSL),
\fBattributes\fR(5)
```

```
**********************************************************
    6620 Fri Dec 27 22:57:52 2013
new/usr/src/man/man3nsl/rpcbind.3nsl
4344 Minor typos in the 3nsl man pages
**********************************************************
'\" 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 RPCBIND 3NSL "Dec 27, 2013"
.TH RPCBIND 3NSL "Feb 20, 1998"
.SH NAME
rpcbind, rpcb_getmaps, rpcb_getaddr, rpcb_gettime, rpcb_rmtcall, rpcb_set,
rpcb_unset \- library routines for RPC bind service
.SH SYNOPSIS
.LP
.nf
#include <rpc/rpc.h>



\fBstruct rpcblist *\fR\fBrpcb_getmaps\fR(\fBconst struct netconfig *\fR\fInetco
\fBstruct rpcblist *\fR\fBrpcb_getmaps\fR(\fBconst struct netconfig *\fR\fInnetc
     \fBconst char *\fR\fIhost\fR);
.fi

.LP
.nf
\fBbool_t\fR  \fBrpcb_getaddr\fR(\fBconst rpcprog_t\fR  \fIprognum\fR, \fBconst
     \fBconst struct netconfig *\fR\fInetconf\fR, \fBstruct netbuf *\fR\fIssvcad
     \fBconst char *\fR\fIhost\fR);
.fi

.LP
.nf
\fBbool_t\fR \fBrpcb_gettime\fR(\fBconst char *\fR\fIhost\fR, \fBtime_t *\fR\fIt
.fi

.LP
.nf
\fBenum clnt_stat\fR \fBrpcb_rmtcall\fR(\fBconst struct netconfig *\fR\fInetconf
     \fBconst char *\fR\fIhost\fR, \fBconst rpcprog_t\fR \fIprognum\fR,
     \fBconst rpcvers_t\fR \fIversnum\fR, \fBconst rpcproc_t\fR \fIprocnum\fR,
     \fBconst xdrproc_t\fR \fIinproc\fR, \fBconst caddr_t\fR \fIin\fR,
     \fBconst xdrproc_t\fR \fIoutproc\fR \fBcaddr_t\fR \fIout\fR,
     \fBconst xdrproc_t\fR \fIoutproc\fR \fBcaddr_t\fR \fIout\fR,,
     \fBconst struct timeval\fR \fItout\fR, \fBstruct netbuf  *\fR\fIsvcaddr\fR)
.fi

.LP
.nf
\fBbool_t\fR \fBrpcb_set\fR(\fBconst rpcprog_t\fR \fIprognum\fR, \fBconst rpcver
     \fBconst struct netconfig *\fR\fInetconf\fR, \fBconst struct netbuf *\fR\fI
.fi

.LP
.nf
\fBbool_t\fR \fBrpcb_unset\fR(\fBconst rpcprog_t\fR \fIprognum\fR, \fBconst rpcv
     \fBconst struct netconfig *\fR\fInetconf\fR);
.fi

.SH DESCRIPTION
.sp
.LP
These routines allow client C programs to make procedure calls to the RPC
```

```
binder service. \fBrpcbind\fR maintains a list of mappings between programs and
their universal addresses. See \fBrpcbind\fR(1M).
.SS "Routines"
.sp
.ne 2
.na
\fB\fBrpcb_getmaps()\fR\fR
.ad
.RS 18n
An interface to the \fBrpcbind\fR service, which returns a list of the current
\fBRPC\fR program-to-address mappings on  \fIhost\fR. It uses the transport
specified through \fInetconf\fR to contact the remote \fBrpcbind\fR service on
\fIhost\fR. This routine will return \fBNULL,\fR if the remote \fBrpcbind\fR
could not be contacted.
.RE

.sp
.ne 2
.na
\fB\fBrpcb_getaddr()\fR\fR
.ad
.RS 18n
An interface to the \fBrpcbind\fR service, which finds the address of the
service on \fIhost\fR that is registered with program number \fIprognum\fR,
version \fIversnum\fR, and speaks the transport protocol associated with
\fInetconf\fR. The address found is returned in \fIsvcaddr\fR. \fIsvcaddr\fR
should be preallocated. This routine returns \fBTRUE\fR if it succeeds.  A
return value of \fBFALSE\fR means that the mapping does not exist or that the
\fBRPC\fR system failed to contact the remote \fBrpcbind\fR service. In the
latter case, the global variable \fBrpc_createerr\fR contains the \fBRPC\fR
status. See \fBrpc_clnt_create\fR(3NSL).
.RE

.sp
.ne 2
.na
\fB\fBrpcb_gettime()\fR\fR
.ad
.RS 18n
This routine returns the time on \fIhost\fR in \fItimep\fR. If \fIhost\fR is
\fINULL\fR, \fBrpcb_gettime()\fR returns the time on its own machine. This
routine returns \fBTRUE\fR if it succeeds, \fBFALSE\fR if it fails.
\fBrpcb_gettime()\fR can be used to synchronize the time between the client and
the remote server.   This routine is particularly useful for secure RPC.
.RE

.sp
.ne 2
.na
\fB\fBrpcb_rmtcall()\fR\fR
.ad
.RS 18n
An interface to the \fBrpcbind\fR service, which instructs \fBrpcbind\fR on
\fIhost\fR to make an \fBRPC\fR call on your behalf to a procedure on that
host. The  \fBnetconfig\fR structure should correspond to a connectionless
transport. The parameter \fB*\fR\fIsvcaddr\fR will be modified to the server's
address if the procedure succeeds. See  \fBrpc_call()\fR and \fBclnt_call()\fR
in \fBrpc_clnt_calls\fR(3NSL) for the definitions of other parameters.
.sp
This procedure should normally be used for a "ping" and nothing else. This
routine allows programs to do lookup and call, all in one step.
.sp
Note: Even if the server is not running \fBrpcbind\fR does not return any error
messages to the caller. In such a case, the caller times out.
.sp
Note: \fBrpcb_rmtcall()\fR is only available for connectionless transports.
```

```
.RE

.sp
.ne 2
.na
\fB\fBrpcb_set()\fR\fR
.ad
.RS 18n
An interface to the \fBrpcbind\fR service, which establishes a mapping between
the triple [\fIprognum\fR, \fIversnum\fR, \fInetconf\fR->\fInc_netid]\fR and
\fIsvcaddr\fR on the machine’s \fBrpcbind\fR service. The value of
\fInc_netid\fR must correspond to a network identifier that is defined by the
netconfig database. This routine returns \fBTRUE\fR if it succeeds, \fBFALSE\fR
otherwise. See also \fBsvc_reg()\fR in \fBrpc_svc_calls\fR(3NSL). If there
already exists such an entry with \fBrpcbind\fR, \fBrpcb_set()\fR will fail.
.RE

.sp
.ne 2
.na
\fB\fBrpcb_unset()\fR\fR
.ad
.RS 18n
An interface to the \fBrpcbind\fR service, which destroys the mapping between
the triple [\fIprognum\fR, \fIversnum\fR, \fInetconf\fR->\fInc_netid]\fR and
the address on the machine’s \fBrpcbind\fR service. If  \fInetconf\fR is
\fINULL\fR, \fBrpcb_unset()\fR destroys all mapping between the triple
[\fIprognum\fR, \fIversnum\fR, \fIall-transports\fR] and the addresses on the
machine’s \fBrpcbind\fR service. This routine returns \fBTRUE\fR if it
succeeds,  \fBFALSE\fR otherwise. Only the owner of the service or the
super-user can destroy the mapping. See also \fBsvc_unreg()\fR in
\fBrpc_svc_calls\fR(3NSL).
.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        MT-Safe
.TE

.SH SEE ALSO
.sp
.LP
\fBrpcbind\fR(1M), \fBrpcinfo\fR(1M), \fBrpc_clnt_calls\fR(3NSL),
\fBrpc_clnt_create\fR(3NSL), \fBrpc_svc_calls\fR(3NSL), \fBattributes\fR(5)
```

```
**********************************************************
   12964 Fri Dec 27 22:57:52 2013
new/usr/src/man/man3nsl/t_bind.3nsl
4344 Minor typos in the 3nsl man pages
**********************************************************
'\" te
.\"  Copyright 1994, The X/Open Company Ltd., All Rights Reserved. Portions Copy
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"  This notice shall appear on any product containing this material.
.\" 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 T_BIND 3NSL "Dec 27, 2013"
.TH T_BIND 3NSL "May 7, 1998"
.SH NAME
t_bind \- bind an address to a transport endpoint
.SH SYNOPSIS
.LP
.nf
#include <xti.h>




\fBint\fR \fBt_bind\fR(\fBint\fR \fIfd\fR, \fBconst struct t_bind *\fR\fIreq\fR,
.fi

.SH DESCRIPTION
.sp
.LP
This routine is part of the \fBXTI\fR interfaces that evolved from the
\fBTLI\fR interfaces. \fBXTI\fR represents the future evolution of these
interfaces. However, \fBTLI\fR interfaces are supported for compatibility. When
using a \fBTLI\fR routine that has the same name as an \fBXTI\fR routine, the
\fBtiuser.h\fR header file must be used.  Refer to the  \fBTLI\fR
\fBtiuser.h\fRheader file must be used.  Refer to the  \fBTLI\fR
\fBCOMPATIBILITY\fR section for a description of differences between the two
interfaces.
.sp
.LP
This function associates a protocol address with the transport endpoint
specified by \fIfd\fR and activates that transport endpoint. In connection
mode, the transport provider may begin enqueuing incoming connect indications,
or servicing a connection request on the transport endpoint. In
connectionless-mode, the transport user may send or receive data units through
the transport endpoint.
.sp
.LP
The \fIreq\fR and \fIret\fR arguments point to a \fBt_bind\fR structure
containing the following members:
.sp
.in +2
.nf
struct netbuf   addr;
unsigned        qlen;
.fi
.in -2

.sp
.LP
The \fIaddr\fR field of the \fBt_bind\fR structure specifies a protocol
address, and the \fIqlen\fR field is used to indicate the maximum number of
outstanding connection indications.
.sp
```

```
.LP
The parameter \fIreq\fR is used to request that an address, represented by the
\fBnetbuf\fR structure, be bound to the given transport endpoint. The parameter
\fIlen\fR specifies the number of bytes in the address, and \fIbuf\fR points to
the address buffer. The parameter \fImaxlen\fR has no meaning for the \fIreq\fR
argument. On return, \fIret\fR contains an encoding for the address that the
transport provider actually bound to the transport endpoint; if an address was
specified in  \fIreq\fR, this will be an encoding of the same address. In
\fIret\fR, the user specifies \fImaxlen,\fR which is the maximum size of the
address buffer, and \fIbuf\fR which points to the buffer where the address is
to be placed. On return, \fIlen\fR specifies the number of bytes in the bound
address, and \fIbuf\fR points to the bound address. If \fImaxlen\fR equals
zero, no address is returned. If  \fImaxlen\fR is greater than zero and less
than the length of the address,  \fBt_bind()\fR fails with \fBt_errno\fR set to
\fBTBUFOVFLW\fR.
.sp
.LP
If the requested address is not available, \fBt_bind()\fR will return  -1 with
\fBt_errno\fR set as appropriate. If no address is specified in \fIreq\fR (the
\fIlen\fR field of \fIaddr\fR in \fIreq\fR is zero or \fIreq\fR is
\fBNULL),\fR the transport provider will assign an appropriate address to be
bound, and will return that address in the \fIaddr\fR field of \fIret\fR. If
the transport provider could not allocate an address, \fBt_bind()\fR will fail
with \fBt_errno\fR set to \fBTNOADDR\fR.
.sp
.LP
The parameter \fIreq\fR may be a null pointer if the user does not wish to
specify an address to be bound. Here, the value of \fIqlen\fR is assumed to be
zero, and the transport provider will assign an address to the transport
endpoint. Similarly, \fIret\fR may be a null pointer if the user does not care
what address was bound by the provider and is not interested in the negotiated
value of \fIqlen\fR. It is valid to set \fIreq\fR and \fIret\fR to the null
pointer for the same call, in which case the provider chooses the address to
bind to the transport endpoint and does not return that information to the
user.
.sp
.LP
The \fIqlen\fR field has meaning only when initializing a connection-mode
service. It specifies the number of outstanding connection indications that the
transport provider should support for the given transport endpoint. An
outstanding connection indication is one that has been passed to the transport
user by the transport provider but which has not been accepted or rejected. A
value of \fIqlen\fR greater than zero is only meaningful when issued by a
passive transport user that expects other users to call it. The value of
\fIqlen\fR will be negotiated by the transport provider and may be changed if
the transport provider cannot support the specified number of outstanding
connection indications. However, this value of \fIqlen\fR will never be
negotiated from a requested value greater than zero to zero. This is a
requirement on transport providers; see \fBWARNINGS\fR below. On return, the
\fIqlen\fR field in \fIret\fR will contain the negotiated value.
.sp
.LP
If \fIfd\fR refers to a connection-mode service, this function allows more than
one transport endpoint to be bound to the same protocol address.  But it is not
one transport endpoint to be bound to the same protocol address.  but it is not
possible to bind more than one protocol address to the same transport endpoint.
However, the transport provider must also support this capability. If a user
binds more than one transport endpoint to the same protocol address, only one
endpoint can be used to listen for connection indications associated with that
protocol address. In other words, only one \fBt_bind()\fR for a given protocol
address may specify a value of \fIqlen\fR greater than zero. In this way, the
transport provider can identify which transport endpoint should be notified of
an incoming connection indication. If a user attempts to bind a protocol
address to a second transport endpoint with a value of \fIqlen\fR greater than
zero, \fBt_bind()\fR will return  -1 and set \fBt_errno\fR to \fBTADDRBUSY\fR.
When a user accepts a connection on the transport endpoint that is being used
```

```
as the listening endpoint, the bound protocol address will be found to be busy
for the duration of the connection, until a \fBt_unbind\fR(3NSL) or
\fBt_close\fR(3NSL) call has been issued. No other transport endpoints may be
bound for listening on that same protocol address while that initial listening
endpoint is active (in the data transfer phase or in the  \fBT_IDLE\fR state).
This will prevent more than one transport endpoint bound to the same protocol
address from accepting connection indications.
.sp
.LP
If  \fIfd\fR refers to connectionless mode service, this function allows for
more than one transport endpoint to be associated with a protocol address,
where the underlying transport provider supports this capability (often in
conjunction with value of a protocol-specific option). If a user attempts to
bind a second transport endpoint to an already bound protocol address when such
capability is not supported for a transport provider, \fBt_bind()\fR will
return  -1 and set \fBt_errno\fR to \fBTADDRBUSY\fR.
.SH RETURN VALUES
.sp
.LP
Upon successful completion, a value of 0 is returned.  Otherwise, a value of
-1 is returned and \fBt_errno\fR is set to indicate an error.
.SH VALID STATES
.sp
.LP
\fBT_UNBND\fR
.SH ERRORS
.sp
.LP
On failure, \fBt_errno\fR is set to one of the following:
.sp
.ne 2
.na
\fB\fBTACCES\fR\fR
.ad
.RS 13n
The user does not have permission to use the specified address.
.RE

.sp
.ne 2
.na
\fB\fBTADDRBUSY\fR\fR
.ad
.RS 13n
The requested address is in use.
.RE

.sp
.ne 2
.na
\fB\fBTBADADDR\fR\fR
.ad
.RS 13n
The specified protocol address was in an incorrect format or contained illegal
information.
.RE

.sp
.ne 2
.na
\fB\fBTBADF\fR\fR
.ad
.RS 13n
The specified file descriptor does not refer to a transport endpoint.
.RE
```

```
.sp
.ne 2
.na
\fB\fBTBUFOVFLW\fR\fR
.ad
.RS 13n
The number of bytes allowed for an incoming argument \fI(maxlen)\fR is greater
than 0 but not sufficient to store the value of that argument. The provider’s
state will change to  \fBT_IDLE\fR and the information to be returned in
\fIret\fR will be discarded.
.RE

.sp
.ne 2
.na
\fB\fBTOUTSTATE\fR\fR
.ad
.RS 13n
The communications endpoint referenced by  \fIfd\fR is not in one of the states
in which a call to this function is valid.
.RE

.sp
.ne 2
.na
\fB\fBTNOADDR\fR\fR
.ad
.RS 13n
The transport provider could not allocate an address.
.RE

.sp
.ne 2
.na
\fB\fBTPROTO\fR\fR
.ad
.RS 13n
This error indicates that a communication problem has been detected between XTI
and the transport provider for which there is no other suitable XTI error
\fB(t_errno)\fR.
.RE

.sp
.ne 2
.na
\fB\fBTSYSERR\fR\fR
.ad
.RS 13n
A system error has occurred during execution of this function.
.RE

.SH TLI COMPATIBILITY
.sp
.LP
The \fBXTI\fR and \fBTLI\fR interface definitions have common names but use
different header files. This, and other semantic differences between the two
interfaces are described in the subsections below.
.SS "Interface Header"
.sp
.LP
The \fBXTI\fR interfaces use the header file, \fBxti.h\fR. \fBTLI\fR interfaces
should \fInot\fR use this header.  They should use the header:
.sp
.LP
\fB#include\fR \fB<tiuser.h>\fR
.SS "Address Bound"
```

```
.sp
.LP
The user can compare the addresses in \fIreq\fR and \fIret\fR to determine
whether the transport provider bound the transport endpoint to a different
address than that requested.
.SS "Error Description Values"
.sp
.LP
The \fBt_errno\fR values \fBTPROTO\fR and \fBTADDRBUSY\fR can be set by the
\fBXTI\fR interface but cannot be set by the \fBTLI\fR interface.
.sp
.LP
A \fBt_errno\fR value that this routine can return under different
circumstances than its \fBXTI\fR counterpart is \fBTBUFOVFLW\fR. It can be
returned even when the \fBmaxlen\fR field of the corresponding buffer has been
set to zero.
.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
\fBt_accept\fR(3NSL), \fBt_alloc\fR(3NSL), \fBt_close\fR(3NSL),
\fBt_connect\fR(3NSL), \fBt_unbind\fR(3NSL), \fBattributes\fR(5)
.SH WARNINGS
.sp
.LP
The requirement that the value of \fIqlen\fR never be negotiated from a
requested value greater than zero to zero implies that transport providers,
rather than the XTI implementation itself, accept this restriction.
.sp
.LP
An implementation need not allow an application explicitly to bind more than
one communications endpoint to a single protocol address, while permitting more
than one connection to be accepted to the same protocol address. That means
that although an attempt to bind a communications endpoint to some address with
\fIqlen=0\fR might be rejected with \fBTADDRBUSY\fR, the user may nevertheless
use this (unbound) endpoint as a responding endpoint in a call to
\fBt_accept\fR(3NSL). To become independent of such implementation differences,
the user should supply unbound responding endpoints to  \fBt_accept\fR(3NSL).
.sp
.LP
The local address bound to an endpoint may change as result of a
\fBt_accept\fR(3NSL) or  \fBt_connect\fR(3NSL) call. Such changes are not
necessarily reversed when the connection is released.
```