new/usr/src/man/man3nsl/rpc_svc_create.3nsl 1 new/usr/src/man/man3nsl/rpc_svc_create.3nsl 59 13902 Fri Dec 27 22:57:52 2013 59 new/usr/src/man/man3nsl/rpc_svc_create.3nsl 60 4344 Minor typos in the 3nsl man pages 61 .fi 1 '\" te 63 .LP 2 .\" Copyright 1989 AT&T 64 .nf 3 . \" Copyright (C) 2005, Sun Microsystems, Inc. All Rights Reserved. 4 .\" The contents of this file are subject to the terms of the Common Development 66 5 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http: 67 .fi 6 . \" When distributing Covered Code, include this CDDL HEADER in each file and in 7 .TH RPC_SVC_CREATE 3NSL "Dec 27, 2013" 69 .LP 7 .TH RPC SVC CREATE 3NSL "Mar 22, 2005" 70 .nf 8 .SH NAME 9 rpc_svc_create, svc_control, svc_create, svc_destroy, svc_dg_create, 72 10 svc_fd_create, svc_raw_create, svc_tli_create, svc_tp_create, svc_vc_create, 73 11 svc_door_create \- server handle creation routines 74 .fi 12 .SH SYNOPSIS 13 .LP 76 .SH DESCRIPTION 14 .nf 77 .sp 78 .LP 15 #include <rpc/rpc.h> 17 \fBbool_t\fR \fBsvc_control\fR(\fBSVCXPRT *\fR\fIsvc\fR, \fBconst uint_t\fR \fIr 18 .fi 83 .SS "Routines" 20 .LP 21 .nf 84 .sp 22 \fBint\fR \fBsvc create\fR(\fBconst void (*\fR\fIdispatch\fR)(const struct svc r 85 .LP 23 const SVCXPRT *), \fBconst rpcprog_t\fR \fIprognum\fR, \fBconst rpcvers_t\f 22 \fBint\fR \fBsvc_create\fR(\fBconst void (*\fR\fIdispatch\fR)const struct svc_re 87 .sp 23 const SVCXPRT *, \fBconst rpcprog_t\fR \fIprognum\fR, \fBconst rpcvers_t\fR 88 .ne 2 24 \fBconst char *\fR\fInettype\fR); 89 .na 25 .fi 90 \fB\fBsvc_control()\fR\fR 91 .ad 27 .LP 92 RS 21n 28 .nf 29 \fBvoid\fR \fBsvc_destroy\fR(\fBSVCXPRT *\fR\fIxprt\fR); 30 .fi 96 .sp 32 .LP 97 .ne 2 33 .nf 98 .na 34 \fBSVCXPRT *\fR\fBsvc_dg_create\fR(\fBconst int\fR \fIfildes\fR, \fBconst uint_t 35 \fBconst uint_t\fR \fIrecvsz\fR); 100 .ad 36 .fi 101 .RS 25n 38 .LP 39 .nf 40 \fBSVCXPRT *\fR\fBsvc_fd_create\fR(\fBconst int\fR \fIfildes\fR, \fBconst uint t 41 \fBconst uint_t\fR \fIrecvsz\fR); 42 .fi 44 .LP 110 .RE 45 .nf 46 \fBSVCXPRT *\fR\fBsvc_raw_create\fR(void) 112 .sp 47 .fi 113 .ne 2 49 .LP 114 .na 50 .nf 51 \fBSVCXPRT *\fR\fBsvc tli create\fR(\fBconst int\fR \fIfildes\fR, \fBconst struc 116 .ad \fBconst struct t_bind *\fR\fIbind_addr\fR, \fBconst uint_t\fR \fIsendsz\fR 117 .RS 25n 52 \fBconst uint_t\fR \fIrecvsz\fR); 53 54 .fi 120 \fBRPC_PROGVERSMISMATCH\fR error will normally be returned. It is sometimes 56 .LP 121 desirable to change this behavior. \flinfo\fR should be a pointer to an integer 57 .nf 58 \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 65 \fBSVCXPRT *\fR\fBsvc vc create\fR(\fBconst int\fR \fIfildes\fR, \fBconst uint t \fBconst uint_t\fR \fIrecvsz\fR); 71 \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); 79 These routines are part of the \fBRPC\fR library which allows C language 80 programs to make procedure calls on servers across the network. These routines 81 deal with the creation of service handles. Once the handle is created, the 82 server can be invoked by calling \fBsvc_run()\fR. 86 See \fBrpc\fR(3NSL) for the definition of the \fBSVCXPRT\fR data structure. 93 A function to change or retrieve information about a service object. \fIreg\fR 94 indicates the type of operation and flinfoR is a pointer to the information. 95 The supported values of \fIreq\fR, their argument types, and what they do are: 99 \fb\fbSVCGET_VERSQUIET\fr\fr 102 If a request is received for a program number served by this server but the 103 version number is outside the range registered with the server, an 104 \fBRPC_PROGVERSMISMATCH\fR error will normally be returned. \flinfo\fR should 105 be a pointer to an integer. Upon successful completion of the 106 \fBSVCGET_VERSQUIET\fR request, *\flinfo\fR contains an integer which 107 describes the server's current behavior: \fB0\fR indicates normal server 108 behavior, that is, an \fBRPC_PROGVERSMISMATCH\fR error will be returned. 109 \fB1\fR indicates that the out of range request will be silently ignored. 115 \fb\fbSVCSET_VERSQUIET\fr\fr 118 If a request is received for a program number served by this server but the 119 version number is outside the range registered with the server, an

2

122 which is either \fB0\fR, indicating normal server behavior and an

123 \fBRPC_PROGVERSMISMATCH\fR error will be returned, or \fB1\fR, indicating that

new/usr/src/man/man3nsl/rpc svc create.3nsl

124 the out of range request should be silently ignored.

125 .RE

127 .sp

- 128 .ne 2
- 129 .na
- 130 \fB\fBSVCGET XID\fR\fR
- 131 .ad
- 132 .RS 25n
- 133 Returns the transaction \fBID\fR of connection\(mioriented and connectionless 134 transport service calls. The transaction \fBID\fR assists in uniquely 135 identifying client requests for a given \fBRPC\fR version, program number, 136 procedure, and client. The transaction \fBID\fR is extracted from the service 137 transport handle \fIsvc\fR. \fIinfo\fR must be a pointer to an unsigned long. 138 Upon successful completion of the \fBSVCGET_XID\fR request, *\flinfo\fR 139 contains the transaction \fBID\fR. Note that rendezvous and raw service 140 handles do not define a transaction $fBLD\FR$. Thus, if the service handle is 141 of rendezvous or raw type, and the request is of type \fBSVCGET_XID,\fR 142 \fBsvc_control()\fR will return \fBFALSE\fR. Note also that the transaction 143 \fBID\fR read by the server can be set by the client through the suboption 144 \fBCLSET_XID\fR in \fBclnt_control()\fR. See \fBclnt_create\fR(3NSL) 145 .RE
- 147 .sp
- 148 .ne 2
- 149 .na
- 150 \fb\fbsvcset recverrhandler\fr\fr
- 151 .ad 152 .RS 25n
- 153 Attaches or detaches a disconnection handler to the service handle, \fIsvc\fR, 154 that will be called when a transport error arrives during the reception of a 155 request or when the server is waiting for a request and the connection shuts 156 down. This handler is only useful for a connection oriented service handle. 157 .sp 158 \fI*info\fR contains the address of the error handler to attach, or \fINULL\fR 159 to detach a previously defined one. The error handler has two arguments. It has 160 a pointer to the erroneous service handle. It also has an integer that 161 indicates if the full service is closed (when equal to zero), or that only one 162 connection on this service is closed (when not equal to zero).
- 163 .sp
- 164 .in +2
- 165 .nf
- 166 void handler (const SVCXPRT *svc, const bool t isAConnection);
- 167 .fi
- 168 .in -2
- 170 With the service handle address, \fIsvc\fR, the error handler is able to detect 171 which connection has failed and to begin an error recovery process. The error 172 handler can be called by multiple threads and should be implemented in an 173 MT-safe way. 174 .RE
- 176 .sp
- 177 .ne 2 178 .na
- 179 \fb\fbsvcget recverrhandler\fr\fr 180 .ad
- 181 .RS 25n
- 182 Upon successful completion of the \fBSVCGET RECVERRHANDLER\fR request,
- 183 \fI*info\fR contains the address of the handler for receiving errors. Upon 184 failure, \fI*info\fR contains \fINULL\fR.
- 185 .RE
- 187 .sp
- 188 .ne 2
- 189 .na

new/usr/src/man/man3nsl/rpc svc create.3nsl

- 190 \fb\fbSVCSET CONNMAXREC\fr\fr
- 191 .ad 192 .RS 25n

3

- 193 Set the maximum record size (in bytes) and enable non-blocking mode for this
- 194 service handle. Value can be set and read for both connection and
- 195 non-connection oriented transports, but is silently ignored for the
- 196 non-connection oriented case. The \flinfo\fR argument should be a pointer to an
- 197 \fBint\fR. 198 .RE
- 200 .sp
- 201 .ne 2
- 202 .na 203 \fB\fBSVCGET_CONNMAXREC\fR\fR
- 204 .ad
- 205 .RS 25n
- 206 Get the maximum record size for this service handle. Zero means no maximum in
- 207 effect and the connection is in blocking mode. The result is not significant
- 208 for non-connection oriented transports. The \flinfo\fR argument should be a
- 209 pointer to an \fBint\fR.
- 210 .RE

212 This routine returns TRUE if the operation was successful. Otherwise, it 213 returns false.

- 214 .RE
- 216 .sp
- 217 .ne 2
- 218 .na
- 219 \fB\fBsvc create()\fR\fR
- 220 .ad 221 .RS 21n
- 222 $fBsvc_create()$ reates server handles for all the transports belonging to
- 223 the class fInettypefR.
- 224 .sp
- 225 \fInettype\fR defines a class of transports which can be used for a particular 226 application. The transports are tried in left to right order in \fBNETPATH\fR
- 227 variable or in top to bottom order in the netconfig database. If \fInettype\fR
- 228 is \fINULL,\fR it defaults to \fBnetpath\fR.
- 229 .sp
- 230 \fBsvc create()\fR registers itself with the \fBrpcbind\fR service (see
- 231 \fBrpcbind\fR(1M)). \fIdispatch\fR is called when there is a remote procedure
- 232 call for the given \fIprognum\fR and \fIversnum\fR; this requires calling
- 233 \fBsvc_run()\fR (see \fBsvc_run()\fR in \fBrpc_svc_reg\fR(3NSL)). If
- 234 \fBsvc_create()\fR succeeds, it returns the number of server handles it 235 created, otherwise it returns \fB0\fR and an error message is logged.
- 236 .RE
- 238 .sp
- 239 .ne 2
- 240 .na
- 241 \fB\fBsvc_destroy()\fR\fR
- 242 .ad 243 .RS 21n
- 244 A function macro that destroys the \fBRPC\fR service handle \fIxprt\fR.
- 245 Destruction usually involves deallocation of private data structures, including 246 \fIxprt\fR itself. Use of \fIxprt\fR is undefined after calling this routine.
- 247 .RE
- 249 .sp 250 .ne 2
- 251 na
- 252 \fB\fBsvc_dg_create()\fR\fR
- 253 .ad
- 254 .RS 21n
- 255 This routine creates a connectionless \fBRPC\fR service handle, and returns a

new/usr/src/man/man3nsl/rpc_svc_create.3nsl

5

256 pointer to it. This routine returns \fINULL\fR if it fails, and an error 257 message is logged. \fIsendsz\fR and \fIrecvsz\fR are parameters used to specify 258 the size of the buffers. If they are $fB0\R$, suitable defaults are chosen. The 259 file descriptor \fIfildes\fR should be open and bound. The server is not 260 registered with \fBrpcbind\fR(1M). 261 .sp 262 Warning: since connectionless-based \fBRPC\fR messages can only hold limited 263 amount of encoded data, this transport cannot be used for procedures that take 264 large arguments or return huge results. 265 .RE 267 .sp 268 .ne 2 269 .na 270 \fB\fBsvc_fd_create()\fR\fR 271 .ad 272 .RS 21n 273 This routine creates a service on top of an open and bound file descriptor, and 274 returns the handle to it. Typically, this descriptor is a connected file 275 descriptor for a connection-oriented transport. \flsendsz\fR and \flrecvsz\fR 276 indicate sizes for the send and receive buffers. If they are \fB0\fR, 277 reasonable defaults are chosen. This routine returns \fINULL\fR if it fails, 278 and an error message is logged. 279 .RE 281 .sp 282 .ne 2 283 .na 284 \fB\fBsvc_raw_create()\fR\fR 285 .ad 286 .RS 21n 287 This routine creates an \fBRPC\fR service handle and returns a pointer to it. 288 The transport is really a buffer within the process's address space, so the 289 corresponding \fBRPC\fR client should live in the same address space; (see 290 \fBclnt_raw_create()\fR in \fBrpc_clnt_create\fR(3NSL)). This routine allows 291 simulation of \fBRPC\fR and acquisition of \fBRPC\fR overheads (such as round 292 trip times), without any kernel and networking interference. This routine 293 returns \fINULL\fR if it fails, and an error message is logged. 294 .sp 295 Note: \fBsvc_run()\fR should not be called when the raw interface is being 296 used. 297 .RE 299 .sp 300 .ne 2 301 .na 302 \fB\fBsvc_tli_create()\fR\fR 303 .ad 304 .RS 21n 305 This routine creates an \fBRPC\fR server handle, and returns a pointer to it. 306 \fIfildes\fR is the file descriptor on which the service is listening. If 307 \fIfildes\fR is \fBRPC_ANYFD\fR, it opens a file descriptor on the transport 308 specified by \fInetconf\fR. If the file descriptor is unbound and 309 \flbindaddr\fR is non-null \flfildes\fR is bound to the address specified by 310 \flbindaddr\fR, otherwise \flfildes\fR is bound to a default address chosen by 311 the transport. In the case where the default address is chosen, the number of 312 outstanding connect requests is set to 8 for connection-oriented transports. 313 The user may specify the size of the send and receive buffers with the 314 parameters \fIsendsz\fR and \fIrecvsz\fR \fI;\fR values of \fB0\fR choose 315 suitable defaults. This routine returns \fINULL\fR if it fails, and an error 316 message is logged. The server is not registered with the fBrpcbind R(1M)317 service 318 .RE 320 .sp 321 .ne 2

new/usr/src/man/man3nsl/rpc svc create.3nsl

322 .na

- 323 \fB\fBsvc_tp_create()\fR\fR
- 324 .ad
- 325 .RS 21n
- 326 \fBsvc_tp_create()\fR creates a server handle for the network specified by
- 327 fInetconffR, and registers itself with the fBrpcbindfR service.
- 328 \fIdispatch\fR is called when there is a remote procedure call for the given
- 329 \fIprognum\fR and \fIversnum\fR; this requires calling \fBsvc_run()\fR.
- 330 \fBsvc_tp_create()\fR returns the service handle if it succeeds, otherwise a
- 331 \fINULL\fR is returned and an error message is logged.
- 332 .RE
- 334 .sp
- 335 .ne 2
- 336 .na
- 337 \fB\fBsvc_vc_create()\fR\fR
- 338 .ad 339 .RS 21n

340 This routine creates a connection-oriented \fBRPC\fR service and returns a 341 pointer to it. This routine returns \fINULL\fR if it fails, and an error 342 message is logged. The users may specify the size of the send and receive 343 buffers with the parameters \fIsendsz\fR and \fIrecvsz\fR; values of \fBO\fR 344 choose suitable defaults. The file descriptor \fIfildes\fR should be open and 345 bound. The server is not registered with the \fBrpcbind\fR(1M) service. 346 .RE

- 348 .sp 349 .ne 2
- 350 .na
- 351 \fB\fBsvc_door_create()\fR\fR
- 352 .ad
- 353 .RS 21n

354 This routine creates an RPC server handle over doors and returns a pointer to 355 it. Doors is a transport mechanism that facilitates fast data transfer between

- 356 processes on the same machine. for the given program The user may set the size
- 357 of the send buffer with the parameter \fIsendsz\fR. If \fIsendsz\fR is 0, the
- 358 corresponding default buffer size is 16 Kbyte. If successful, the
- 359 \fBsvc_door_create()\fR routine returns the service handle. Otherwise it
- 360 returns finullfR and sets a value for $fBrpc_createerrfR$. The server is not
- 361 registered with fBrpcbind(fR(1M)). The $fBSVCSET_CONNMAXREC(fR and fR(1M))$
- 362 \fBSVCGET_CONNMAXREC\fR \fBsvc_control()\fR requests can be used to set and
- 363 change the maximum allowed request size for the doors transport.
- 364 .RE
- 366 .SH ATTRIBUTES
- 367 .sp
- 368 .LP
- 369 See fBattributes fR(5) for descriptions of the following attributes: 370 .sp

372 .sp 373 .TS 374 box; 375 c | С 376 1 1 377 ATTRIBUTE TYPE ATTRIBUTE VALUE 378 379 Architecture A11 380 381 Interface Stability Evolving 382 383 MT-Level MT-Safe 384 .TE 386 .SH SEE ALSO 387 .sp

new/usr/src/man/man3nsl/rpc_svc_create.3nsl

- 388 .LP
 388 \fBrpcbind\fR(1M), \fBrpc\fR(3NSL), \fBrpc_clnt_create\fR(3NSL),
 390 \fBrpc_svc_calls\fR(3NSL), \fBrpc_svc_err\fR(3NSL), \fBrpc_svc_reg\fR(3NSL),
 391 \fBattributes\fR(5)

new/usr/src/man/man3nsl/rpcbind.3nsl 1 new/usr/src/man/man3nsl/rpcbind.3nsl 6620 Fri Dec 27 22:57:52 2013 new/usr/src/man/man3nsl/rpcbind.3nsl 4344 Minor typos in the 3nsl man pages 1 '\" te 2 . Copyright 1989 AT&T Copyright (c) 1997, Sun Microsystems, Inc. All Rights 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 RPCBIND 3NSL "Dec 27, 2013" 6 .TH RPCBIND 3NSL "Feb 20, 1998" 7 .SH NAME 8 rpcbind, rpcb_getmaps, rpcb_getaddr, rpcb_gettime, rpcb_rmtcall, rpcb_set, 9 rpcb_unset \- library routines for RPC bind service 10 .SH SYNOPSIS 11 .LP 12 .nf 13 #include <rpc/rpc.h> 17 \fBstruct rpcblist *\fR\fBrpcb_getmaps\fR(\fBconst struct netconfig *\fR\fInetco 17 \fBstruct rpcblist *\fR\fBrpcb_getmaps\fR(\fBconst struct netconfig *\fR\fInnetc 18 \fBconst char *\fR\fIhost\fR); 19 .fi 21 .LP 22 .nf 23 \fBbool_t\fR \fBrpcb_getaddr\fR(\fBconst rpcprog_t\fR \fIprognum\fR, \fBconst 24 \fBconst struct netconfig *\fR\fInetconf\fR, \fBstruct netbuf *\fR\fIssvcad 25 \fBconst char *\fR\fIhost\fR); 26 .fi 28 .LP 29 .nf 30 \fBbool_t\fR \fBrpcb_gettime\fR(\fBconst char *\fR\fIhost\fR, \fBtime_t *\fR\fIt 31 .fi 33 .LP 34 .nf 35 \fBenum clnt_stat\fR \fBrpcb_rmtcall\fR(\fBconst struct netconfig *\fR\fInetconf \fBconst char *\fR\fIhost\fR, \fBconst rpcprog_t\fR \fIprognum\fR, 36 37 \fBconst rpcvers_t\fR \fIversnum\fR, \fBconst rpcproc_t\fR \fIprocnum\fR, \fBconst xdrproc_t\fR \flinproc\fR, \fBconst caddr_t\fR \flin\fR, 38 39 \fBconst xdrproc_t\fR \fIoutproc\fR \fBcaddr_t\fR \fIout\fR, 39 \fBconst xdrproc_t\fR \fIoutproc\fR \fBcaddr_t\fR \fIout\fR,, 40 \fBconst struct timeval\fR \fItout\fR, \fBstruct netbuf *\fR\fIsvcaddr\fR) 41 .fi 43 .LP 44 .nf 45 \fBbool_t\fR \fBrpcb_set\fR(\fBconst rpcprog_t\fR \fIprognum\fR, \fBconst rpcver 46 \fBconst struct netconfig *\fR\fInetconf\fR, \fBconst struct netbuf *\fR\fI 47 .fi 49 .LP 50 .nf 51 \fBbool_t\fR \fBrpcb_unset\fR(\fBconst rpcprog_t\fR \fIprognum\fR, \fBconst rpcv \fBconst struct netconfig *\fR\fInetconf\fR); 52 53 .fi 55 .SH DESCRIPTION 56 .sp 57 .LP 58 These routines allow client C programs to make procedure calls to the RPC

59 binder service. \fBrpcbind\fR maintains a list of mappings between programs and 60 their universal addresses. See \fBrpcbind\fR(1M). 61 SS "Routines" 62 .sp 63 .ne 2 64 .na 65 \fB\fBrpcb_getmaps()\fR\fR 66 .ad 67 .RS 18n 68 An interface to the \fBrpcbind\fR service, which returns a list of the current 69 \fBRPC\fR program-to-address mappings on \fIhost\fR. It uses the transport 70 specified through \fInetconf\fR to contact the remote \fBrpcbind\fR service on 71 \fIhost\fR. This routine will return \fBNULL,\fR if the remote \fBrpcbind\fR 72 could not be contacted. 73 RE 75 .sp 76 .ne 2 77 .na 78 \fB\fBrpcb_getaddr()\fR\fR 79 .ad 80 .RS 18n 81 An interface to the \fBrpcbind\fR service, which finds the address of the 82 service on \flhost\fR that is registered with program number \flprognum\fR, 83 version \fIversnum\fR, and speaks the transport protocol associated with 84 \fInetconf\fR. The address found is returned in \fIsvcaddr\fR. \fIsvcaddr\fR 85 should be preallocated. This routine returns \fBTRUE\fR if it succeeds. A 86 return value of \fBFALSE\fR means that the mapping does not exist or that the 87 \fBRPC\fR system failed to contact the remote \fBrpcbind\fR service. In the 88 latter case, the global variable \fBrpc_createerr\fR contains the \fBRPC\fR 89 status. See \fBrpc_clnt_create\fR(3NSL). 90 .RE 92 .sp 93 .ne 2 94 .na 95 \fB\fBrpcb_gettime()\fR\fR 96 .ad 97 .RS 18n 98 This routine returns the time on \flhost\fR in \fltimep\fR. If \flhost\fR is 99 \fINULL\fR, \fBrpcb gettime()\fR returns the time on its own machine. This 100 routine returns \fBTRUE\fR if it succeeds, \fBFALSE\fR if it fails. 101 \fBrpcb_gettime()\fR can be used to synchronize the time between the client and 102 the remote server. This routine is particularly useful for secure RPC. 103 RE 105 .sp 106 .ne 2

- 107 .na
- 108 \fB\fBrpcb rmtcall()\fR\fR
- 109 .ad

110 .RS 18n 111 An interface to the fBrpcbindfR service, which instructs fBrpcbindfR on 112 \fIhost\fR to make an \fBRPC\fR call on your behalf to a procedure on that 113 host. The \fBnetconfig\fR structure should correspond to a connectionless 114 transport. The parameter \fB*\fR\fIsvcaddr\fR will be modified to the server's 115 address if the procedure succeeds. See \fBrpc_call()\fR and \fBclnt_call()\fR 116 in \fBrpc_clnt_calls\fR(3NSL) for the definitions of other parameters. 117 .sp 118 This procedure should normally be used for a "ping" and nothing else. This 119 routine allows programs to do lookup and call, all in one step. 120 .sp 121 Note: Even if the server is not running \fBrpcbind\fR does not return any error 122 messages to the caller. In such a case, the caller times out. 123 .sp

124 Note: \fBrpcb_rmtcall()\fR is only available for connectionless transports.

new/usr/src/man/man3nsl/rpcbind.3nsl

125 .RE

127 .sp 128 .ne 2 129 .na 130 \fB\fBrpcb_set()\fR\fR 131 .ad 132 .RS 18n 133 An interface to the \fBrpcbind\fR service, which establishes a mapping between 134 the triple [flprognum/fR, flversnum/fR, $flnetconf/fR->flnc_netid]$ fR and 135 \fIsvcaddr\fR on the machine's \fBrpcbind\fR service. The value of 136 \fInc_netid\fR must correspond to a network identifier that is defined by the 137 netconfig database. This routine returns \fBTRUE\fR if it succeeds, \fBFALSE\fR 138 otherwise. See also $fBsvc_reg()\fR$ in $fBrpc_svc_calls$ (3NSL). If there 139 already exists such an entry with fBrpcbind, fR, $fBrpcb_set()\fR$ will fail. 140 .RE 142 .sp 143 .ne 2 144 .na 145 \fB\fBrpcb_unset()\fR\fR 146 .ad 147 .RS 18n 148 An interface to the $fBrpcbind\fR$ service, which destroys the mapping between 149 the triple [\fIprognum\fR, \fIversnum\fR, \fInetconf\fR->\fInc_netid]\fR and 150 the address on the machine's \fBrpcbind\fR service. If \fInetconf\fR is 151 \fINULL\fR, \fBrpcb_unset()\fR destroys all mapping between the triple 152 [\flprognum\fR, \flversnum\fR, \flall-transports\fR] and the addresses on the 153 machine's \fBrpcbind\fR service. This routine returns \fBTRUE\fR if it 154 succeeds, \fBFALSE\fR otherwise. Only the owner of the service or the 155 super-user can destroy the mapping. See also \fBsvc_unreg()\fR in 156 \fBrpc_svc_calls\fR(3NSL). 157 .RE 159 .SH ATTRIBUTES 160 .sp 161 .LP 162 See \fBattributes\fR(5) for descriptions of the following attributes: 163 .sp 165 .sp 166 .TS 167 box; 168 c | c 169 1 1 . 170 ATTRIBUTE TYPE ATTRIBUTE VALUE 171 _ 172 MT-Level MT-Safe 173 .TE 175 .SH SEE ALSO 176 .sp 177 .LP 178 \fBrpcbind\fR(1M), \fBrpcinfo\fR(1M), \fBrpc_clnt_calls\fR(3NSL),

179 \fBrpc_clnt_create\fR(3NSL), \fBrpc_svc_calls\fR(3NSL), \fBattributes\fR(5)

new/usr/src/man/man3nsl/t_bind.3nsl

12964 Fri Dec 27 22:57:52 2013 new/usr/src/man/man3nsl/t_bind.3nsl 4344 Minor typos in the 3nsl man pages

- 1 '\" te
- 2 .\" Copyright 1994, The X/Open Company Ltd., All Rights Reserved. Portions Copy
- 3 ./" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
- 4 .\" http://www.opengroup.org/bookstore/.
- 5 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
- 6 . \" This notice shall appear on any product containing this material.
- 7 . \" The contents of this file are subject to the terms of the Common Development
- 8 . \" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
- 9 . \" When distributing Covered Code, include this CDDL HEADER in each file and in
- 10 .TH T_BIND 3NSL "Dec 27, 2013"
- 10 .TH T_BIND 3NSL "May 7, 1998"
- 11 .SH NAME
- 12 t_bind \- bind an address to a transport endpoint
- 13 .SH SYNOPSIS
- 14 .LP
- 15 .nf
- 16 #include <xti.h>
- 21 \fBint\fR \fBt_bind\fR(\fBint\fR \fIfd\fR, \fBconst struct t_bind *\fR\fIreq\fR, 22 .fi
- 24 .SH DESCRIPTION
- 25 .sp
- 26 .LP

27 This routine is part of the \fBXTI\fR interfaces that evolved from the

- 28 \fBTLI\fR interfaces. \fBXTI\fR represents the future evolution of these
- 29 interfaces. However, $fBTLI\fR$ interfaces are supported for compatibility. When
- 30 using a \fBTLI\fR routine that has the same name as an \fBXTI\fR routine, the

31 \fBtiuser.h\fR header file must be used. Refer to the \fBTLI\fR

31 \fBtiuser.h\fRheader file must be used. Refer to the \fBTLI\fR

- 32 \fBCOMPATIBILITY\fR section for a description of differences between the two
- 33 interfaces.
- 34 .sp

35 .LP

36 This function associates a protocol address with the transport endpoint 37 specified by $fIfd\f and activates that transport endpoint. In connection$

38 mode, the transport provider may begin enqueuing incoming connect indications,

39 or servicing a connection request on the transport endpoint. In

- 40 connectionless-mode, the transport user may send or receive data units through 41 the transport endpoint.
- 42 .sp
- 43 .LP

44 The \fIreq\fR and \fIret\fR arguments point to a \fBt_bind\fR structure 45 containing the following members:

- 46 .sp
- 47 .in +2 48 .nf
- 49 struct netbuf
- 50 unsigned glen;
- 51 .fi
- 52 .in -2
- 54 .sp

55 .LP

- 56 The \fIaddr\fR field of the \fBt_bind\fR structure specifies a protocol
- 57 address, and the flqlenfR field is used to indicate the maximum number of
- 58 outstanding connection indications.

addr;

59 .sp

new/usr/src/man/man3nsl/t_bind.3nsl

60 .LP

1

61 The parameter \fIreq\fR is used to request that an address, represented by the 62 \fBnetbuf\fR structure, be bound to the given transport endpoint. The parameter 63 \fIlen\fR specifies the number of bytes in the address, and \fIbuf\fR points to 64 the address buffer. The parameter \fImaxlen\fR has no meaning for the \fIreq\fR 65 argument. On return, \fIret\fR contains an encoding for the address that the 66 transport provider actually bound to the transport endpoint; if an address was 67 specified in \fIreq\fR, this will be an encoding of the same address. In 68 \fIret\fR, the user specifies \fImaxlen,\fR which is the maximum size of the 69 address buffer, and \fIbuf\fR which points to the buffer where the address is 70 to be placed. On return, \fIlen\fR specifies the number of bytes in the bound 71 address, and \fIbuf\fR points to the bound address. If \fImaxlen\fR equals 72 zero, no address is returned. If \fImaxlen\fR fails with \fBt_errno\fR set to 74 \fBTBUF0VFLW\fR. 75 .sp

76 .LP

77 If the requested address is not available, \fBt_bind()\fR will return -1 with
78 \fBt_errno\fR set as appropriate. If no address is specified in \fIreq\fR (the
79 \fIlen\fR field of \fIaddr\fR in \fIreq\fR is zero or \fIreq\fR is
80 \fBNULL),\fR the transport provider will assign an appropriate address to be
81 bound, and will return that address in the \fIaddr\fR field of \fIret\fR. If
82 the transport provider could not allocate an address, \fBt_bind()\fR will fail
83 with \fBt_errno\fR set to \fBTNOADDR\fR.
84 .sp
85 .LP
86 The parameter \fIreq\fR may be a null pointer if the user does not wish to
87 specify an address to be bound. Here, the value of \fIqlen\fR is assumed to be
80 or address to be bound. Here, the value of \fIqlen\fR is assumed to be
81 be the the two provider of the

88 zero, and the transport provider will assign an address to the transport 89 endpoint. Similarly, \fIret\fR may be a null pointer if the user does not care 90 what address was bound by the provider and is not interested in the negotiated 91 value of \fIqlen\fR. It is valid to set \fIreq\fR and \fIret\fR to the null 92 pointer for the same call, in which case the provider chooses the address to

- 93 bind to the transport endpoint and does not return that information to the
- 94 user
- 95 .sp

96 .LP

97 The \fIqlen\fR field has meaning only when initializing a connection-mode 98 service. It specifies the number of outstanding connection indications that the 99 transport provider should support for the given transport endpoint. An 100 outstanding connection indication is one that has been passed to the transport 101 user by the transport provider but which has not been accepted or rejected. A 102 value of \fIqlen\fR greater than zero is only meaningful when issued by a 103 passive transport user that expects other users to call it. The value of 104 \fIqlen\fR will be negotiated by the transport provider and may be changed if 105 the transport provider cannot support the specified number of outstanding 106 connection indications. However, this value of \fIqlen\fR will never be 107 negotiated from a requested value greater than zero to zero. This is a 108 requirement on transport providers; see \fBWARNINGS\fR below. On return, the 109 \fIqlen\fR field in \fIret\fR will contain the negotiated value. 110 .sp

112 If \fIfd\fR refers to a connection-mode service, this function allows more than 113 one transport endpoint to be bound to the same protocol address. But it is not 113 one transport endpoint to be bound to the same protocol address. But it is not 114 possible to bind more than one protocol address to the same transport endpoint. 115 However, the transport provider must also support this capability. If a user 116 binds more than one transport endpoint to the same protocol address, only one 117 endpoint can be used to listen for connection indications associated with that 118 protocol address. In other words, only one \fBt_bind()\fR for a given protocol 119 address may specify a value of \fIqlen\fR greater than zero. In this way, the 120 transport provider can identify which transport endpoint should be notified of 121 an incoming connection indication. If a user attempts to bind a protocol 122 address to a second transport endpoint with a value of \fIqlen\fR greater than 123 zero, \fBt_bind()\fR will return -1 and set \fBt_errno\fR to \fBTADDRBUSY\fR. 124 When a user accepts a connection on the transport endpoint that is being used

new/usr/src/man/man3nsl/t_bind.3nsl	3	ne
125 as the listening endpoint, the bound protocol address will be found to be bus 126 for the duration of the connection, until a \fBt_unbind\fR(3NSL) or 127 \fBt_close\fR(3NSL) call has been issued. No other transport endpoints may be 128 bound for listening on that same protocol address while that initial listenir 129 endpoint is active (in the data transfer phase or in the \fBT_IDLE\fR state) 130 This will prevent more than one transport endpoint bound to the same protocol 131 address from accepting connection indications. 132 .sp	G	1 1 1 1 1 1 1 1 1
133 .LP 134 If \flfd\fR refers to connectionless mode service, this function allows for 135 more than one transport endpoint to be associated with a protocol address, 136 where the underlying transport provider supports this capability (often in 137 conjunction with value of a protocol-specific option). If a user attempts to 138 bind a second transport endpoint to an already bound protocol address when su 139 capability is not supported for a transport provider, \fBt_bind()\fR will 140 return -1 and set \fBt_errno\fR to \fBTADDRBUSY\fR. 141 .SH RETURN VALUES 142 .sp	lch	1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
<pre>143 .LP 144 Upon successful completion, a value of 0 is returned. Otherwise, a value of 145 -1 is returned and \fBt_errno\fR is set to indicate an error. 146 .SH VALID STATES 147 .sp 148 .LP</pre>		2 2 2 2 2 2 2
149 \fBT_UNBND\fR 150 .SH ERRORS 151 .sp 152 .LP 153 On failure, \fBt_errno\fR is set to one of the following: 154 .sp 155 .ne 2 156 .na		2 2 2 2 2 2 2 2 2 2 2 2 2
157 \fB\fBTACCES\fR\fR 158 .ad 159 .RS 13n 160 The user does not have permission to use the specified address. 161 .RE		2 2 2 2 2 2 2 2
<pre>163 .sp 164 .ne 2 165 .na 166 \fB\fBTADDRBUSY\fR\fR 167 .ad 168 .RS 13n 169 The requested address is in use. 170 .RE</pre>		2 2 2 2 2 2 2 2 2 2 2
<pre>172 .sp 173 .ne 2 174 .na 175 \fB\fBTBADADDR\fR\fR 176 .ad 177 .RS 13n 178 The specified protocol address was in an incorrect format or contained illega 179 information. 180 .RE</pre>	.1	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
<pre>182 .sp 183 .ne 2 184 .na 185 \fB\fBTBADF\fR\fR 186 .ad 187 .RS 13n 188 The specified file descriptor does not refer to a transport endpoint. 189 .RE</pre>		2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

new/usr/src/man/man3nsl/t_bind.3nsl

191 .sp 192 .ne 2 193 .na 194 \fb\fbTBUFOVFLW\fr\fr 195 .ad 196 .RS 13n 197 The number of bytes allowed for an incoming argument \fI(maxlen)\fR is greater 198 than 0 but not sufficient to store the value of that argument. The provider's 199 state will change to \fBT_IDLE\fR and the information to be returned in 200 \fIret\fR will be discarded. 201 .RE 203 .sp 204 .ne 2 205 .na 206 \fb\fbTOUTSTATE\fR\fR 207 .ad 208 .RS 13n 209 The communications endpoint referenced by \flfd\fR is not in one of the states 210 in which a call to this function is valid. 211 .RE 213 .sp 214 .ne 2 215 .na 216 \fb\fbTNOADDR\fr\fr 217 .ad 218 .RS 13n 219 The transport provider could not allocate an address. 220 .RE 222 .sp 223 .ne 2 224 .na 225 \fb\fbTpROTO\fr\fr 226 .ad 227 .RS 13n 228 This error indicates that a communication problem has been detected between XTI 229 and the transport provider for which there is no other suitable XTI error 230 \fB(t_errno)\fR. 231 .RE 233 .sp 234 .ne 2 235 .na 236 \fb\fbTSYSERR\fr\fr 237 .ad 238 .RS 13n 239 A system error has occurred during execution of this function. 240 .RE 242 .SH TLI COMPATIBILITY 243 .sp 244 .LP 245 The \fBXTI\fR and \fBTLI\fR interface definitions have common names but use 246 different header files. This, and other semantic differences between the two 247 interfaces are described in the subsections below. 248 .SS "Interface Header" 249 .sp 250 .LP 251 The \fBXTI\fR interfaces use the header file, \fBxti.h\fR. \fBTLI\fR interfaces 252 should \fInot\fR use this header. They should use the header: 253 .sp 254 .LP 255 \fB#include\fR \fB<tiuser.h>\fR 256 .SS "Address Bound"

new/usr/src/man/man3nsl/t_bind.3nsl

```
257 .sp
258 .LP
259 The user can compare the addresses in \fIreq\fR and \fIret\fR to determine
260 whether the transport provider bound the transport endpoint to a different
261 address than that requested.
262 .SS "Error Description Values"
263 .sp
264 .LP
265 The \fBt_errno\fR values \fBTPROTO\fR and \fBTADDRBUSY\fR can be set by the
266 fBXTI fR interface but cannot be set by the fBTLI fR interface.
267 .sp
268 .LP
269 A \fBt errno\fR value that this routine can return under different
270 circumstances than its \fBXTI\fR counterpart is \fBTBUFOVFLW\fR. It can be
271 returned even when the \fBmaxlen\fR field of the corresponding buffer has been
272 set to zero.
273 .SH ATTRIBUTES
274 .sp
275 .LP
276 See \fBattributes\fR(5) for descriptions of the following attributes:
277 .sp
279 .sp
282 c | c
283 1 1 .
284 ATTRIBUTE TYPE ATTRIBUTE VALUE
286 MT Level
                    Safe
287 .TE
289 .SH SEE ALSO
290 .sp
291 .LP
292 \fBt_accept\fR(3NSL), \fBt_alloc\fR(3NSL), \fBt_close\fR(3NSL),
293 \fBt_connect\fR(3NSL), \fBt_unbind\fR(3NSL), \fBattributes\fR(5)
294 .SH WARNINGS
295 .sp
296 .LP
297 The requirement that the value of flqlen\R never be negotiated from a
298 requested value greater than zero to zero implies that transport providers,
299 rather than the XTI implementation itself, accept this restriction.
300 .sp
301 .LP
302 An implementation need not allow an application explicitly to bind more than
303 one communications endpoint to a single protocol address, while permitting more
304 than one connection to be accepted to the same protocol address. That means
305 that although an attempt to bind a communications endpoint to some address with
306 \fIglen=0\fR might be rejected with \fBTADDRBUSY\fR, the user may nevertheless
307 use this (unbound) endpoint as a responding endpoint in a call to
308 \fBt_accept\fR(3NSL). To become independent of such implementation differences,
309 the user should supply unbound responding endpoints to \fBt_accept\fR(3NSL).
310 .sp
311 .LP
312 The local address bound to an endpoint may change as result of a
313 \fBt_accept\fR(3NSL) or \fBt_connect\fR(3NSL) call. Such changes are not
```

280 .TS 281 box;

285

314 necessarily reversed when the connection is released.