```
**********************************************************
   12532 Mon Oct 28 19:32:57 2013
new/usr/src/man/man3nsl/rpc_svc_calls.3nsl
4239 rpc_svc_calls(3nsl): Typo service service
**********************************************************
'\" te
.\"  Copyright 1989 AT&T
.\" Copyright (C) 2004 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_CALLS 3NSL "Oct 28, 2013"
.TH RPC_SVC_CALLS 3NSL "Jan 26, 2004"
.SH NAME
rpc_svc_calls, svc_dg_enablecache, svc_done, svc_exit, svc_fdset, svc_freeargs,
svc_getargs, svc_getreq_common, svc_getreq_poll, svc_getreqset,
svc_getrpccaller, svc_max_pollfd, svc_pollfd, svc_run, svc_sendreply,
svc_getcallerucred, svc_fd_negotiate_ucred \- library routines for RPC servers
.SH SYNOPSIS
.LP
.nf
\fBcc\fR [ \fIflag\fR... ] \fIfile\fR... \fB-lnsl\fR [ \fIlibrary\fR...]
#include <rpc/rpc.h>

\fBint\fR \fBsvc_dg_enablecache\fR(\fBSVCXPRT *\fR\fIxprt\fR, \fBconst uint_t\fR
.fi

.LP
.nf
\fBint\fR \fBsvc_done\fR(\fBSVCXPRT *\fR\fIxprt\fR);
.fi

.LP
.nf
\fBvoid\fR \fBsvc_exit\fR(\fBvoid\fR);
.fi

.LP
.nf
\fBvoid\fR \fBsvc_fd_negotiate_ucred\fR(\fBint\fR \fIfd\fR);
.fi

.LP
.nf
\fBbool_t\fR \fBsvc_freeargs\fR(\fBconst SVCXPRT *\fR\fIxprt\fR, \fBconst txdrpr
     \fBcaddr_t\fR \fIin\fR);
.fi

.LP
.nf
\fBbool_t\fR \fBsvc_getargs\fR(\fBconst SVCXPRT *\fR\fIxprt\fR, \fBconst xdrproc
     \fBcaddr_t\fR \fIin\fR);
.fi

.LP
.nf
\fBint\fR \fBsvc_getcallerucred\fR(\fBconst SVCXPRT *\fR\fIxprt\fR, \fBucred_t *
.fi

.LP
.nf
\fBvoid\fR \fBsvc_getreq_common\fR(\fBconst int\fR \fIfd\fR);
.fi

.LP
.nf
```



```
\fBvoid\fR \fBsvc_getreqset\fR(\fBfd_set *\fR\fIrdfds\fR);
.fi

.LP
.nf
\fBvoid\fR \fBsvc_getreq_poll\fR(\fBstruct pollfd *\fR\fIpfdp\fR, \fBconst int\f
.fi

.LP
.nf
\fBstruct netbuf *\fR\fBsvc_getrpccaller\fR(\fBconst SVCXPRT *\fR\fIxprt\fR);
.fi

.LP
.nf
\fBvoid\fR \fBsvc_run\fR(\fBvoid\fR);
.fi

.LP
.nf
\fBbool_t\fR \fBsvc_sendreply\fR(\fBconst SVCXPRT *\fR\fIxprt\fR, \fBconst xdrpr
     \fBcaddr_t\fR \fIout\fRint svc_max_pollfd;
fd_set svc_fdset;
pollfd_t *svc_pollfd;
.fi

.SH DESCRIPTION
.sp
.LP
These routines are part of the \fBRPC\fR library which allows C language
programs to make procedure calls on other machines across the network.
.sp
.LP
These routines are associated with the server side of the \fBRPC\fR mechanism.
Some of them are called by the server side dispatch function. Others, such as
\fBsvc_run()\fR, are called when the server is initiated.
.sp
.LP
Because the service transport handle \fBSVCXPRT\fR contains a single data area
for decoding arguments and encoding results, the structure cannot freely be
shared between threads that call functions to decode arguments and encode
results. When a server is operating in the Automatic or User MT modes, however,
a copy of this structure is passed to the service dispatch procedure in order
to enable concurrent request processing. Under these circumstances, some
routines which would otherwise be Unsafe, become Safe. These are marked as
such. Also marked are routines that are Unsafe for multithreaded applications,
and are not to be used by such applications. See \fBrpc\fR(3NSL) for the
definition of the \fBSVCXPRT\fR data structure.
.sp
.LP
The \fBsvc_dg_enablecache()\fR function allocates a duplicate request cache for
the service endpoint \fIxprt\fR, large enough to hold \fIcache_size\fR entries.
Once enabled, there is no way to disable caching. The function returns \fB1\fR
if space necessary for a cache of the given size was successfully allocated,
and \fB0\fR otherwise. This function is Safe in multithreaded applications.
.sp
.LP
The \fBsvc_done()\fR function frees resources allocated to service a client
request directed to the service endpoint \fIxprt\fR. This call pertains only to
servers executing in the User MT mode. In the User MT mode, service procedures
must invoke this call before returning, either after a client request has been
serviced, or after an error or abnormal condition that prevents a reply from
being sent. After \fBsvc_done\fR() is invoked, the service endpoint \fIxprt\fR
should not be referenced by the service procedure. Server multithreading modes
and parameters can be set using the \fBrpc_control\fR() call. This function is
Safe in multithreaded applications. It will have no effect if invoked in modes
```

other than the User MT mode.
.sp
.LP
The \fBsvc_exit()\fR function when called by any of the RPC server procedures or otherwise, destroys all services registered by the server and causes \fBsvc_run()\fR to return. If RPC server activity is to be resumed, services must be reregistered with the RPC library either through one of the \fBrpc_svc_create\fR(3NSL) functions, or using \fBxprt_register\fR(3NSL). The \fBsvc_exit()\fR function has global scope and ends all RPC server activity.
.sp
.LP
The \fBsvc_freeargs()\fR function macro frees any data allocated by the \fBRPC/XDR\fR system when it decoded the arguments to a service procedure using \fBsvc_getargs()\fR. This routine returns \fBTRUE\fR if the results were successfully freed, and \fBFALSE\fR otherwise. This function macro is Safe in multithreaded applications utilizing the Automatic or User MT modes.
.sp
.LP
The \fBsvc_getargs()\fR function macro decodes the arguments of an \fBRPC\fR request associated with the \fBRPC\fR service transport handle \fIxprt\fR. The parameter \fIin\fR is the address where the arguments will be placed; \fIinproc\fR is the \fBXDR\fR routine used to decode the arguments. This routine returns \fBTRUE\fR if decoding succeeds, and \fBFALSE\fR otherwise. This function macro is Safe in multithreaded applications utilizing the Automatic or User MT modes.
.sp
.LP
The \fBsvc_getreq_common()\fR function is called to handle a request on a file descriptor.
.sp
.LP
The \fBsvc_getreq_poll()\fR function is only of interest if a service implementor does not call \fBsvc_run()\fR, but instead implements custom asynchronous event processing. It is called when \fBpoll\fR(2) has determined that an RPC request has arrived on some RPC file descriptors; \fIpollretval\fR is the return value from \fBpoll\fR(2) and \fIpfdp\fR is the array of \fIpollfd\fR structures on which the \fBpoll\fR(2) was done. It is assumed to be an array large enough to contain the maximal number of descriptors allowed. The \fBsvc_getreq_poll()\fR function macro is Unsafe in multithreaded applications.
.sp
.LP
The \fBsvc_getreqset()\fR function is only of interest if a service implementor does not call \fBsvc_run()\fR, but instead implements custom asynchronous event processing. It is called when \fBselect\fR(3C) has determined that an \fBRPC\fR request has arrived on some \fBRPC\fR file descriptors; \fIrdfds\fR is the resultant read file descriptor bit mask. The routine returns when all file descriptors associated with the value of \fIrdfds\fR have been serviced. This function macro is Unsafe in multithreaded applications.
.sp
.LP
The \fBsvc_getrpccaller()\fR function is the approved way of getting the network address of the caller of a procedure associated with the \fBRPC\fR service transport handle \fIxprt\fR. This function macro is Safe in multithreaded applications.
.sp
.LP
The \fBsvc_run()\fR function never returns. In single-threaded mode, the function waits for \fBRPC\fR requests to arrive. When an RPC request arrives, the \fBsvc_run()\fR function calls the appropriate service procedure. This procedure is usually waiting for the \fBpoll\fR(2) library call to return.
.sp
.LP
Applications that execute in the Automatic or the User MT mode should invoke the \fBsvc_run()\fR function exactly once. In the Automatic MT mode, the \fBsvc_run()\fR function creates threads to service client requests. In the

User MT mode, the function provides a framework for service developers to create and manage their own threads for servicing client requests.
.sp
.LP
The \fBsvc_fdset\fR global variable reflects the \fBRPC\fR server's read file descriptor bit mask. This is only of interest if service implementors do not call \fBsvc_run()\fR, but rather do their own asynchronous event processing. This variable is read-only may change after calls to \fBsvc_getreqset()\fR or after any creation routine. Do not pass its address to \fBselect\fR(3C). Instead, pass the address of a copy. multithreaded applications executing in either the Automatic MT mode or the user MT mode should never read this variable. They should use auxiliary threads to do asynchronous event processing. The \fBsvc_fdset\fR variable is limited to 1024 file descriptors and is considered obsolete. Use of \fBsvc_pollfd\fR is recommended instead.
.sp
.LP
The \fBsvc_pollfd\fR global variable points to an array of \fBpollfd_t\fR structures that reflect the \fBRPC\fR server's read file descriptor array. This **is only of interest if service implementors do not call \fBsvc_run()\fR** *is only of interest if service service implementors do not call \fBsvc_run()\fR* but rather do their own asynchronous event processing. This variable is read-only, and it may change after calls to \fBsvc_getreg_poll()\fR or any creation routines. Do no pass its address to \fBpoll\fR(2). Instead, pass the address of a copy. By default, \fBsvc_pollfd\fR is limited to 1024 entries. Use \fBrpc_control\fR(3NSL) to remove this limitation. multithreaded applications executing in either the Automatic MT mode or the user MT mode should never be read this variable. They should use auxiliary threads to do asynchronous event processing.
.sp
.LP
The \fBsvc_max_pollfd\fR global variable contains the maximum length of the \fBsvc_pollfd\fR array. This variable is read-only, and it may change after calls to \fBsvc_getreg_poll()\fR or any creation routines.
.sp
.LP
The \fBsvc_sendreply()\fR function is called by an \fBRPC\fR service dispatch routine to send the results of a remote procedure call. The \fIxprt\fR parameter is the transport handle of the request. The \fIoutproc\fR parameter is the \fBXDR\fR routine used to encode the results. The \fIout\fR parameter is the address of the results. This routine returns \fBTRUE\fR if it succeeds, \fBFALSE\fR otherwise. The \fBsvc_sendreply()\fR function macro is Safe in multithreaded applications that use the Automatic or the User MT mode.
.sp
.LP
The \fBsvc_fd_negotiate_ucred()\fR function is called by an RPC server to inform the underlying transport that the function wishes to receive \fBucreds\fR for local calls, including those over IP transports.
.sp
.LP
The \fBsvc_getcallerucred()\fR function attempts to retrieve the \fBucred_t\fR associated with the caller. The function returns \fB\fR0 when successful and \fB-1\fR when not.
.sp
.LP
When successful, the \fBsvc_getcallerucred()\fR function stores the pointer to a freshly allocated \fBucred_t\fR in the memory location pointed to by the \fIucred\fR argument if that memory location contains the null pointer. If the memory location is non-null, the function reuses the existing \fBucred_t\fR. When \fIucred\fR is no longer needed, a credential allocated by \fBsvc_getcallerucred()\fR should be freed with \fBucred_free\fR(3C).
.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of attribute types and values.
.sp

```
 258 .sp
 259 .TS
 260 box;
 261 c | c
 262 l | l .
 263 ATTRIBUTE TYPE  ATTRIBUTE VALUE
 264 _
 265 MT-Level        See below.
 266 .TE

 268 .sp
 269 .LP
 270 The \fBsvc_fd_negotiate_ucred()\fR, \fBsvc_dg_enablecache()\fR,
 271 \fBsvc_getrpccaller()\fR, and \fBsvc_getcallerucred()\fR functions are Safe in
 272 multithreaded applications. The \fBsvc_freeargs()\fR, \fBsvc_getargs()\fR, and
 273 \fBsvc_sendreply()\fR functions are Safe in multithreaded applications that use
 274 the Automatic or the User MT mode. The \fBsvc_getreq_common()\fR,
 275 \fBsvc_getreqset()\fR, and \fBsvc_getreq_poll()\fR functions are Unsafe in
 276 multithreaded applications and should be called only from the main thread.
 277 .SH SEE ALSO
 278 .sp
 279 .LP
 280 \fBrpcgen\fR(1), \fBpoll\fR(2), \fBgetpeerucred\fR(3C), \fBrpc\fR(3NSL),
 281 \fBrpc_control\fR(3NSL), \fBrpc_svc_create\fR(3NSL), \fBrpc_svc_err\fR(3NSL),
 282 \fBrpc_svc_reg\fR(3NSL), \fBselect\fR(3C), \fBucred_free\fR(3C),
 283 \fBxprt_register\fR(3NSL), \fBattributes\fR(5)
```