```
**********************************************************
    9255 Sun Mar 18 01:12:55 2018
new/usr/src/lib/smbsrv/libmlsvc/common/libmlsvc.h
1575 untangle libmlrpc ... pre2:
 Get rid of ndr_rpc_server_{info,os}
**********************************************************
______unchanged_portion_omitted_

 130 /*
 131  * Information about a server as reported by NetServerGetInfo.
 132  * The SV_PLATFORM and SV_TYPE definitions are in srvsvc.ndl.
 133  */
 134 typedef struct srvsvc_server_info {
 135         uint32_t        sv_platform_id;
 136         char            *sv_name;
 137         uint32_t        sv_version_major;
 138         uint32_t        sv_version_minor;
 139         uint32_t        sv_type;
 140         char            *sv_comment;
 141         uint32_t        sv_os;
 142 } srvsvc_server_info_t;

 144 int srvsvc_net_server_getinfo(char *, char *, srvsvc_server_info_t *);
 130 int srvsvc_net_remote_tod(char *, char *, struct timeval *, struct tm *);


 133 /*
 134  * A client_t is created while binding a client connection to hold the
 135  * context for calls made using that connection.
 136  *
 137  * Handles are RPC call specific and we use an inheritance mechanism to
 138  * ensure that each handle has a pointer to the client_t.  When the top
 139  * level (bind) handle is released, we close the connection.
 140  */
 141 typedef struct mlsvc_handle {
 142         ndr_hdid_t                      handle;
 143         ndr_client_t                    *clnt;
 159         srvsvc_server_info_t            svinfo;
 144 } mlsvc_handle_t;

 146 void ndr_rpc_init(void);
 147 void ndr_rpc_fini(void);
 148 uint32_t ndr_rpc_bind(mlsvc_handle_t *, char *, char *, char *, const char *);
 149 void ndr_rpc_unbind(mlsvc_handle_t *);
 150 int ndr_rpc_call(mlsvc_handle_t *, int, void *);
 151 void ndr_rpc_set_nonull(mlsvc_handle_t *);
 168 const srvsvc_server_info_t *ndr_rpc_server_info(mlsvc_handle_t *);
 169 uint32_t ndr_rpc_server_os(mlsvc_handle_t *);
 152 int ndr_rpc_get_ssnkey(mlsvc_handle_t *, unsigned char *, size_t);
 153 void *ndr_rpc_malloc(mlsvc_handle_t *, size_t);
 154 ndr_heap_t *ndr_rpc_get_heap(mlsvc_handle_t *);
 155 void ndr_rpc_release(mlsvc_handle_t *);
 156 boolean_t ndr_is_null_handle(mlsvc_handle_t *);
 157 boolean_t ndr_is_bind_handle(mlsvc_handle_t *);
 158 void ndr_inherit_handle(mlsvc_handle_t *, mlsvc_handle_t *);
 159 void ndr_rpc_status(mlsvc_handle_t *, int, uint32_t);

 161 /* SVCCTL service */
 162 /*
 163  * Calculate the wide-char equivalent string length required to
 164  * store a string - including the terminating null wide-char.
 165  */
 166 #define SVCCTL_WNSTRLEN(S)      ((strlen((S)) + 1) * sizeof (smb_wchar_t))

 168 /* An AVL-storable node representing each service in the SCM database. */
 169 typedef struct svcctl_svc_node {
```



```
 170         uu_avl_node_t           sn_node;
 171         char                    *sn_name;       /* Service Name (Key) */
 172         char                    *sn_fmri;       /* Display Name (FMRI) */
 173         char                    *sn_desc;       /* Description */
 174         char                    *sn_state;      /* State */
 175 } svcctl_svc_node_t;
______unchanged_portion_omitted_
```

```
**********************************************************
   36674 Sun Mar 18 01:12:55 2018
new/usr/src/lib/smbsrv/libmlsvc/common/lsar_clnt.c
1575 untangle libmlrpc ... pre2:
 Get rid of ndr_rpc_server_{info,os}
**********************************************************
______unchanged_portion_omitted_

 373 /*
 374  * Lookup a name and obtain the sid/rid.
 375  * This is a wrapper for the various lookup sid RPCs.
 376  */
 377 uint32_t
 378 lsar_lookup_names(mlsvc_handle_t *lsa_handle, char *name, smb_account_t *info)
 379 {
 380         static lsar_nameop_t ops[] = {
 381                 lsar_lookup_names3,
 382                 lsar_lookup_names2,
 383                 lsar_lookup_names1
 384         };

 386         const srvsvc_server_info_t      *svinfo;
 386         lsa_names_t     names;
 387         char            *p;
 388         uint32_t        length;
 389         uint32_t        status = NT_STATUS_INVALID_PARAMETER;
 390         int             n_op = (sizeof (ops) / sizeof (ops[0]));
 391         int             i;

 393         if (lsa_handle == NULL || name == NULL || info == NULL)
 394                 return (NT_STATUS_INVALID_PARAMETER);

 396         bzero(info, sizeof (smb_account_t));

 399         svinfo = ndr_rpc_server_info(lsa_handle);
 400         if (svinfo->sv_os == NATIVE_OS_WIN2000 &&
 401             svinfo->sv_version_major == 5 && svinfo->sv_version_minor == 0) {
 398         /*
 399          * Windows 2000 (or later) doesn't like an LSA lookup for
 403                  * Windows 2000 doesn't like an LSA lookup for
 400          * DOMAIN\Administrator.
 401          */
 402         if ((p = strchr(name, '\\')) != 0) {
 403                 ++p;

 405                 if (strcasecmp(p, "administrator") == 0)
 406                         name = p;
 407         }

 413         }

 409         length = smb_wcequiv_strlen(name);
 410         names.name[0].length = length;
 411         names.name[0].allosize = length;
 412         names.name[0].str = (unsigned char *)name;
 413         names.n_entry = 1;

 421         if (ndr_rpc_server_os(lsa_handle) == NATIVE_OS_WIN2000) {
 415         for (i = 0; i < n_op; ++i) {
 416                 ndr_rpc_set_nonull(lsa_handle);
 417                 status = (*ops[i])(lsa_handle, &names, info);

 419                 if (status != NT_STATUS_INVALID_PARAMETER)
 420                         break;
 421         }
 429         } else {
```



```
 430                 ndr_rpc_set_nonull(lsa_handle);
 431                 status = lsar_lookup_names1(lsa_handle, &names, info);
 432         }

 423         if (status == NT_STATUS_SUCCESS) {
 424                 info->a_name = lsar_get_username(name);

 426                 if (!smb_account_validate(info)) {
 427                         smb_account_free(info);
 428                         status = NT_STATUS_NO_MEMORY;
 429                 } else {
 430                         smb_account_trace(info);
 431                 }
 432         }

 434         return (status);
 435 }
______unchanged_portion_omitted_

 699 /*
 700  * Lookup a sid and obtain the domain sid and account name.
 701  * This is a wrapper for the various lookup sid RPCs.
 702  */
 703 uint32_t
 704 lsar_lookup_sids(mlsvc_handle_t *lsa_handle, smb_sid_t *sid,
 705     smb_account_t *account)
 706 {
 707         char            sidbuf[SMB_SID_STRSZ];
 708         uint32_t        status;

 710         if (lsa_handle == NULL || sid == NULL || account == NULL)
 711                 return (NT_STATUS_INVALID_PARAMETER);

 713         bzero(account, sizeof (smb_account_t));
 714         bzero(sidbuf, SMB_SID_STRSZ);
 715         smb_sid_tostr(sid, sidbuf);
 716         smb_tracef("%s", sidbuf);

 718         status = lsar_lookup_sids2(lsa_handle, (lsa_sid_t *)sid, account);
 719         if (status == RPC_NT_PROCNUM_OUT_OF_RANGE)
 729         if (ndr_rpc_server_os(lsa_handle) == NATIVE_OS_WIN2000)
 730                 status = lsar_lookup_sids2(lsa_handle, (lsa_sid_t *)sid,
 731                     account);
 732         else
 720                 status = lsar_lookup_sids1(lsa_handle, (lsa_sid_t *)sid,
 721                     account);

 723         if (status == NT_STATUS_SUCCESS) {
 724                 if (!smb_account_validate(account)) {
 725                         smb_account_free(account);
 726                         status = NT_STATUS_NO_MEMORY;
 727                 } else {
 728                         smb_account_trace(account);
 729                 }
 730         }

 732         return (status);
 733 }
______unchanged_portion_omitted_

1130 /*
1131  * lsar_lookup_priv_value
1132  *
1133  * Map a privilege name to a local unique id (LUID). Privilege names
1134  * are consistent across the network. LUIDs are machine specific.
1135  * This function provides the means to map a privilege name to the
```

```
1136  * LUID used by a remote server to represent it. The handle here is
1137  * a policy handle.
1138  */
1139 int
1140 lsar_lookup_priv_value(mlsvc_handle_t *lsa_handle, char *name,
1141     struct ms_luid *luid)
1142 {
1143         struct mslsa_LookupPrivValue    arg;
1144         int     opnum;
1145         int     rc;
1146         size_t  length;

1148         if (lsa_handle == NULL || name == NULL || luid == NULL)
1149                 return (-1);

1151         opnum = LSARPC_OPNUM_LookupPrivValue;

1153         bzero(&arg, sizeof (struct mslsa_LookupPrivValue));
1154         (void) memcpy(&arg.handle, lsa_handle, sizeof (mslsa_handle_t));

1156         length = smb_wcequiv_strlen(name);
1170         if (ndr_rpc_server_os(lsa_handle) == NATIVE_OS_WIN2000)
1157         length += sizeof (smb_wchar_t);

1159         arg.name.length = length;
1160         arg.name.allosize = length;
1161         arg.name.str = (unsigned char *)name;

1163         rc = ndr_rpc_call(lsa_handle, opnum, &arg);
1164         if (rc == 0) {
1165                 if (arg.status != 0)
1166                         rc = -1;
1167                 else
1168                         (void) memcpy(luid, &arg.luid, sizeof (struct ms_luid));
1169         }

1171         ndr_rpc_release(lsa_handle);
1172         return (rc);
1173 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   13629 Sun Mar 18 01:12:55 2018
new/usr/src/lib/smbsrv/libmlsvc/common/mlsvc_client.c
1575 untangle libmlrpc ... pre2:
 Get rid of ndr_rpc_server_{info,os}
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */

  22 /*
  23  * Copyright (c) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
  24  * Copyright 2015 Nexenta Systems, Inc.  All rights reserved.
  25  */

  27 /*
  28  * Client NDR RPC interface.
  29  */

  31 #include <sys/types.h>
  32 #include <sys/errno.h>
  33 #include <sys/fcntl.h>
  34 #include <time.h>
  35 #include <strings.h>
  36 #include <assert.h>
  37 #include <errno.h>
  38 #include <thread.h>
  39 #include <syslog.h>
  40 #include <synch.h>

  42 #include <netsmb/smbfs_api.h>
  43 #include <smbsrv/libsmb.h>
  44 #include <smbsrv/libsmbns.h>
  45 #include <smbsrv/libmlrpc.h>
  46 #include <smbsrv/libmlsvc.h>
  47 #include <smbsrv/ndl/srvsvc.ndl>
  48 #include <libsmbrdr.h>
  49 #include <mlsvc.h>

  51 static int ndr_xa_init(ndr_client_t *, ndr_xa_t *);
  52 static int ndr_xa_exchange(ndr_client_t *, ndr_xa_t *);
  53 static int ndr_xa_read(ndr_client_t *, ndr_xa_t *);
  54 static void ndr_xa_preserve(ndr_client_t *, ndr_xa_t *);
  55 static void ndr_xa_destruct(ndr_client_t *, ndr_xa_t *);
  56 static void ndr_xa_release(ndr_client_t *);


  59 /*
  60  * This call must be made to initialize an RPC client structure and bind
```



```
  61  * to the remote service before any RPCs can be exchanged with that service.
  62  *
  63  * The mlsvc_handle_t is a wrapper that is used to associate an RPC handle
  64  * with the client context for an instance of the interface.  The handle
  65  * is zeroed to ensure that it doesn't look like a valid handle -
  66  * handle content is provided by the remove service.
  67  *
  68  * The client points to this top-level handle so that we know when to
  69  * unbind and teardown the connection.  As each handle is initialized it
  70  * will inherit a reference to the client context.
  71  *
  72  * Returns 0 or an NT_STATUS:
  73  *      NT_STATUS_BAD_NETWORK_PATH      (get server addr)
  74  *      NT_STATUS_NETWORK_ACCESS_DENIED (connect, auth)
  75  *      NT_STATUS_BAD_NETWORK_NAME      (tcon, open)
  76  *      NT_STATUS_ACCESS_DENIED         (open pipe)
  77  *      NT_STATUS_INVALID_PARAMETER     (rpc bind)
  78  *
  79  *      NT_STATUS_INTERNAL_ERROR        (bad args etc)
  80  *      NT_STATUS_NO_MEMORY
  81  */
  82 DWORD
  83 ndr_rpc_bind(mlsvc_handle_t *handle, char *server, char *domain,
  84     char *username, const char *service)
  85 {
  86         struct smb_ctx          *ctx = NULL;
  87         ndr_client_t            *clnt = NULL;
  88         ndr_service_t           *svc;
  89         srvsvc_server_info_t    svinfo;
  89         DWORD                   status;
  90         int                     fd = -1;
  91         int                     rc;

  93         if (handle == NULL || server == NULL || server[0] == '\0' ||
  94             domain == NULL || username == NULL)
  95                 return (NT_STATUS_INTERNAL_ERROR);

  97         /* In case the service was not registered... */
  98         if ((svc = ndr_svc_lookup_name(service)) == NULL)
  99                 return (NT_STATUS_INTERNAL_ERROR);

 101         /*
 103          * Set the default based on the assumption that most
 104          * servers will be Windows 2000 or later.  This used to
 105          * try to get the actual server version, but that RPC
 106          * is not necessarily allowed anymore, so don't bother.
 107          */
 108         bzero(&svinfo, sizeof (srvsvc_server_info_t));
 109         svinfo.sv_platform_id = SV_PLATFORM_ID_NT;
 110         svinfo.sv_version_major = 5;
 111         svinfo.sv_version_minor = 0;
 112         svinfo.sv_type = SV_TYPE_DEFAULT;
 113         svinfo.sv_os = NATIVE_OS_WIN2000;

 115         /*
 102          * Some callers pass this when they want a NULL session.
 103          * Todo: have callers pass an empty string for that.
 104          */
 105         if (strcmp(username, MLSVC_ANON_USER) == 0)
 106                 username = "";

 108         /*
 109          * Setup smbfs library handle, authenticate, connect to
 110          * the IPC$ share.  This will reuse an existing connection
 111          * if the driver already has one for this combination of
 112          * server, user, domain.  It may return any of:
```

```
 113          *      NT_STATUS_BAD_NETWORK_PATH      (get server addr)
 114          *      NT_STATUS_NETWORK_ACCESS_DENIED (connect, auth)
 115          *      NT_STATUS_BAD_NETWORK_NAME      (tcon)
 116          */
 117         status = smbrdr_ctx_new(&ctx, server, domain, username);
 118         if (status != NT_STATUS_SUCCESS) {
 119                 syslog(LOG_ERR, "ndr_rpc_bind: smbrdr_ctx_new"
 120                     "(Srv=%s Dom=%s User=%s), %s (0x%x)",
 121                     server, domain, username,
 122                     xlate_nt_status(status), status);
 123                 /* Tell the DC Locator this DC failed. */
 124                 smb_ddiscover_bad_dc(server);
 125                 goto errout;
 126         }

 128         /*
 129          * Open the named pipe.
 130          */
 131         fd = smb_fh_open(ctx, svc->endpoint, O_RDWR);
 132         if (fd < 0) {
 133                 rc = errno;
 134                 syslog(LOG_DEBUG, "ndr_rpc_bind: "
 135                     "smb_fh_open (%s) err=%d",
 136                     svc->endpoint, rc);
 137                 switch (rc) {
 138                 case EACCES:
 139                         status = NT_STATUS_ACCESS_DENIED;
 140                         break;
 141                 default:
 142                         status = NT_STATUS_BAD_NETWORK_NAME;
 143                         break;
 144                 }
 145                 goto errout;
 146         }

 148         /*
 149          * Setup the RPC client handle.
 150          */
 151         if ((clnt = malloc(sizeof (ndr_client_t))) == NULL) {
 152                 status = NT_STATUS_NO_MEMORY;
 153                 goto errout;
 154         }
 155         bzero(clnt, sizeof (ndr_client_t));

 157         clnt->handle = &handle->handle;
 158         clnt->xa_init = ndr_xa_init;
 159         clnt->xa_exchange = ndr_xa_exchange;
 160         clnt->xa_read = ndr_xa_read;
 161         clnt->xa_preserve = ndr_xa_preserve;
 162         clnt->xa_destruct = ndr_xa_destruct;
 163         clnt->xa_release = ndr_xa_release;
 164         clnt->xa_private = ctx;
 165         clnt->xa_fd = fd;

 167         ndr_svc_binding_pool_init(&clnt->binding_list,
 168             clnt->binding_pool, NDR_N_BINDING_POOL);

 170         if ((clnt->heap = ndr_heap_create()) == NULL) {
 171                 status = NT_STATUS_NO_MEMORY;
 172                 goto errout;
 173         }

 175         /*
 176          * Fill in the caller's handle.
 177          */
 178         bzero(&handle->handle, sizeof (ndr_hdid_t));
```



```
 179         handle->clnt = clnt;
 194         bcopy(&svinfo, &handle->svinfo, sizeof (srvsvc_server_info_t));

 181         /*
 182          * Do the OtW RPC bind.
 183          */
 184         rc = ndr_clnt_bind(clnt, service, &clnt->binding);
 185         switch (rc) {
 186         case NDR_DRC_FAULT_OUT_OF_MEMORY:
 187                 status = NT_STATUS_NO_MEMORY;
 188                 break;
 189         case NDR_DRC_FAULT_API_SERVICE_INVALID: /* not registered */
 190                 status = NT_STATUS_INTERNAL_ERROR;
 191                 break;
 192         default:
 193                 if (NDR_DRC_IS_FAULT(rc)) {
 194                         status = NT_STATUS_INVALID_PARAMETER;
 195                         break;
 196                 }
 197                 /* FALLTHROUGH */
 198         case NDR_DRC_OK:
 199                 return (NT_STATUS_SUCCESS);
 200         }

 202         syslog(LOG_DEBUG, "ndr_rpc_bind: "
 203             "ndr_clnt_bind, %s (0x%x)",
 204             xlate_nt_status(status), status);

 206 errout:
 207         handle->clnt = NULL;
 208         if (clnt != NULL) {
 209                 ndr_heap_destroy(clnt->heap);
 210                 free(clnt);
 211         }
 212         if (ctx != NULL) {
 213                 if (fd != -1)
 214                         (void) smb_fh_close(fd);
 215                 smbrdr_ctx_free(ctx);
 216         }

 218         return (status);
 219 }
```

unchanged_portion_omitted_

```
 290 /*
 306  * Return a reference to the server info.
 307  */
 308 const srvsvc_server_info_t *
 309 ndr_rpc_server_info(mlsvc_handle_t *handle)
 310 {
 311         return (&handle->svinfo);
 312 }

 314 /*
 315  * Return the RPC server OS level.
 316  */
 317 uint32_t
 318 ndr_rpc_server_os(mlsvc_handle_t *handle)
 319 {
 320         return (handle->svinfo.sv_os);
 321 }

 323 /*
 291  * Get the session key from a bound RPC client handle.
 292  *
 293  * The key returned is the 16-byte "user session key"
```

```
 294  * established by the underlying authentication protocol
 295  * (either Kerberos or NTLM).  This key is needed for
 296  * SAM RPC calls such as SamrSetInformationUser, etc.
 297  * See [MS-SAMR] sections: 2.2.3.3, 2.2.7.21, 2.2.7.25.
 298  *
 299  * Returns zero (success) or an errno.
 300  */
 301 int
 302 ndr_rpc_get_ssnkey(mlsvc_handle_t *handle,
 303         unsigned char *ssn_key, size_t len)
 304 {
 305         ndr_client_t *clnt = handle->clnt;
 306         int rc;

 308         if (clnt == NULL)
 309                 return (EINVAL);

 311         rc = smb_fh_getssnkey(clnt->xa_fd, ssn_key, len);
 312         return (rc);
 313 }
______unchanged_portion_omitted_

 384 /*
 385  * Pass the client reference from parent to child.
 386  */
 387 void
 388 ndr_inherit_handle(mlsvc_handle_t *child, mlsvc_handle_t *parent)
 389 {
 390         child->clnt = parent->clnt;
 424         bcopy(&parent->svinfo, &child->svinfo, sizeof (srvsvc_server_info_t));
 391 }
______unchanged_portion_omitted_
```

```
**********************************************************
   16191 Sun Mar 18 01:12:55 2018
new/usr/src/lib/smbsrv/libmlsvc/common/netr_auth.c
1575 untangle libmlrpc ... pre2:
 Get rid of ndr_rpc_server_{info,os}
**********************************************************
______unchanged_portion_omitted_

194 uint32_t netr_server_auth2_flags =
195     NETR_NEGOTIATE_BASE_FLAGS |
196     NETR_NEGOTIATE_STRONGKEY_FLAG;

198 /*
199  * netr_server_authenticate2
200  */
201 static int
202 netr_server_authenticate2(mlsvc_handle_t *netr_handle, netr_info_t *netr_info)
203 {
204         struct netr_ServerAuthenticate2 arg;
205         /* sizeof netr_info->hostname, + 1 for the '$' */
206         char account_name[(NETBIOS_NAME_SZ * 2) + 1];
207         int opnum;
208         int rc;

210         bzero(&arg, sizeof (struct netr_ServerAuthenticate2));
211         opnum = NETR_OPNUM_ServerAuthenticate2;

213         (void) snprintf(account_name, sizeof (account_name), "%s$",
214             netr_info->hostname);

216         smb_tracef("server=[%s] account_name=[%s] hostname=[%s]\n",
217             netr_info->server, account_name, netr_info->hostname);

219         arg.servername = (unsigned char *)netr_info->server;
220         arg.account_name = (unsigned char *)account_name;
221         arg.account_type = NETR_WKSTA_TRUST_ACCOUNT_TYPE;
222         arg.hostname = (unsigned char *)netr_info->hostname;
223         arg.negotiate_flags = netr_server_auth2_flags;
219         arg.negotiate_flags = NETR_NEGOTIATE_BASE_FLAGS;

225         if (arg.negotiate_flags & NETR_NEGOTIATE_STRONGKEY_FLAG) {
221         if (ndr_rpc_server_os(netr_handle) == NATIVE_OS_WIN2000) {
222                 arg.negotiate_flags |= NETR_NEGOTIATE_STRONGKEY_FLAG;
226                 if (netr_gen_skey128(netr_info) != SMBAUTH_SUCCESS)
227                         return (-1);
228         } else {
229                 if (netr_gen_skey64(netr_info) != SMBAUTH_SUCCESS)
230                         return (-1);
231         }

233         if (netr_gen_credentials(netr_info->session_key.key,
234             &netr_info->client_challenge, 0,
235             &netr_info->client_credential) != SMBAUTH_SUCCESS) {
236                 return (-1);
237         }

239         if (netr_gen_credentials(netr_info->session_key.key,
240             &netr_info->server_challenge, 0,
241             &netr_info->server_credential) != SMBAUTH_SUCCESS) {
242                 return (-1);
243         }

245         (void) memcpy(&arg.client_credential, &netr_info->client_credential,
246             sizeof (struct netr_credential));

248         if (ndr_rpc_call(netr_handle, opnum, &arg) != 0)
```



```
249                 return (-1);

251         if (arg.status != 0) {
252                 ndr_rpc_status(netr_handle, opnum, arg.status);
253                 ndr_rpc_release(netr_handle);
254                 return (-1);
255         }

257         rc = memcmp(&netr_info->server_credential, &arg.server_credential,
258             sizeof (struct netr_credential));

260         ndr_rpc_release(netr_handle);
261         return (rc);
262 }
______unchanged_portion_omitted_
```

```
**********************************************************
   13246 Sun Mar 18 01:12:55 2018
new/usr/src/lib/smbsrv/libmlsvc/common/srvsvc_clnt.c
1575 untangle libmlrpc ... pre2:
 Get rid of ndr_rpc_server_{info,os}
**********************************************************
______unchanged_portion_omitted_

 352 /*
 353  * Windows 95+ and Windows NT4.0 both report the version as 4.0.
 354  * Windows 2000+ reports the version as 5.x.
 355  */
 356 int
 357 srvsvc_net_server_getinfo(char *server, char *domain,
 358     srvsvc_server_info_t *svinfo)
 359 {
 360         mlsvc_handle_t handle;
 361         struct mslm_NetServerGetInfo arg;
 362         struct mslm_SERVER_INFO_101 *sv101;
 363         int len, opnum, rc;
 364         char user[SMB_USERNAME_MAXLEN];

 366         smb_ipc_get_user(user, SMB_USERNAME_MAXLEN);

 368         if (srvsvc_open(server, domain, user, &handle) != 0)
 369                 return (-1);

 371         opnum = SRVSVC_OPNUM_NetServerGetInfo;
 372         bzero(&arg, sizeof (arg));

 374         len = strlen(server) + 4;
 375         arg.servername = ndr_rpc_malloc(&handle, len);
 376         if (arg.servername == NULL)
 377                 return (-1);

 379         (void) snprintf((char *)arg.servername, len, "\\\\%s", server);
 380         arg.level = 101;

 382         rc = ndr_rpc_call(&handle, opnum, &arg);
 383         if ((rc != 0) || (arg.status != 0)) {
 384                 srvsvc_close(&handle);
 385                 return (-1);
 386         }

 388         sv101 = arg.result.bufptr.bufptr101;

 390         bzero(svinfo, sizeof (srvsvc_server_info_t));
 391         svinfo->sv_platform_id = sv101->sv101_platform_id;
 392         svinfo->sv_version_major = sv101->sv101_version_major;
 393         svinfo->sv_version_minor = sv101->sv101_version_minor;
 394         svinfo->sv_type = sv101->sv101_type;
 395         if (sv101->sv101_name)
 396                 svinfo->sv_name = strdup((char *)sv101->sv101_name);
 397         if (sv101->sv101_comment)
 398                 svinfo->sv_comment = strdup((char *)sv101->sv101_comment);

 400         if (svinfo->sv_type & SV_TYPE_WFW)
 401                 svinfo->sv_os = NATIVE_OS_WIN95;
 402         if (svinfo->sv_type & SV_TYPE_WINDOWS)
 403                 svinfo->sv_os = NATIVE_OS_WIN95;
 404         if ((svinfo->sv_type & SV_TYPE_NT) ||
 405             (svinfo->sv_type & SV_TYPE_SERVER_NT))
 406                 svinfo->sv_os = NATIVE_OS_WINNT;
 407         if (svinfo->sv_version_major > 4)
 408                 svinfo->sv_os = NATIVE_OS_WIN2000;
```



```
 410         srvsvc_close(&handle);
 411         return (0);
 412 }

 414 /*
 353  * Compare the time here with the remote time on the server
 354  * and report clock skew.
 355  */
 356 void
 357 srvsvc_timecheck(char *server, char *domain)
 358 {
 359         char                    hostname[MAXHOSTNAMELEN];
 360         struct timeval          dc_tv;
 361         struct tm               dc_tm;
 362         struct tm               *tm;
 363         time_t                  tnow;
 364         time_t                  tdiff;
 365         int                     priority;

 367         if (srvsvc_net_remote_tod(server, domain, &dc_tv, &dc_tm) < 0) {
 368                 syslog(LOG_DEBUG, "srvsvc_net_remote_tod failed");
 369                 return;
 370         }

 372         tnow = time(NULL);

 374         if (tnow > dc_tv.tv_sec)
 375                 tdiff = (tnow - dc_tv.tv_sec) / SECSPERMIN;
 376         else
 377                 tdiff = (dc_tv.tv_sec - tnow) / SECSPERMIN;

 379         if (tdiff != 0) {
 380                 (void) strlcpy(hostname, "localhost", MAXHOSTNAMELEN);
 381                 (void) gethostname(hostname, MAXHOSTNAMELEN);

 383                 priority = (tdiff > 2) ? LOG_NOTICE : LOG_DEBUG;
 384                 syslog(priority, "DC [%s] clock skew detected: %u minutes",
 385                     server, tdiff);

 387                 tm = gmtime(&dc_tv.tv_sec);
 388                 syslog(priority, "%-8s  UTC: %s", server, asctime(tm));
 389                 tm = gmtime(&tnow);
 390                 syslog(priority, "%-8s  UTC: %s", hostname, asctime(tm));
 391         }
 392 }
______unchanged_portion_omitted_

 441 /*
 442  * This is a client side routine for NetRemoteTOD, which gets the time
 443  * and date from a remote system. The time information is returned in
 444  * the timeval and tm.
 445  *
 446  * typedef struct _TIME_OF_DAY_INFO {
 447  *      DWORD tod_elapsedt;  // seconds since 00:00:00 January 1 1970 GMT
 448  *      DWORD tod_msecs;     // arbitrary milliseconds (since reset)
 449  *      DWORD tod_hours;     // current hour [0-23]
 450  *      DWORD tod_mins;      // current minute [0-59]
 451  *      DWORD tod_secs;      // current second [0-59]
 452  *      DWORD tod_hunds;     // current hundredth (0.01) second [0-99]
 453  *      LONG tod_timezone;   // time zone of the server
 454  *      DWORD tod_tinterval; // clock tick time interval
 455  *      DWORD tod_day;       // day of the month [1-31]
 456  *      DWORD tod_month;     // month of the year [1-12]
 457  *      DWORD tod_year;      // current year
 458  *      DWORD tod_weekday;   // day of the week since sunday [0-6]
 459  * } TIME_OF_DAY_INFO;
```

```
 460  *
 461  * The time zone of the server is calculated in minutes from Greenwich
 462  * Mean Time (GMT). For time zones west of Greenwich, the value is
 463  * positive; for time zones east of Greenwich, the value is negative.
 464  * A value of -1 indicates that the time zone is undefined.
 465  *
 466  * The clock tick value represents a resolution of one ten-thousandth
 467  * (0.0001) second.
 468  */
 469 int
 470 srvsvc_net_remote_tod(char *server, char *domain, struct timeval *tv,
 471     struct tm *tm)
 472 {
 473         struct mslm_NetRemoteTOD        arg;
 474         struct mslm_TIME_OF_DAY_INFO    *tod;
 475         mlsvc_handle_t                  handle;
 476         int                             rc;
 477         int                             opnum;
 478         int                             len;
 479         char                            user[SMB_USERNAME_MAXLEN];

 481         smb_ipc_get_user(user, SMB_USERNAME_MAXLEN);

 483         rc = srvsvc_open(server, domain, user, &handle);
 484         if (rc != 0)
 485                 return (-1);

 487         opnum = SRVSVC_OPNUM_NetRemoteTOD;
 488         bzero(&arg, sizeof (struct mslm_NetRemoteTOD));

 490         len = strlen(server) + 4;
 491         arg.servername = ndr_rpc_malloc(&handle, len);
 492         if (arg.servername == NULL) {
 493                 srvsvc_close(&handle);
 494                 return (-1);
 495         }

 497         (void) snprintf((char *)arg.servername, len, "\\\\%s", server);

 499         rc = ndr_rpc_call(&handle, opnum, &arg);
 500         if ((rc != 0) || (arg.status != 0)) {
 501                 srvsvc_close(&handle);
 502                 return (-1);
 503         }

 505         /*
 506          * We're assigning milliseconds to microseconds
 507          * here but the value's not really relevant.
 508          */
 509         tod = arg.bufptr;

 511         if (tv) {
 512                 tv->tv_sec = tod->tod_elapsedt;
 513                 tv->tv_usec = tod->tod_msecs;
 514         }

 516         if (tm) {
 517                 tm->tm_sec = tod->tod_secs;
 518                 tm->tm_min = tod->tod_mins;
 519                 tm->tm_hour = tod->tod_hours;
 520                 tm->tm_mday = tod->tod_day;
 521                 tm->tm_mon = tod->tod_month - 1;
 522                 tm->tm_year = tod->tod_year - 1900;
 523                 tm->tm_wday = tod->tod_weekday;
 524         }
```



```
 526         srvsvc_close(&handle);
 527         return (0);
 590 }

 592 void
 593 srvsvc_net_test(char *server, char *domain, char *netname)
 594 {
 595         smb_domainex_t di;
 596         srvsvc_server_info_t svinfo;

 598         (void) smb_tracef("%s %s %s", server, domain, netname);

 600         if (smb_domain_getinfo(&di)) {
 601                 server = di.d_dci.dc_name;
 602                 domain = di.d_primary.di_nbname;
 603         }

 605         if (srvsvc_net_server_getinfo(server, domain, &svinfo) == 0) {
 606                 smb_tracef("NetServerGetInfo: %s %s (%d.%d) id=%d type=0x%08x",
 607                     svinfo.sv_name ? svinfo.sv_name : "NULL",
 608                     svinfo.sv_comment ? svinfo.sv_comment : "NULL",
 609                     svinfo.sv_version_major, svinfo.sv_version_minor,
 610                     svinfo.sv_platform_id, svinfo.sv_type);

 612                 free(svinfo.sv_name);
 613                 free(svinfo.sv_comment);
 614         }

 616         (void) srvsvc_net_share_get_info(server, domain, netname);
 617 #if 0
 618         /*
 619          * The NetSessionEnum server-side definition was updated.
 620          * Disabled until the client-side has been updated.
 621          */
 622         (void) srvsvc_net_session_enum(server, domain, netname);
 623 #endif
 624         (void) srvsvc_net_connect_enum(server, domain, netname, 0);
 625         (void) srvsvc_net_connect_enum(server, domain, netname, 1);
 528 }
```

_____***unchanged_portion_omitted_***