```
**********************************************************
    2811 Sun Mar 18 01:12:53 2018
new/usr/src/lib/smbsrv/libmlsvc/common/mlsvc.h
1575 untangle libmlrpc ... pre1:
 Move srvsvc_timecheck where it belongs
**********************************************************
```

```
   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  */
  21 /*
  22  * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
  23  * Copyright 2015 Nexenta Systems, Inc.  All rights reserved.
  24  */

  26 #ifndef _SMBSRV_MLSVC_H
  27 #define _SMBSRV_MLSVC_H

  29 #include <smbsrv/smb_share.h>
  30 #include <smbsrv/ndl/netlogon.ndl>

  32 #ifdef __cplusplus
  33 extern "C" {
  34 #endif

  36 struct netr_info;

  38 int smb_dclocator_init(void);
  39 void smbrdr_initialize(void);
  40 void dssetup_initialize(void);
  41 void srvsvc_initialize(void);
  42 void wkssvc_initialize(void);
  43 void lsarpc_initialize(void);
  44 void logr_initialize(void);
  45 void netr_initialize(void);
  46 void samr_initialize(void);
  47 void svcctl_initialize(void);
  48 void winreg_initialize(void);
  49 int srvsvc_gettime(unsigned long *);
  49 void msgsvcsend_initialize(void);
  50 void spoolss_initialize(void);
  51 void netdfs_initialize(void);

  53 void logr_finalize(void);
  54 void svcctl_finalize(void);
  55 void spoolss_finalize(void);
  56 void netdfs_finalize(void);

  58 /* netr_auth.c */
  59 DWORD netr_open(char *, char *, mlsvc_handle_t *);
```



```
  60 int netr_close(mlsvc_handle_t *);
  61 DWORD netlogon_auth(char *, mlsvc_handle_t *, DWORD);
  62 int netr_setup_authenticator(struct netr_info *, struct netr_authenticator *,
  63     struct netr_authenticator *);
  64 DWORD netr_validate_chain(struct netr_info *, struct netr_authenticator *);

  66 int srvsvc_gettime(unsigned long *);
  67 void srvsvc_timecheck(char *, char *);
  67 void ndr_srvsvc_timecheck(char *, char *);

  69 /* Generic functions to get/set windows Security Descriptors */
  70 uint32_t srvsvc_sd_get(smb_share_t *, uint8_t *, uint32_t *);
  71 uint32_t srvsvc_sd_set(smb_share_t *, uint8_t *);

  73 uint32_t smb_logon_init(void);
  74 void smb_logon_fini(void);

  76 /* Locking for process-wide settings (i.e. privileges) */
  77 void smb_proc_initsem(void);    /* init (or re-init in child) */
  78 int  smb_proc_takesem(void);    /* parent before */
  79 void smb_proc_givesem(void);    /* parent after */

  81 /* Quota */
  82 void smb_quota_init(void);
  83 void smb_quota_fini(void);
  84 void smb_quota_add_fs(const char *);
  85 void smb_quota_remove_fs(const char *);

  87 uint32_t smb_ddiscover_main(char *, smb_domainex_t *);

  89 #ifdef __cplusplus
  90 }
_____unchanged_portion_omitted_
```

```
**********************************************************
   14535 Sun Mar 18 01:12:53 2018
new/usr/src/lib/smbsrv/libmlsvc/common/mlsvc_client.c
1575 untangle libmlrpc ... pre1:
 Move srvsvc_timecheck where it belongs
**********************************************************
```

```
   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 <sys/tzfile.h>
  34 #include <time.h>
  35 #include <strings.h>
  36 #include <assert.h>
  37 #include <errno.h>
  38 #include <thread.h>
  40 #include <unistd.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;
  90         DWORD                   status;
  91         int                     fd = -1;
  92         int                     rc;

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

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

 102         /*
 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         /*
 116          * Some callers pass this when they want a NULL session.
 117          * Todo: have callers pass an empty string for that.
 118          */
 119         if (strcmp(username, MLSVC_ANON_USER) == 0)
 120                 username = "";

 122         /*
 123          * Setup smbfs library handle, authenticate, connect to
 124          * the IPC$ share.  This will reuse an existing connection
```

```
125          * if the driver already has one for this combination of
126          * server, user, domain.  It may return any of:
127          *      NT_STATUS_BAD_NETWORK_PATH      (get server addr)
128          *      NT_STATUS_NETWORK_ACCESS_DENIED (connect, auth)
129          *      NT_STATUS_BAD_NETWORK_NAME      (tcon)
130          */
131         status = smbrdr_ctx_new(&ctx, server, domain, username);
132         if (status != NT_STATUS_SUCCESS) {
133                 syslog(LOG_ERR, "ndr_rpc_bind: smbrdr_ctx_new"
134                     "(Srv=%s Dom=%s User=%s), %s (0x%x)",
135                     server, domain, username,
136                     xlate_nt_status(status), status);
137                 /* Tell the DC Locator this DC failed. */
138                 smb_ddiscover_bad_dc(server);
139                 goto errout;
140         }
141
142         /*
143          * Open the named pipe.
144          */
145         fd = smb_fh_open(ctx, svc->endpoint, O_RDWR);
146         if (fd < 0) {
147                 rc = errno;
148                 syslog(LOG_DEBUG, "ndr_rpc_bind: "
149                     "smb_fh_open (%s) err=%d",
150                     svc->endpoint, rc);
151                 switch (rc) {
152                 case EACCES:
153                         status = NT_STATUS_ACCESS_DENIED;
154                         break;
155                 default:
156                         status = NT_STATUS_BAD_NETWORK_NAME;
157                         break;
158                 }
159                 goto errout;
160         }
161
162         /*
163          * Setup the RPC client handle.
164          */
165         if ((clnt = malloc(sizeof (ndr_client_t))) == NULL) {
166                 status = NT_STATUS_NO_MEMORY;
167                 goto errout;
168         }
169         bzero(clnt, sizeof (ndr_client_t));
170
171         clnt->handle = &handle->handle;
172         clnt->xa_init = ndr_xa_init;
173         clnt->xa_exchange = ndr_xa_exchange;
174         clnt->xa_read = ndr_xa_read;
175         clnt->xa_preserve = ndr_xa_preserve;
176         clnt->xa_destruct = ndr_xa_destruct;
177         clnt->xa_release = ndr_xa_release;
178         clnt->xa_private = ctx;
179         clnt->xa_fd = fd;
180
181         ndr_svc_binding_pool_init(&clnt->binding_list,
182             clnt->binding_pool, NDR_N_BINDING_POOL);
183
184         if ((clnt->heap = ndr_heap_create()) == NULL) {
185                 status = NT_STATUS_NO_MEMORY;
186                 goto errout;
187         }
188
189         /*
190          * Fill in the caller's handle.
```



```
191          */
192         bzero(&handle->handle, sizeof (ndr_hdid_t));
193         handle->clnt = clnt;
194         bcopy(&svinfo, &handle->svinfo, sizeof (srvsvc_server_info_t));
195
196         /*
197          * Do the OtW RPC bind.
198          */
199         rc = ndr_clnt_bind(clnt, service, &clnt->binding);
200         switch (rc) {
201         case NDR_DRC_FAULT_OUT_OF_MEMORY:
202                 status = NT_STATUS_NO_MEMORY;
203                 break;
204         case NDR_DRC_FAULT_API_SERVICE_INVALID: /* not registered */
205                 status = NT_STATUS_INTERNAL_ERROR;
206                 break;
207         default:
208                 if (NDR_DRC_IS_FAULT(rc)) {
209                         status = NT_STATUS_INVALID_PARAMETER;
210                         break;
211                 }
212                 /* FALLTHROUGH */
213         case NDR_DRC_OK:
214                 return (NT_STATUS_SUCCESS);
215         }
216
217         syslog(LOG_DEBUG, "ndr_rpc_bind: "
218             "ndr_clnt_bind, %s (0x%x)",
219             xlate_nt_status(status), status);
220
221 errout:
222         handle->clnt = NULL;
223         if (clnt != NULL) {
224                 ndr_heap_destroy(clnt->heap);
225                 free(clnt);
226         }
227         if (ctx != NULL) {
228                 if (fd != -1)
229                         (void) smb_fh_close(fd);
230                 smbrdr_ctx_free(ctx);
231         }
232
233         return (status);
234 }
```

**_____unchanged_portion_omitted_**

```
608 /*
609  * Compare the time here with the remote time on the server
610  * and report clock skew.
611  */
612 void
613 ndr_srvsvc_timecheck(char *server, char *domain)
614 {
615         char                    hostname[MAXHOSTNAMELEN];
616         struct timeval          dc_tv;
617         struct tm               dc_tm;
618         struct tm               *tm;
619         time_t                  tnow;
620         time_t                  tdiff;
621         int                     priority;
622
623         if (srvsvc_net_remote_tod(server, domain, &dc_tv, &dc_tm) < 0) {
624                 syslog(LOG_DEBUG, "srvsvc_net_remote_tod failed");
625                 return;
626         }
```

```
628         tnow = time(NULL);

630         if (tnow > dc_tv.tv_sec)
631                 tdiff = (tnow - dc_tv.tv_sec) / SECSPERMIN;
632         else
633                 tdiff = (dc_tv.tv_sec - tnow) / SECSPERMIN;

635         if (tdiff != 0) {
636                 (void) strlcpy(hostname, "localhost", MAXHOSTNAMELEN);
637                 (void) gethostname(hostname, MAXHOSTNAMELEN);

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

643                 tm = gmtime(&dc_tv.tv_sec);
644                 syslog(priority, "%-8s  UTC: %s", server, asctime(tm));
645                 tm = gmtime(&tnow);
646                 syslog(priority, "%-8s  UTC: %s", hostname, asctime(tm));
647         }
648 }
```

***********************************************************
***2780 Sun Mar 18 01:12:53 2018***
***new/usr/src/lib/smbsrv/libmlsvc/common/mlsvc_init.c***
***1575 untangle libmlrpc ... pre1:***
***Move srvsvc_timecheck where it belongs***
***********************************************************
_____***unchanged_portion_omitted_***

```
 95 /*ARGSUSED*/
 96 static void *
 97 mlsvc_timecheck(void *arg)
 98 {
 99         smb_domainex_t di;

101         for (;;) {
102                 (void) sleep(MLSVC_TIMECHECK_INTERVAL);

104                 if (smb_config_get_secmode() != SMB_SECMODE_DOMAIN)
105                         continue;

107                 /* Avoid interfering with DC discovery. */
108                 if (smb_ddiscover_wait() != 0)
109                         continue;

111                 if (!smb_domain_getinfo(&di))
112                         continue;

114                 srvsvc_timecheck(di.d_dci.dc_name,
114                 ndr_srvsvc_timecheck(di.d_dci.dc_name,
115                     di.d_primary.di_nbname);
116         }

118         /*NOTREACHED*/
119         return (NULL);
120 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   15998 Sun Mar 18 01:12:53 2018
new/usr/src/lib/smbsrv/libmlsvc/common/srvsvc_clnt.c
1575 untangle libmlrpc ... pre1:
 Move srvsvc_timecheck where it belongs
**********************************************************
```

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

/*
 * Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2015 Nexenta Systems, Inc.  All rights reserved.
 */

/*
 * Server Service (srvsvc) client side RPC library interface. The
 * srvsvc interface allows a client to query a server for information
 * on shares, sessions, connections and files on the server. Some
 * functions are available via anonymous IPC while others require
 * administrator privilege. Also, some functions return NT status
 * values while others return Win32 errors codes.
 */

#include <sys/errno.h>
#include <sys/tzfile.h>
#include <stdio.h>
#include <time.h>
#include <strings.h>
#include <unistd.h>

#include <smbsrv/libsmb.h>
#include <smbsrv/libmlsvc.h>
#include <smbsrv/smbinfo.h>
#include <smbsrv/ndl/srvsvc.ndl>

/*
 * Information level for NetShareGetInfo.
 */
DWORD srvsvc_info_level = 1;

/*
 * Bind to the the SRVSVC.
 *
 * If username argument is NULL, an anonymous connection will be established.
 * Otherwise, an authenticated connection will be established.
 */
static int
srvsvc_open(char *server, char *domain, char *username, mlsvc_handle_t *handle)
```



```
{
        smb_domainex_t di;

        if (server == NULL || domain == NULL) {
                if (!smb_domain_getinfo(&di))
                        return (-1);

                server = di.d_dci.dc_name;
                domain = di.d_primary.di_nbname;
        }

        if (username == NULL)
                username = MLSVC_ANON_USER;

        if (ndr_rpc_bind(handle, server, domain, username, "SRVSVC") != 0)
                return (-1);

        return (0);
}
```

*unchanged_portion_omitted_*

```
/*
 * Compare the time here with the remote time on the server
 * and report clock skew.
 */
void
srvsvc_timecheck(char *server, char *domain)
{
        char                    hostname[MAXHOSTNAMELEN];
        struct timeval          dc_tv;
        struct tm               dc_tm;
        struct tm               *tm;
        time_t                  tnow;
        time_t                  tdiff;
        int                     priority;

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

        tnow = time(NULL);

        if (tnow > dc_tv.tv_sec)
                tdiff = (tnow - dc_tv.tv_sec) / SECSPERMIN;
        else
                tdiff = (dc_tv.tv_sec - tnow) / SECSPERMIN;

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

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

                tm = gmtime(&dc_tv.tv_sec);
                syslog(priority, "%-8s  UTC: %s", server, asctime(tm));
                tm = gmtime(&tnow);
                syslog(priority, "%-8s  UTC: %s", hostname, asctime(tm));
        }
}

/*
 * Synchronize the local system clock with the domain controller.
 */
```

```
 459 void
 460 srvsvc_timesync(void)
 461 {
 462         smb_domainex_t di;
 463         struct timeval tv;
 464         struct tm tm;
 465         time_t tsecs;

 467         if (!smb_domain_getinfo(&di))
 468                 return;

 470         if (srvsvc_net_remote_tod(di.d_dci.dc_name, di.d_primary.di_nbname,
 471             &tv, &tm) != 0)
 472                 return;

 474         if (settimeofday(&tv, 0))
 475                 smb_tracef("unable to set system time");

 477         tsecs = time(0);
 478         (void) localtime_r(&tsecs, &tm);
 479         smb_tracef("SrvsvcTimeSync %s", ctime((time_t *)&tv.tv_sec));
 480 }
```

_____***unchanged_portion_omitted_***