# gsasl

2.2.3

Generated by Doxygen 1.9.8

# Chapter 1

# GNU SASL Library

## 1.1 Introduction

GNU SASL is an implementation of the Simple Authentication and Security Layer framework and a few common SASL mechanisms. SASL is used by network servers (e.g., IMAP, SMTP) to request authentication from clients, and in clients to authenticate against servers.

GNU SASL consists of a library ('libgsasl'), a command line utility ('gsasl') to access the library from the shell, and a manual. The library includes support for the framework (with authentication functions and application data privacy and integrity functions) and at least partial support for the CRAM-MD5, EXTERNAL, GSSAPI, ANONYMOUS, PLAIN, SECURID, DIGEST-MD5, LOGIN, and NTLM mechanisms.

The library is easily ported because it does not do network communication by itself, but rather leaves it up to the calling application. The library is flexible with regards to the authorization infrastructure used, as it utilize a callback into the application to decide whether a user is authorized or not.

GNU SASL is developed for the GNU/Linux system, but runs on over 20 platforms including most major Unix platforms and Windows, and many kind of devices including iPAQ handhelds and S/390 mainframes.

GNU SASL is written in pure ANSI C89 to be portable to embedded and otherwise limited platforms. The entire library, with full support for ANONYMOUS, EXTERNAL, PLAIN, LOGIN and CRAM-MD5, and the front-end that support client and server mode, and the IMAP and SMTP protocols, fits in under 60kb on an Intel x86 platform, without any modifications to the code. (This figure was accurate as of version 0.0.13.)

The library is licensed under the GNU Lesser General Public License, and the command-line interface, self-tests and examples are licensed under the GNU General Public License.

The project web page:
http://www.gnu.org/software/gsasl/

The software archive:
ftp://alpha.gnu.org/pub/gnu/gsasl/

Further information and paid contract development:
Simon Josefsson simon@josefsson.org

## 1.2 Logical overview

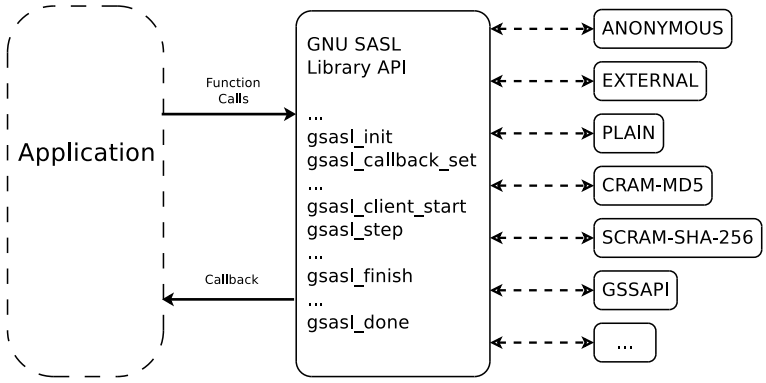


**Figure 1.1 Logical overview**

## 1.3 Control flow in application using the library

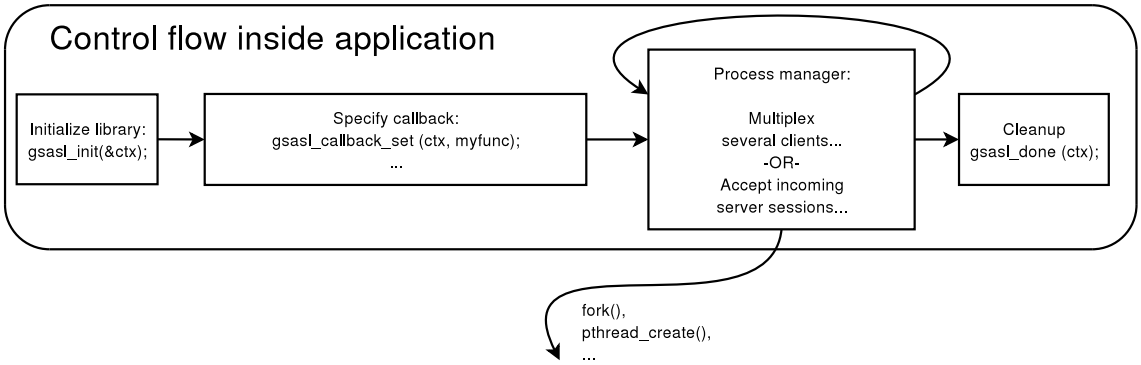


**Figure 1.2 Control flow**

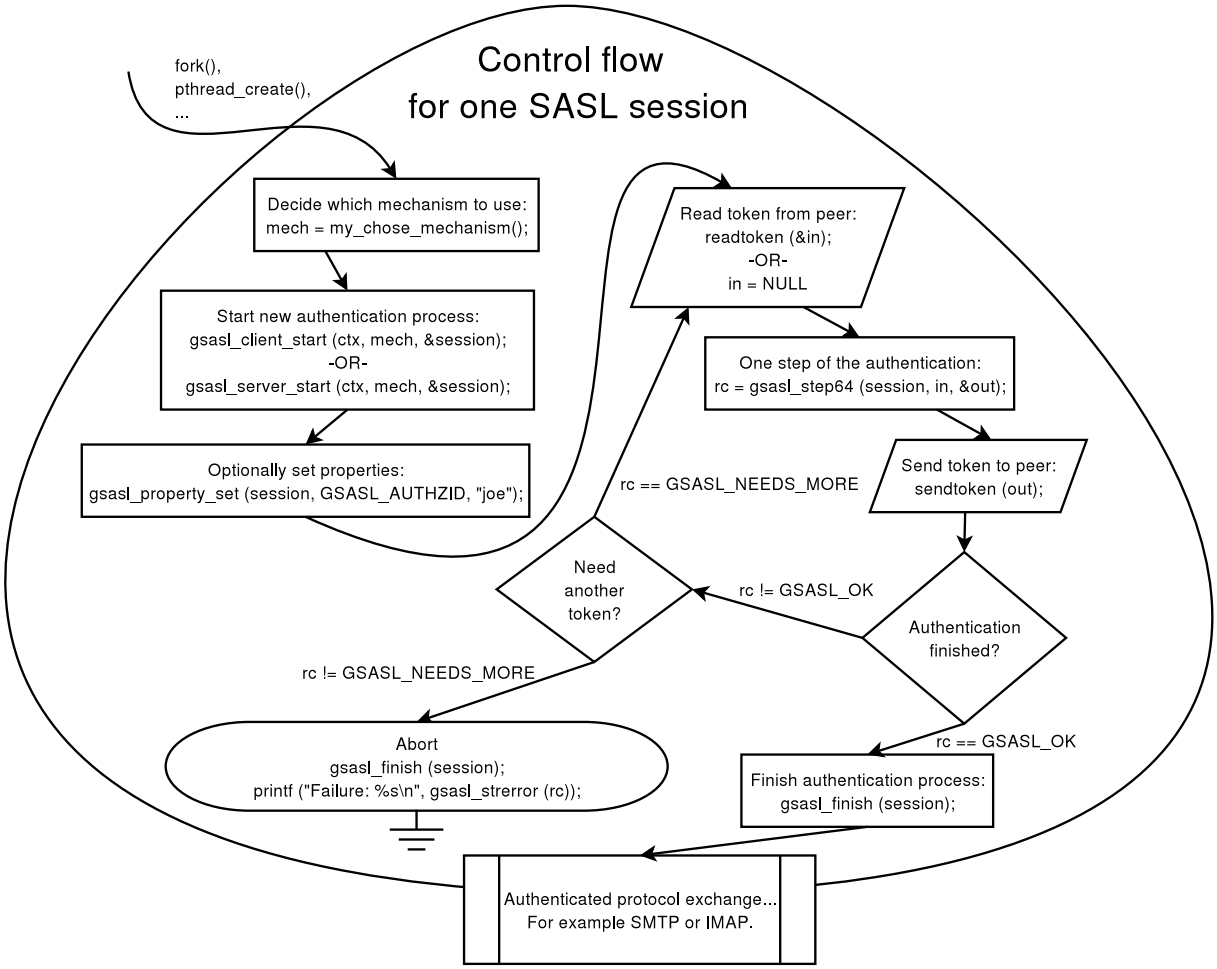


**Figure 1.3 Control flow**

## 1.4 Examples

```
/* client.c --- Example SASL client.
 * Copyright (C) 2004-2026 Simon Josefsson
 *
 * This file is part of GNU SASL.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

#include <config.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <gsasl.h>

static void
client_authenticate (Gsasl_session *session)
{
  char buf[BUFSIZ] = "";
  char *p;
```

```
  int rc;

  /* This loop mimics a protocol where the client send data first. */

  do
    {
      /* Generate client output. */
      rc = gsasl_step64 (session, buf, &p);

      if (rc == GSASL_NEEDS_MORE || rc == GSASL_OK)
        {
          /* If successful, print it. */
          printf ("Output:\n%s\n", p);
          gsasl_free (p);
        }

      if (rc == GSASL_NEEDS_MORE)
        {
          /* If the client need more data from server, get it here. */
          printf ("Input base64 encoded data from server:\n");
          p = fgets (buf, sizeof (buf) - 1, stdin);
          if (p == NULL)
            {
              perror ("fgets");
              return;
            }
          if (buf[strlen (buf) - 1] == '\n')
            buf[strlen (buf) - 1] = '\0';
        }
    }
  while (rc == GSASL_NEEDS_MORE);

  printf ("\n");

  if (rc != GSASL_OK)
    {
      printf ("Authentication error (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* The client is done.  Here you would typically check if the server
     let the client in.  If not, you could try again. */

  printf ("If server accepted us, we're done.\n");
}

static void
client (Gsasl *ctx)
{
  Gsasl_session *session;
  const char *mech = "PLAIN";
  int rc;

  /* Create new authentication session. */
  if ((rc = gsasl_client_start (ctx, mech, &session)) != GSASL_OK)
    {
      printf ("Cannot initialize client (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* Set username and password in session handle.  This info will be
     lost when this session is deallocated below.  */
  rc = gsasl_property_set (session, GSASL_AUTHID, "jas");
  if (rc != GSASL_OK)
    {
      printf ("Cannot set property (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }
  rc = gsasl_property_set (session, GSASL_PASSWORD, "secret");
  if (rc != GSASL_OK)
    {
      printf ("Cannot set property (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* Do it. */
  client_authenticate (session);

  /* Cleanup. */
  gsasl_finish (session);
}

int
main (void)
{
  Gsasl *ctx = NULL;
```

```
  int rc;

  /* Initialize library. */
  if ((rc = gsasl_init (&ctx)) != GSASL_OK)
    {
      printf ("Cannot initialize libgsasl (%d): %s", rc, gsasl_strerror (rc));
      return 1;
    }

  /* Do it. */
  client (ctx);

  /* Cleanup. */
  gsasl_done (ctx);

  return 0;
}
/* client-serverfirst.c --- Example SASL client, where server send data first.
 * Copyright (C) 2004-2026 Simon Josefsson
 *
 * This file is part of GNU SASL.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

#include <config.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <gsasl.h>

static void
client_authenticate (Gsasl_session *session)
{
  char buf[BUFSIZ] = "";
  char *p;
  int rc;

  /* This loop mimics a protocol where the server send data first. */

  do
    {
      printf ("Input base64 encoded data from server:\n");
      p = fgets (buf, sizeof (buf) - 1, stdin);
      if (p == NULL)
        {
          perror ("fgets");
          return;
        }
      if (buf[strlen (buf) - 1] == '\n')
        buf[strlen (buf) - 1] = '\0';

      rc = gsasl_step64 (session, buf, &p);

      if (rc == GSASL_NEEDS_MORE || rc == GSASL_OK)
        {
          printf ("Output:\n%s\n", p);
          gsasl_free (p);
        }
    }
  while (rc == GSASL_NEEDS_MORE);

  printf ("\n");

  if (rc != GSASL_OK)
    {
      printf ("Authentication error (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* The client is done.  Here you would typically check if the server
```

```
     let the client in.  If not, you could try again. */

  printf ("If server accepted us, we're done.\n");
}

static void
client (Gsasl *ctx)
{
  Gsasl_session *session;
  const char *mech = "CRAM-MD5";
  int rc;

  /* Create new authentication session. */
  if ((rc = gsasl_client_start (ctx, mech, &session)) != GSASL_OK)
    {
      printf ("Cannot initialize client (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* Set username and password in session handle.  This info will be
     lost when this session is deallocated below.  */
  rc = gsasl_property_set (session, GSASL_AUTHID, "jas");
  if (rc != GSASL_OK)
    {
      printf ("Cannot set property (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }
  rc = gsasl_property_set (session, GSASL_PASSWORD, "secret");
  if (rc != GSASL_OK)
    {
      printf ("Cannot set property (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* Do it. */
  client_authenticate (session);

  /* Cleanup. */
  gsasl_finish (session);
}

int
main (void)
{
  Gsasl *ctx = NULL;
  int rc;

  /* Initialize library. */
  if ((rc = gsasl_init (&ctx)) != GSASL_OK)
    {
      printf ("Cannot initialize libgsasl (%d): %s", rc, gsasl_strerror (rc));
      return 1;
    }

  /* Do it. */
  client (ctx);

  /* Cleanup. */
  gsasl_done (ctx);

  return 0;
}
/* client-mech.c --- Example SASL client, with a choice of mechanism to use.
 * Copyright (C) 2004-2026 Simon Josefsson
 *
 * This file is part of GNU SASL.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

#include <config.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>

#include <gsasl.h>

static void
client_authenticate (Gsasl_session *session)
{
  char buf[BUFSIZ] = "";
  char *p;
  int rc;

  /* This loop mimics a protocol where the server send data first. */

  do
    {
      printf ("Input base64 encoded data from server:\n");
      p = fgets (buf, sizeof (buf) - 1, stdin);
      if (p == NULL)
        {
          perror ("fgets");
          return;
        }
      if (buf[strlen (buf) - 1] == '\n')
        buf[strlen (buf) - 1] = '\0';

      rc = gsasl_step64 (session, buf, &p);

      if (rc == GSASL_NEEDS_MORE || rc == GSASL_OK)
        {
          printf ("Output:\n%s\n", p);
          gsasl_free (p);
        }
    }
  while (rc == GSASL_NEEDS_MORE);

  printf ("\n");

  if (rc != GSASL_OK)
    {
      printf ("Authentication error (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* The client is done.  Here you would typically check if the server
     let the client in.  If not, you could try again. */

  printf ("If server accepted us, we're done.\n");
}

static const char *
client_mechanism (Gsasl *ctx)
{
  static char mech[GSASL_MAX_MECHANISM_SIZE + 1] = "";
  char mechlist[BUFSIZ] = "";
  const char *suggestion;
  char *p;

  printf ("Enter list of server supported mechanisms, separate by SPC:\n");
  p = fgets (mechlist, sizeof (mechlist) - 1, stdin);
  if (p == NULL)
    {
      perror ("fgets");
      return NULL;
    }

  suggestion = gsasl_client_suggest_mechanism (ctx, mechlist);
  if (suggestion)
    printf ("Library suggests use of `%s'.\n", suggestion);

  printf ("Enter mechanism to use:\n");
  p = fgets (mech, sizeof (mech) - 1, stdin);
  if (p == NULL)
    {
      perror ("fgets");
      return NULL;
    }

  mech[strlen (mech) - 1] = '\0';

  return mech;
}

static void
client (Gsasl *ctx)
{
  Gsasl_session *session;
  const char *mech;
```

```
  int rc;

  /* Find out which mechanism to use. */
  mech = client_mechanism (ctx);

  /* Create new authentication session. */
  if ((rc = gsasl_client_start (ctx, mech, &session)) != GSASL_OK)
    {
      printf ("Cannot initialize client (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* Set username and password in session handle.  This info will be
     lost when this session is deallocated below.  */
  rc = gsasl_property_set (session, GSASL_AUTHID, "jas");
  if (rc != GSASL_OK)
    {
      printf ("Cannot set property (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }
  rc = gsasl_property_set (session, GSASL_PASSWORD, "secret");
  if (rc != GSASL_OK)
    {
      printf ("Cannot set property (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* Do it. */
  client_authenticate (session);

  /* Cleanup. */
  gsasl_finish (session);
}

int
main (void)
{
  Gsasl *ctx = NULL;
  int rc;

  /* Initialize library. */
  if ((rc = gsasl_init (&ctx)) != GSASL_OK)
    {
      printf ("Cannot initialize libgsasl (%d): %s", rc, gsasl_strerror (rc));
      return 1;
    }

  /* Do it. */
  client (ctx);

  /* Cleanup. */
  gsasl_done (ctx);

  return 0;
}
/* client-callback.c --- Example SASL client, with callback for user info.
 * Copyright (C) 2004-2026 Simon Josefsson
 *
 * This file is part of GNU SASL.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

#include <config.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <gsasl.h>

static void
client_authenticate (Gsasl_session *session)
{
  char buf[BUFSIZ] = "";
```

```
  char *p;
  int rc;

  /* This loop mimics a protocol where the server send data first. */

  do
    {
      printf ("Input base64 encoded data from server:\n");
      p = fgets (buf, sizeof (buf) - 1, stdin);
      if (p == NULL)
        {
          perror ("fgets");
          return;
        }
      if (buf[strlen (buf) - 1] == '\n')
        buf[strlen (buf) - 1] = '\0';

      rc = gsasl_step64 (session, buf, &p);

      if (rc == GSASL_NEEDS_MORE || rc == GSASL_OK)
        {
          printf ("Output:\n%s\n", p);
          gsasl_free (p);
        }
    }
  while (rc == GSASL_NEEDS_MORE);

  printf ("\n");

  if (rc != GSASL_OK)
    {
      printf ("Authentication error (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* The client is done.  Here you would typically check if the server
     let the client in.  If not, you could try again. */

  printf ("If server accepted us, we're done.\n");
}

static void
client (Gsasl *ctx)
{
  Gsasl_session *session;
  const char *mech = "SECURID";
  int rc;

  /* Create new authentication session. */
  if ((rc = gsasl_client_start (ctx, mech, &session)) != GSASL_OK)
    {
      printf ("Cannot initialize client (%d): %s\n", rc, gsasl_strerror (rc));
      return;
    }

  /* Do it. */
  client_authenticate (session);

  /* Cleanup. */
  gsasl_finish (session);
}

static int
callback (Gsasl *ctx, Gsasl_session *sctx, Gsasl_property prop)
{
  char buf[BUFSIZ] = "";
  int rc = GSASL_NO_CALLBACK;
  char *p;

  (void) ctx;

  /* Get user info from user. */

  printf ("Callback invoked, for property %u.\n", prop);

  switch (prop)
    {
    case GSASL_PASSCODE:
      printf ("Enter passcode:\n");
      p = fgets (buf, sizeof (buf) - 1, stdin);
      if (p == NULL)
        {
          perror ("fgets");
          break;
        }
```

```
      buf[strlen (buf) - 1] = '\0';

      rc = gsasl_property_set (sctx, GSASL_PASSCODE, buf);
      break;

    case GSASL_AUTHID:
      printf ("Enter username:\n");
      p = fgets (buf, sizeof (buf) - 1, stdin);
      if (p == NULL)
        {
          perror ("fgets");
          break;
        }
      buf[strlen (buf) - 1] = '\0';

      rc = gsasl_property_set (sctx, GSASL_AUTHID, buf);
      break;

    default:
      printf ("Unknown property!  Don't worry.\n");
      break;
    }

  return rc;
}

int
main (void)
{
  Gsasl *ctx = NULL;
  int rc;

  /* Initialize library. */
  if ((rc = gsasl_init (&ctx)) != GSASL_OK)
    {
      printf ("Cannot initialize libgsasl (%d): %s", rc, gsasl_strerror (rc));
      return 1;
    }

  /* Set the callback handler for the library. */
  gsasl_callback_set (ctx, callback);

  /* Do it. */
  client (ctx);

  /* Cleanup. */
  gsasl_done (ctx);

  return 0;
}
```

# Chapter 2

# Data Structure Index

## 2.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all files with brief descriptions:

# Chapter 4

# Data Structure Documentation

## 4.1 _Gsasl_digest_md5_client_state Struct Reference

**Data Fields**

- int step
- unsigned long readseqnum
- unsigned long sendseqnum
- char secret [DIGEST_MD5_LENGTH]
- char kic [DIGEST_MD5_LENGTH]
- char kcc [DIGEST_MD5_LENGTH]
- char kis [DIGEST_MD5_LENGTH]
- char kcs [DIGEST_MD5_LENGTH]
- digest_md5_challenge challenge
- digest_md5_response response
- digest_md5_finish finish

### 4.1.1 Detailed Description

Definition at line 49 of file digest-md5/client.c.

### 4.1.2 Field Documentation

#### 4.1.2.1 challenge

```
digest_md5_challenge _Gsasl_digest_md5_client_state::challenge
```

Definition at line 58 of file digest-md5/client.c.

#### 4.1.2.2 finish

```
digest_md5_finish _Gsasl_digest_md5_client_state::finish
```

Definition at line 60 of file digest-md5/client.c.

#### 4.1.2.3 kcc

```
char _Gsasl_digest_md5_client_state::kcc[DIGEST_MD5_LENGTH]
```

Definition at line 55 of file digest-md5/client.c.

#### 4.1.2.4 kcs

```
char _Gsasl_digest_md5_client_state::kcs[DIGEST_MD5_LENGTH]
```

Definition at line 57 of file digest-md5/client.c.

#### 4.1.2.5 kic

```
char _Gsasl_digest_md5_client_state::kic[DIGEST_MD5_LENGTH]
```

Definition at line 54 of file digest-md5/client.c.

#### 4.1.2.6 kis

```
char _Gsasl_digest_md5_client_state::kis[DIGEST_MD5_LENGTH]
```

Definition at line 56 of file digest-md5/client.c.

#### 4.1.2.7 readseqnum

```
unsigned long _Gsasl_digest_md5_client_state::readseqnum
```

Definition at line 52 of file digest-md5/client.c.

#### 4.1.2.8 response

```
digest_md5_response _Gsasl_digest_md5_client_state::response
```

Definition at line 59 of file digest-md5/client.c.

#### 4.1.2.9 secret

```
char _Gsasl_digest_md5_client_state::secret[DIGEST_MD5_LENGTH]
```

Definition at line 53 of file digest-md5/client.c.

#### 4.1.2.10 sendseqnum

```
unsigned long _Gsasl_digest_md5_client_state::sendseqnum
```

Definition at line 52 of file digest-md5/client.c.

#### 4.1.2.11 step

```
int _Gsasl_digest_md5_client_state::step
```

Definition at line 51 of file digest-md5/client.c.

The documentation for this struct was generated from the following file:

- digest-md5/client.c

## 4.2 _Gsasl_digest_md5_server_state Struct Reference

**Data Fields**

- int step
- unsigned long readseqnum
- unsigned long sendseqnum
- char secret [DIGEST_MD5_LENGTH]
- char kic [DIGEST_MD5_LENGTH]
- char kcc [DIGEST_MD5_LENGTH]
- char kis [DIGEST_MD5_LENGTH]
- char kcs [DIGEST_MD5_LENGTH]
- digest_md5_challenge challenge
- digest_md5_response response
- digest_md5_finish finish

### 4.2.1 Detailed Description

Definition at line 50 of file digest-md5/server.c.

### 4.2.2 Field Documentation

#### 4.2.2.1 challenge

```
digest_md5_challenge _Gsasl_digest_md5_server_state::challenge
```

Definition at line 59 of file digest-md5/server.c.

#### 4.2.2.2 finish

```
digest_md5_finish _Gsasl_digest_md5_server_state::finish
```

Definition at line 61 of file digest-md5/server.c.

#### 4.2.2.3 kcc

```
char _Gsasl_digest_md5_server_state::kcc[DIGEST_MD5_LENGTH]
```

Definition at line 56 of file digest-md5/server.c.

#### 4.2.2.4 kcs

```
char _Gsasl_digest_md5_server_state::kcs[DIGEST_MD5_LENGTH]
```

Definition at line 58 of file digest-md5/server.c.

#### 4.2.2.5 kic

```
char _Gsasl_digest_md5_server_state::kic[DIGEST_MD5_LENGTH]
```

Definition at line 55 of file digest-md5/server.c.

#### 4.2.2.6 kis

```
char _Gsasl_digest_md5_server_state::kis[DIGEST_MD5_LENGTH]
```

Definition at line 57 of file digest-md5/server.c.

#### 4.2.2.7 readseqnum

```
unsigned long _Gsasl_digest_md5_server_state::readseqnum
```

Definition at line 53 of file digest-md5/server.c.

#### 4.2.2.8 response

```
digest_md5_response _Gsasl_digest_md5_server_state::response
```

Definition at line 60 of file digest-md5/server.c.

#### 4.2.2.9 secret

```
char _Gsasl_digest_md5_server_state::secret[DIGEST_MD5_LENGTH]
```

Definition at line 54 of file digest-md5/server.c.

#### 4.2.2.10 sendseqnum

```
unsigned long _Gsasl_digest_md5_server_state::sendseqnum
```

Definition at line 53 of file digest-md5/server.c.

#### 4.2.2.11 step

```
int _Gsasl_digest_md5_server_state::step
```

Definition at line 52 of file digest-md5/server.c.

The documentation for this struct was generated from the following file:

- digest-md5/server.c

## 4.3 _gsasl_gs2_client_state Struct Reference

**Data Fields**

- int step
- gss_name_t service
- gss_ctx_id_t context
- gss_OID mech_oid
- gss_buffer_desc token
- struct gss_channel_bindings_struct cb

### 4.3.1 Detailed Description

Definition at line 36 of file gs2/client.c.

### 4.3.2 Field Documentation

#### 4.3.2.1 cb

```
struct gss_channel_bindings_struct _gsasl_gs2_client_state::cb
```

Definition at line 44 of file gs2/client.c.

#### 4.3.2.2 context

```
gss_ctx_id_t _gsasl_gs2_client_state::context
```

Definition at line 41 of file gs2/client.c.

#### 4.3.2.3 mech_oid

```
gss_OID _gsasl_gs2_client_state::mech_oid
```

Definition at line 42 of file gs2/client.c.

#### 4.3.2.4 service

```
gss_name_t _gsasl_gs2_client_state::service
```

Definition at line 40 of file gs2/client.c.

#### 4.3.2.5 step

```
int _gsasl_gs2_client_state::step
```

Definition at line 39 of file gs2/client.c.

#### 4.3.2.6 token

```
gss_buffer_desc _gsasl_gs2_client_state::token
```

Definition at line 43 of file gs2/client.c.

The documentation for this struct was generated from the following file:

- gs2/client.c

## 4.4 _Gsasl_gs2_server_state Struct Reference

### Data Fields

- int step
- gss_name_t client
- gss_cred_id_t cred
- gss_ctx_id_t context
- gss_OID mech_oid
- struct gss_channel_bindings_struct cb

### 4.4.1 Detailed Description

Definition at line 40 of file gs2/server.c.

### 4.4.2 Field Documentation

#### 4.4.2.1 cb

```
struct gss_channel_bindings_struct _Gsasl_gs2_server_state::cb
```

Definition at line 48 of file gs2/server.c.

#### 4.4.2.2 client

```
gss_name_t _Gsasl_gs2_server_state::client
```

Definition at line 44 of file gs2/server.c.

#### 4.4.2.3 context

```
gss_ctx_id_t _Gsasl_gs2_server_state::context
```

Definition at line 46 of file gs2/server.c.

#### 4.4.2.4 cred

```
gss_cred_id_t _Gsasl_gs2_server_state::cred
```

Definition at line 45 of file gs2/server.c.

#### 4.4.2.5 mech_oid

```
gss_OID _Gsasl_gs2_server_state::mech_oid
```

Definition at line 47 of file gs2/server.c.

#### 4.4.2.6 step

```
int _Gsasl_gs2_server_state::step
```

Definition at line 43 of file gs2/server.c.

The documentation for this struct was generated from the following file:

- gs2/server.c

## 4.5 _Gsasl_gssapi_client_state Struct Reference

### Data Fields

- int step
- gss_name_t service
- gss_ctx_id_t context
- gss_qop_t qop

### 4.5.1 Detailed Description

Definition at line 36 of file gssapi/client.c.

### 4.5.2 Field Documentation

#### 4.5.2.1 context

```
gss_ctx_id_t _Gsasl_gssapi_client_state::context
```

Definition at line 40 of file gssapi/client.c.

#### 4.5.2.2 qop

```
gss_qop_t _Gsasl_gssapi_client_state::qop
```

Definition at line 41 of file gssapi/client.c.

#### 4.5.2.3 service

```
gss_name_t _Gsasl_gssapi_client_state::service
```

Definition at line 39 of file gssapi/client.c.

#### 4.5.2.4 step

```
int _Gsasl_gssapi_client_state::step
```

Definition at line 38 of file gssapi/client.c.

The documentation for this struct was generated from the following file:

- gssapi/client.c

## 4.6 _Gsasl_gssapi_server_state Struct Reference

**Data Fields**

- int step
- gss_name_t client
- gss_cred_id_t cred
- gss_ctx_id_t context

### 4.6.1 Detailed Description

Definition at line 36 of file gssapi/server.c.

### 4.6.2 Field Documentation

#### 4.6.2.1 client

```
gss_name_t _Gsasl_gssapi_server_state::client
```

Definition at line 39 of file gssapi/server.c.

#### 4.6.2.2 context

```
gss_ctx_id_t _Gsasl_gssapi_server_state::context
```

Definition at line 41 of file gssapi/server.c.

#### 4.6.2.3 cred

```
gss_cred_id_t _Gsasl_gssapi_server_state::cred
```

Definition at line 40 of file gssapi/server.c.

#### 4.6.2.4 step

```
int _Gsasl_gssapi_server_state::step
```

Definition at line 38 of file gssapi/server.c.

The documentation for this struct was generated from the following file:

- gssapi/server.c

## 4.7 _Gsasl_login_client_state Struct Reference

**Data Fields**

- int step

### 4.7.1 Detailed Description

Definition at line 33 of file login/client.c.

### 4.7.2 Field Documentation

#### 4.7.2.1 step

```
int _Gsasl_login_client_state::step
```

Definition at line 35 of file login/client.c.

The documentation for this struct was generated from the following file:

- login/client.c

## 4.8 _Gsasl_login_server_state Struct Reference

**Data Fields**

- int step
- char * username
- char * password

### 4.8.1 Detailed Description

Definition at line 33 of file login/server.c.

### 4.8.2 Field Documentation

#### 4.8.2.1 password

```
char* _Gsasl_login_server_state::password
```

Definition at line 37 of file login/server.c.

#### 4.8.2.2 step

```
int _Gsasl_login_server_state::step
```

Definition at line 35 of file login/server.c.

#### 4.8.2.3 username

```
char* _Gsasl_login_server_state::username
```

Definition at line 36 of file login/server.c.

The documentation for this struct was generated from the following file:

- login/server.c

## 4.9 _Gsasl_ntlm_state Struct Reference

### Data Fields

- int step

### 4.9.1 Detailed Description

Definition at line 35 of file ntlm.c.

### 4.9.2 Field Documentation

#### 4.9.2.1 step

```
int _Gsasl_ntlm_state::step
```

Definition at line 37 of file ntlm.c.

The documentation for this struct was generated from the following file:

- ntlm.c

## 4.10 digest_md5_challenge Struct Reference

```
#include <tokens.h>
```

### Data Fields

- size_t nrealms
- char ** realms
- char * nonce
- int qops
- int stale
- unsigned long servermaxbuf
- int utf8
- int ciphers

### 4.10.1 Detailed Description

Definition at line 81 of file digest-md5/tokens.h.

## 4.10.2 Field Documentation

### 4.10.2.1 ciphers

```
int digest_md5_challenge::ciphers
```

Definition at line 90 of file digest-md5/tokens.h.

### 4.10.2.2 nonce

```
char* digest_md5_challenge::nonce
```

Definition at line 85 of file digest-md5/tokens.h.

### 4.10.2.3 nrealms

```
size_t digest_md5_challenge::nrealms
```

Definition at line 83 of file digest-md5/tokens.h.

### 4.10.2.4 qops

```
int digest_md5_challenge::qops
```

Definition at line 86 of file digest-md5/tokens.h.

### 4.10.2.5 realms

```
char** digest_md5_challenge::realms
```

Definition at line 84 of file digest-md5/tokens.h.

### 4.10.2.6 servermaxbuf

```
unsigned long digest_md5_challenge::servermaxbuf
```

Definition at line 88 of file digest-md5/tokens.h.

### 4.10.2.7 stale

```
int digest_md5_challenge::stale
```

Definition at line 87 of file digest-md5/tokens.h.

#### 4.10.2.8 utf8

```
int digest_md5_challenge::utf8
```

Definition at line 89 of file digest-md5/tokens.h.

The documentation for this struct was generated from the following file:

- digest-md5/tokens.h

## 4.11 digest_md5_finish Struct Reference

```
#include <tokens.h>
```

### Data Fields

- char rspauth [DIGEST_MD5_RESPONSE_LENGTH+1]

### 4.11.1 Detailed Description

Definition at line 145 of file digest-md5/tokens.h.

### 4.11.2 Field Documentation

#### 4.11.2.1 rspauth

```
char digest_md5_finish::rspauth[DIGEST_MD5_RESPONSE_LENGTH+1]
```

Definition at line 147 of file digest-md5/tokens.h.

The documentation for this struct was generated from the following file:

- digest-md5/tokens.h

## 4.12 digest_md5_response Struct Reference

```
#include <tokens.h>
```

**Data Fields**

- char * username
- char * realm
- char * nonce
- char * cnonce
- unsigned long nc
- digest_md5_qop qop
- char * digesturi
- unsigned long clientmaxbuf
- int utf8
- digest_md5_cipher cipher
- char * authzid
- char response [DIGEST_MD5_RESPONSE_LENGTH+1]

### 4.12.1 Detailed Description

Definition at line 125 of file digest-md5/tokens.h.

### 4.12.2 Field Documentation

#### 4.12.2.1 authzid

```
char* digest_md5_response::authzid
```

Definition at line 137 of file digest-md5/tokens.h.

#### 4.12.2.2 cipher

```
digest_md5_cipher digest_md5_response::cipher
```

Definition at line 136 of file digest-md5/tokens.h.

#### 4.12.2.3 clientmaxbuf

```
unsigned long digest_md5_response::clientmaxbuf
```

Definition at line 134 of file digest-md5/tokens.h.

#### 4.12.2.4 cnonce

```
char* digest_md5_response::cnonce
```

Definition at line 130 of file digest-md5/tokens.h.

#### 4.12.2.5 digesturi

```
char* digest_md5_response::digesturi
```

Definition at line 133 of file digest-md5/tokens.h.

#### 4.12.2.6 nc

```
unsigned long digest_md5_response::nc
```

Definition at line 131 of file digest-md5/tokens.h.

#### 4.12.2.7 nonce

```
char* digest_md5_response::nonce
```

Definition at line 129 of file digest-md5/tokens.h.

#### 4.12.2.8 qop

```
digest_md5_qop digest_md5_response::qop
```

Definition at line 132 of file digest-md5/tokens.h.

#### 4.12.2.9 realm

```
char* digest_md5_response::realm
```

Definition at line 128 of file digest-md5/tokens.h.

#### 4.12.2.10 response

```
char digest_md5_response::response[DIGEST_MD5_RESPONSE_LENGTH+1]
```

Definition at line 138 of file digest-md5/tokens.h.

#### 4.12.2.11 username

```
char* digest_md5_response::username
```

Definition at line 127 of file digest-md5/tokens.h.

#### 4.12.2.12 utf8

```
int digest_md5_response::utf8
```

Definition at line 135 of file digest-md5/tokens.h.

The documentation for this struct was generated from the following file:

- digest-md5/tokens.h

## 4.13 Gsasl Struct Reference

```
#include <internal.h>
```

### Data Fields

- size_t n_client_mechs
- Gsasl_mechanism * client_mechs
- size_t n_server_mechs
- Gsasl_mechanism * server_mechs
- Gsasl_callback_function cb
- void * application_hook

### 4.13.1 Detailed Description

Definition at line 35 of file internal.h.

### 4.13.2 Field Documentation

#### 4.13.2.1 application_hook

```
void* Gsasl::application_hook
```

Definition at line 43 of file internal.h.

#### 4.13.2.2 cb

```
Gsasl_callback_function Gsasl::cb
```

Definition at line 42 of file internal.h.

#### 4.13.2.3 client_mechs

```
Gsasl_mechanism* Gsasl::client_mechs
```

Definition at line 38 of file internal.h.

#### 4.13.2.4 n_client_mechs

```
size_t Gsasl::n_client_mechs
```

Definition at line 37 of file internal.h.

#### 4.13.2.5 n_server_mechs

```
size_t Gsasl::n_server_mechs
```

Definition at line 39 of file internal.h.

#### 4.13.2.6 server_mechs

```
Gsasl_mechanism* Gsasl::server_mechs
```

Definition at line 40 of file internal.h.

The documentation for this struct was generated from the following file:

- internal.h

## 4.14 Gsasl_mechanism Struct Reference

```
#include <gsasl-mech.h>
```

### Data Fields

- const char * name
- struct Gsasl_mechanism_functions client
- struct Gsasl_mechanism_functions server

### 4.14.1 Detailed Description

Gsasl_mechanism:

**Parameters**

| | |
|---|---|
| *name* | string holding name of mechanism, e.g., "PLAIN". |
| *client* | client-side Gsasl_mechanism_functions structure. |
| *server* | server-side Gsasl_mechanism_functions structure. |

Holds all implementation details about a mechanism.

Definition at line 170 of file gsasl-mech.h.

### 4.14.2 Field Documentation

#### 4.14.2.1 client

```
struct Gsasl_mechanism_functions Gsasl_mechanism::client
```

Definition at line 174 of file gsasl-mech.h.

#### 4.14.2.2 name

```
const char* Gsasl_mechanism::name
```

Definition at line 172 of file gsasl-mech.h.

#### 4.14.2.3 server

```
struct Gsasl_mechanism_functions Gsasl_mechanism::server
```

Definition at line 175 of file gsasl-mech.h.

The documentation for this struct was generated from the following file:

- gsasl-mech.h

## 4.15 Gsasl_mechanism_functions Struct Reference

```
#include <gsasl-mech.h>
```

**Data Fields**

- Gsasl_init_function init
- Gsasl_done_function done
- Gsasl_start_function start
- Gsasl_step_function step
- Gsasl_finish_function finish
- Gsasl_code_function encode
- Gsasl_code_function decode

### 4.15.1 Detailed Description

Gsasl_mechanism_functions:

**Parameters**

| | |
|---|---|
| *init* | a Gsasl_init_function(). |
| *done* | a Gsasl_done_function(). |
| *start* | a Gsasl_start_function(). |
| *step* | a Gsasl_step_function(). |
| *finish* | a Gsasl_finish_function(). |
| *encode* | a Gsasl_code_function(). |

Holds all function pointers to implement a mechanism, in either client or server mode.

Definition at line 150 of file gsasl-mech.h.

## 4.15.2 Field Documentation

### 4.15.2.1 decode

```
Gsasl_code_function Gsasl_mechanism_functions::decode
```

Definition at line 158 of file gsasl-mech.h.

### 4.15.2.2 done

```
Gsasl_done_function Gsasl_mechanism_functions::done
```

Definition at line 153 of file gsasl-mech.h.

### 4.15.2.3 encode

```
Gsasl_code_function Gsasl_mechanism_functions::encode
```

Definition at line 157 of file gsasl-mech.h.

### 4.15.2.4 finish

```
Gsasl_finish_function Gsasl_mechanism_functions::finish
```

Definition at line 156 of file gsasl-mech.h.

### 4.15.2.5 init

```
Gsasl_init_function Gsasl_mechanism_functions::init
```

Definition at line 152 of file gsasl-mech.h.

### 4.15.2.6 start

```
Gsasl_start_function Gsasl_mechanism_functions::start
```

Definition at line 154 of file gsasl-mech.h.

#### 4.15.2.7 step

```
Gsasl_step_function Gsasl_mechanism_functions::step
```

Definition at line 155 of file gsasl-mech.h.

The documentation for this struct was generated from the following file:

- gsasl-mech.h

## 4.16 Gsasl_session Struct Reference

```
#include <internal.h>
```

### Data Fields

- Gsasl * ctx
- int clientp
- Gsasl_mechanism * mech
- void * mech_data
- void * application_hook
- char * anonymous_token
- char * authid
- char * authzid
- char * password
- char * passcode
- char * pin
- char * suggestedpin
- char * service
- char * hostname
- char * gssapi_display_name
- char * realm
- char * digest_md5_hashed_password
- char * qops
- char * qop
- char * scram_iter
- char * scram_salt
- char * scram_salted_password
- char * scram_serverkey
- char * scram_storedkey
- char * cb_tls_unique
- char * cb_tls_exporter
- char * saml20_idp_identifier
- char * saml20_redirect_url
- char * openid20_redirect_url
- char * openid20_outcome_data

### 4.16.1 Detailed Description

Definition at line 47 of file internal.h.

## 4.16.2 Field Documentation

### 4.16.2.1 anonymous_token

```
char* Gsasl_session::anonymous_token
```

Definition at line 56 of file internal.h.

### 4.16.2.2 application_hook

```
void* Gsasl_session::application_hook
```

Definition at line 53 of file internal.h.

### 4.16.2.3 authid

```
char* Gsasl_session::authid
```

Definition at line 57 of file internal.h.

### 4.16.2.4 authzid

```
char* Gsasl_session::authzid
```

Definition at line 58 of file internal.h.

### 4.16.2.5 cb_tls_exporter

```
char* Gsasl_session::cb_tls_exporter
```

Definition at line 76 of file internal.h.

### 4.16.2.6 cb_tls_unique

```
char* Gsasl_session::cb_tls_unique
```

Definition at line 75 of file internal.h.

### 4.16.2.7 clientp

```
int Gsasl_session::clientp
```

Definition at line 50 of file internal.h.

#### 4.16.2.8 ctx

```
Gsasl* Gsasl_session::ctx
```

Definition at line 49 of file internal.h.

#### 4.16.2.9 digest_md5_hashed_password

```
char* Gsasl_session::digest_md5_hashed_password
```

Definition at line 67 of file internal.h.

#### 4.16.2.10 gssapi_display_name

```
char* Gsasl_session::gssapi_display_name
```

Definition at line 65 of file internal.h.

#### 4.16.2.11 hostname

```
char* Gsasl_session::hostname
```

Definition at line 64 of file internal.h.

#### 4.16.2.12 mech

```
Gsasl_mechanism* Gsasl_session::mech
```

Definition at line 51 of file internal.h.

#### 4.16.2.13 mech_data

```
void* Gsasl_session::mech_data
```

Definition at line 52 of file internal.h.

#### 4.16.2.14 openid20_outcome_data

```
char* Gsasl_session::openid20_outcome_data
```

Definition at line 80 of file internal.h.

#### 4.16.2.15 openid20_redirect_url

```
char* Gsasl_session::openid20_redirect_url
```

Definition at line 79 of file internal.h.

#### 4.16.2.16 passcode

```
char* Gsasl_session::passcode
```

Definition at line 60 of file internal.h.

#### 4.16.2.17 password

```
char* Gsasl_session::password
```

Definition at line 59 of file internal.h.

#### 4.16.2.18 pin

```
char* Gsasl_session::pin
```

Definition at line 61 of file internal.h.

#### 4.16.2.19 qop

```
char* Gsasl_session::qop
```

Definition at line 69 of file internal.h.

#### 4.16.2.20 qops

```
char* Gsasl_session::qops
```

Definition at line 68 of file internal.h.

#### 4.16.2.21 realm

```
char* Gsasl_session::realm
```

Definition at line 66 of file internal.h.

#### 4.16.2.22 saml20_idp_identifier

```
char* Gsasl_session::saml20_idp_identifier
```

Definition at line 77 of file internal.h.

#### 4.16.2.23 saml20_redirect_url

```
char* Gsasl_session::saml20_redirect_url
```

Definition at line 78 of file internal.h.

#### 4.16.2.24 scram_iter

```
char* Gsasl_session::scram_iter
```

Definition at line 70 of file internal.h.

#### 4.16.2.25 scram_salt

```
char* Gsasl_session::scram_salt
```

Definition at line 71 of file internal.h.

#### 4.16.2.26 scram_salted_password

```
char* Gsasl_session::scram_salted_password
```

Definition at line 72 of file internal.h.

#### 4.16.2.27 scram_serverkey

```
char* Gsasl_session::scram_serverkey
```

Definition at line 73 of file internal.h.

#### 4.16.2.28 scram_storedkey

```
char* Gsasl_session::scram_storedkey
```

Definition at line 74 of file internal.h.

#### 4.16.2.29 service

```
char* Gsasl_session::service
```

Definition at line 63 of file internal.h.

#### 4.16.2.30 suggestedpin

```
char* Gsasl_session::suggestedpin
```

Definition at line 62 of file internal.h.

The documentation for this struct was generated from the following file:

- internal.h

## 4.17 openid20_client_state Struct Reference

### Data Fields

- int step

### 4.17.1 Detailed Description

Definition at line 39 of file openid20/client.c.

### 4.17.2 Field Documentation

#### 4.17.2.1 step

```
int openid20_client_state::step
```

Definition at line 41 of file openid20/client.c.

The documentation for this struct was generated from the following file:

- openid20/client.c

## 4.18 openid20_server_state Struct Reference

### Data Fields

- int step
- int allow_error_step

### 4.18.1 Detailed Description

Definition at line 36 of file openid20/server.c.

### 4.18.2 Field Documentation

#### 4.18.2.1 allow_error_step

```
int openid20_server_state::allow_error_step
```

Definition at line 39 of file openid20/server.c.

#### 4.18.2.2 step

```
int openid20_server_state::step
```

Definition at line 38 of file openid20/server.c.

The documentation for this struct was generated from the following file:

- openid20/server.c

## 4.19 saml20_client_state Struct Reference

**Data Fields**

- int step

### 4.19.1 Detailed Description

Definition at line 39 of file saml20/client.c.

### 4.19.2 Field Documentation

#### 4.19.2.1 step

```
int saml20_client_state::step
```

Definition at line 41 of file saml20/client.c.

The documentation for this struct was generated from the following file:

- saml20/client.c

## 4.20 saml20_server_state Struct Reference

**Data Fields**

- int step

### 4.20.1 Detailed Description

Definition at line 36 of file saml20/server.c.

### 4.20.2 Field Documentation

#### 4.20.2.1 step

```
int saml20_server_state::step
```

Definition at line 38 of file saml20/server.c.

The documentation for this struct was generated from the following file:

- saml20/server.c

## 4.21 scram_client_final Struct Reference

```
#include <tokens.h>
```

**Data Fields**

- char * cbind
- char * nonce
- char * proof

### 4.21.1 Detailed Description

Definition at line 44 of file scram/tokens.h.

### 4.21.2 Field Documentation

#### 4.21.2.1 cbind

```
char* scram_client_final::cbind
```

Definition at line 46 of file scram/tokens.h.

#### 4.21.2.2 nonce

```
char* scram_client_final::nonce
```

Definition at line 47 of file scram/tokens.h.

#### 4.21.2.3 proof

```
char* scram_client_final::proof
```

Definition at line 48 of file scram/tokens.h.

The documentation for this struct was generated from the following file:

- scram/tokens.h

## 4.22 scram_client_first Struct Reference

```
#include <tokens.h>
```

### Data Fields

- char cbflag
- char * cbname
- char * authzid
- char * username
- char * client_nonce

### 4.22.1 Detailed Description

Definition at line 28 of file scram/tokens.h.

### 4.22.2 Field Documentation

#### 4.22.2.1 authzid

```
char* scram_client_first::authzid
```

Definition at line 32 of file scram/tokens.h.

#### 4.22.2.2 cbflag

```
char scram_client_first::cbflag
```

Definition at line 30 of file scram/tokens.h.

#### 4.22.2.3 cbname

```
char* scram_client_first::cbname
```

Definition at line 31 of file scram/tokens.h.

#### 4.22.2.4 client_nonce

```
char* scram_client_first::client_nonce
```

Definition at line 34 of file scram/tokens.h.

#### 4.22.2.5 username

```
char* scram_client_first::username
```

Definition at line 33 of file scram/tokens.h.

The documentation for this struct was generated from the following file:

- scram/tokens.h

## 4.23 scram_client_state Struct Reference

**Data Fields**

- bool plus
- Gsasl_hash hash
- int step
- char * cfmb
- char * serversignature
- char * authmessage
- struct scram_client_first cf
- struct scram_server_first sf
- struct scram_client_final cl
- struct scram_server_final sl

### 4.23.1 Detailed Description

Definition at line 46 of file scram/client.c.

### 4.23.2 Field Documentation

#### 4.23.2.1 authmessage

```
char* scram_client_state::authmessage
```

Definition at line 53 of file scram/client.c.

#### 4.23.2.2 cf

```
struct scram_client_first scram_client_state::cf
```

Definition at line 54 of file scram/client.c.

#### 4.23.2.3 cfmb

```
char* scram_client_state::cfmb
```

Definition at line 51 of file scram/client.c.

#### 4.23.2.4 cl

```
struct scram_client_final scram_client_state::cl
```

Definition at line 56 of file scram/client.c.

#### 4.23.2.5 hash

```
Gsasl_hash scram_client_state::hash
```

Definition at line 49 of file scram/client.c.

#### 4.23.2.6 plus

```
bool scram_client_state::plus
```

Definition at line 48 of file scram/client.c.

#### 4.23.2.7 serversignature

```
char* scram_client_state::serversignature
```

Definition at line 52 of file scram/client.c.

#### 4.23.2.8 sf

```
struct scram_server_first scram_client_state::sf
```

Definition at line 55 of file scram/client.c.

#### 4.23.2.9 sl

```
struct scram_server_final scram_client_state::sl
```

Definition at line 57 of file scram/client.c.

#### 4.23.2.10 step

```
int scram_client_state::step
```

Definition at line 50 of file scram/client.c.

The documentation for this struct was generated from the following file:

- scram/client.c

## 4.24 scram_server_final Struct Reference

```
#include <tokens.h>
```

### Data Fields

- char * verifier

### 4.24.1 Detailed Description

Definition at line 51 of file scram/tokens.h.

### 4.24.2 Field Documentation

#### 4.24.2.1 verifier

```
char* scram_server_final::verifier
```

Definition at line 53 of file scram/tokens.h.

The documentation for this struct was generated from the following file:

- scram/tokens.h

## 4.25 scram_server_first Struct Reference

```
#include <tokens.h>
```

### Data Fields

- char * nonce
- char * salt
- size_t iter

### 4.25.1 Detailed Description

Definition at line 37 of file scram/tokens.h.

### 4.25.2 Field Documentation

#### 4.25.2.1 iter

```
size_t scram_server_first::iter
```

Definition at line 41 of file scram/tokens.h.

#### 4.25.2.2 nonce

```
char* scram_server_first::nonce
```

Definition at line 39 of file scram/tokens.h.

#### 4.25.2.3 salt

```
char* scram_server_first::salt
```

Definition at line 40 of file scram/tokens.h.

The documentation for this struct was generated from the following file:

- scram/tokens.h

## 4.26 scram_server_state Struct Reference

**Data Fields**

- bool plus
- Gsasl_hash hash
- int step
- char * cbind
- char * gs2header
- char * cfmb_str
- char * sf_str
- char * snonce
- char * clientproof
- char storedkey [GSASL_HASH_MAX_SIZE]
- char serverkey [GSASL_HASH_MAX_SIZE]
- char * authmessage
- char * cb
- size_t cblen
- struct scram_client_first cf
- struct scram_server_first sf
- struct scram_client_final cl
- struct scram_server_final sl

### 4.26.1 Detailed Description

Definition at line 50 of file scram/server.c.

### 4.26.2 Field Documentation

#### 4.26.2.1 authmessage

```
char* scram_server_state::authmessage
```

Definition at line 63 of file scram/server.c.

#### 4.26.2.2 cb

```
char* scram_server_state::cb
```

Definition at line 64 of file scram/server.c.

#### 4.26.2.3 cbind

```
char* scram_server_state::cbind
```

Definition at line 55 of file scram/server.c.

#### 4.26.2.4 cblen

```
size_t scram_server_state::cblen
```

Definition at line 65 of file scram/server.c.

#### 4.26.2.5 cf

```
struct scram_client_first scram_server_state::cf
```

Definition at line 66 of file scram/server.c.

#### 4.26.2.6 cfmb_str

```
char* scram_server_state::cfmb_str
```

Definition at line 57 of file scram/server.c.

#### 4.26.2.7 cl

```
struct scram_client_final scram_server_state::cl
```

Definition at line 68 of file scram/server.c.

#### 4.26.2.8 clientproof

```
char* scram_server_state::clientproof
```

Definition at line 60 of file scram/server.c.

#### 4.26.2.9 gs2header

```
char* scram_server_state::gs2header
```

Definition at line 56 of file scram/server.c.

#### 4.26.2.10 hash

```
Gsasl_hash scram_server_state::hash
```

Definition at line 53 of file scram/server.c.

#### 4.26.2.11 plus

```
bool scram_server_state::plus
```

Definition at line 52 of file scram/server.c.

#### 4.26.2.12 serverkey

```
char scram_server_state::serverkey[GSASL_HASH_MAX_SIZE]
```

Definition at line 62 of file scram/server.c.

#### 4.26.2.13 sf

```
struct scram_server_first scram_server_state::sf
```

Definition at line 67 of file scram/server.c.

#### 4.26.2.14 sf_str

```
char* scram_server_state::sf_str
```

Definition at line 58 of file scram/server.c.

#### 4.26.2.15 sl

```
struct scram_server_final scram_server_state::sl
```

Definition at line 69 of file scram/server.c.

#### 4.26.2.16 snonce

```
char* scram_server_state::snonce
```

Definition at line 59 of file scram/server.c.

#### 4.26.2.17 step

```
int scram_server_state::step
```

Definition at line 54 of file scram/server.c.

#### 4.26.2.18 storedkey

```
char scram_server_state::storedkey[GSASL_HASH_MAX_SIZE]
```

Definition at line 61 of file scram/server.c.

The documentation for this struct was generated from the following file:

- scram/server.c

# Chapter 5

# File Documentation

## 5.1 anonymous.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_ANONYMOUS_NAME "ANONYMOUS"

### Functions

- int _gsasl_anonymous_client_step (Gsasl_session ∗sctx, void ∗mech_data, const char ∗input, size_t input↩ _len, char ∗∗output, size_t ∗output_len)
- int _gsasl_anonymous_server_step (Gsasl_session ∗sctx, void ∗mech_data, const char ∗input, size_t input↩ _len, char ∗∗output, size_t ∗output_len)

### Variables

- Gsasl_mechanism _gsasl_anonymous_mechanism

### 5.1.1 Macro Definition Documentation

#### 5.1.1.1 GSASL_ANONYMOUS_NAME

```
#define GSASL_ANONYMOUS_NAME "ANONYMOUS"
```

Definition at line 27 of file anonymous.h.

### 5.1.2 Function Documentation

#### 5.1.2.1 _gsasl_anonymous_client_step()

```
int _gsasl_anonymous_client_step (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )  [extern]
```

#### 5.1.2.2 _gsasl_anonymous_server_step()

```
int _gsasl_anonymous_server_step (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )  [extern]
```

### 5.1.3 Variable Documentation

#### 5.1.3.1 _gsasl_anonymous_mechanism

```
Gsasl_mechanism _gsasl_anonymous_mechanism  [extern]
```

Definition at line 27 of file anonymous/mechinfo.c.

## 5.2 anonymous.h

Go to the documentation of this file.

```
00001 /* anonymous.h --- Prototypes for ANONYMOUS mechanism as defined in RFC 2245.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef ANONYMOUS_H
00023 # define ANONYMOUS_H
00024
00025 # include <gsasl.h>
00026
```

```
00027 # define GSASL_ANONYMOUS_NAME "ANONYMOUS"
00028
00029 extern Gsasl_mechanism _gsasl_anonymous_mechanism;
00030
00031 extern int _gsasl_anonymous_client_step (Gsasl_session * sctx,
00032                                          void *mech_data,
00033                                          const char *input, size_t input_len,
00034                                          char **output, size_t *output_len);
00035
00036 extern int _gsasl_anonymous_server_step (Gsasl_session * sctx,
00037                                          void *mech_data,
00038                                          const char *input, size_t input_len,
00039                                          char **output, size_t *output_len);
00040
00041 #endif /* ANONYMOUS_H */
```

## 5.3 challenge.c File Reference

```
#include <config.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include "challenge.h"
#include <gc.h>
```

### Macros

- #define NONCELEN 10
- #define TEMPLATE "<XXXXXXXXXXXXXXXXXXXX.0@localhost>"
- #define DIGIT(c)

### Functions

- int cram_md5_challenge (char challenge[CRAM_MD5_CHALLENGE_LEN])

### 5.3.1 Macro Definition Documentation

#### 5.3.1.1 DIGIT

```
#define DIGIT(
              c )
```

**Value:**

```
(((c) & 0x0F) > 9 ?          \
'0' + ((c) & 0x0F) - 10 :    \
'0' + ((c) & 0x0F))
```

Definition at line 60 of file challenge.c.

#### 5.3.1.2 NONCELEN

```
#define NONCELEN 10
```

Definition at line 55 of file challenge.c.

#### 5.3.1.3 TEMPLATE

```
#define TEMPLATE "<XXXXXXXXXXXXXXXXXXXX.0@localhost>"
```

Definition at line 56 of file challenge.c.

### 5.3.2 Function Documentation

#### 5.3.2.1 cram_md5_challenge()

```
int cram_md5_challenge (
            char challenge[CRAM_MD5_CHALLENGE_LEN] )
```

Definition at line 65 of file challenge.c.

## 5.4 challenge.c

Go to the documentation of this file.

```
00001 /* challenge.c --- Generate a CRAM-MD5 challenge string.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 #include <stdio.h>
00025 #include <string.h>
00026 #include <assert.h>
00027
00028 /* Get prototype. */
00029 #include "challenge.h"
00030
00031 /* Get gc_nonce. */
00032 #include <gc.h>
00033
00034 /*
00035  * From draft-ietf-sasl-crammd5-02.txt:
00036  *
00037  *   The data encoded in the challenge contains a presumptively
00038  *   arbitrary string of random digits, a time-stamp, and the
00039  *   fully-qualified primary host name of the server.
00040  * ...
00041  *   challenge  = "<" 1*DIGIT "." 1*DIGIT "@" hostname ">"
00042  *   hostname   = 1*(ALPHA / DIGIT) *("." / "-" / ALPHA / DIGIT)
00043  *
00044  * This implementation avoid the information leakage by always using 0
00045  * as the time-stamp and a fixed host name.  This should be
00046  * unproblematic, as any client that try to validate the challenge
00047  * string somehow, would violate the same specification:
00048  *
00049  *   The client MUST NOT interpret or attempt to validate the
00050  *   contents of the challenge in any way.
00051  *
00052  */
```

```
00053
00054 /* The sequence of X in TEMPLATE must be twice as long as NONCELEN. */
00055 #define NONCELEN 10
00056 #define TEMPLATE "<XXXXXXXXXXXXXXXXXXXX.0@localhost>"
00057
00058 /* The probabilities for each digit are skewed (0-5 is more likely to
00059    occur than 6-9), but it is just used as a nonce anyway. */
00060 #define DIGIT(c) (((c) & 0x0F) > 9 ?            \
00061                   '0' + ((c) & 0x0F) - 10 :     \
00062                   '0' + ((c) & 0x0F))
00063
00064 int
00065 cram_md5_challenge (char challenge[CRAM_MD5_CHALLENGE_LEN])
00066 {
00067   char nonce[NONCELEN];
00068   size_t i;
00069   int rc;
00070
00071   assert (strlen (TEMPLATE) == CRAM_MD5_CHALLENGE_LEN - 1);
00072
00073   memcpy (challenge, TEMPLATE, CRAM_MD5_CHALLENGE_LEN);
00074
00075   rc = gc_nonce (nonce, sizeof (nonce));
00076   if (rc != GC_OK)
00077     return -1;
00078
00079   for (i = 0; i < sizeof (nonce); i++)
00080     {
00081       challenge[1 + i] = DIGIT (nonce[i]);
00082       challenge[11 + i] = DIGIT (nonce[i] » 4);
00083     }
00084
00085   return 0;
00086 }
```

# 5.5 challenge.h File Reference

### Macros

- #define CRAM_MD5_CHALLENGE_LEN 35

### Functions

- int cram_md5_challenge (char challenge[CRAM_MD5_CHALLENGE_LEN])

## 5.5.1 Macro Definition Documentation

### 5.5.1.1 CRAM_MD5_CHALLENGE_LEN

```
#define CRAM_MD5_CHALLENGE_LEN 35
```

Definition at line 25 of file challenge.h.

## 5.5.2 Function Documentation

### 5.5.2.1 cram_md5_challenge()

```
int cram_md5_challenge (
            char challenge[CRAM_MD5_CHALLENGE_LEN] )  [extern]
```

Definition at line 65 of file challenge.c.

## 5.6 challenge.h

Go to the documentation of this file.

```
00001 /* challenge.h --- Generate a CRAM-MD5 challenge string.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef CHALLENGE_H
00023 # define CHALLENGE_H
00024
00025 # define CRAM_MD5_CHALLENGE_LEN 35
00026
00027 /* Store zero terminated CRAM-MD5 challenge in output buffer.  The
00028    CHALLENGE buffer must be allocated by the caller, and must have
00029    room for CRAM_MD5_CHALLENGE_LEN characters.  Returns 0 on success,
00030    and -1 on randomness problems.  */
00031 extern int cram_md5_challenge (char challenge[CRAM_MD5_CHALLENGE_LEN]);
00032
00033 #endif /* CHALLENGE_H */
```

## 5.7 cram-md5.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_CRAM_MD5_NAME "CRAM-MD5"

### Functions

- int _gsasl_cram_md5_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input↩ _len, char **output, size_t *output_len)
- int _gsasl_cram_md5_server_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_cram_md5_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input↩ _len, char **output, size_t *output_len)
- void _gsasl_cram_md5_server_finish (Gsasl_session *sctx, void *mech_data)

### Variables

- Gsasl_mechanism _gsasl_cram_md5_mechanism

## 5.7.1 Macro Definition Documentation

### 5.7.1.1 GSASL_CRAM_MD5_NAME

```
#define GSASL_CRAM_MD5_NAME "CRAM-MD5"
```

Definition at line 27 of file cram-md5.h.

## 5.7.2 Function Documentation

### 5.7.2.1 _gsasl_cram_md5_client_step()

```
int _gsasl_cram_md5_client_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

### 5.7.2.2 _gsasl_cram_md5_server_finish()

```
void _gsasl_cram_md5_server_finish (
            Gsasl_session * sctx,
            void * mech_data )  [extern]
```

### 5.7.2.3 _gsasl_cram_md5_server_start()

```
int _gsasl_cram_md5_server_start (
            Gsasl_session * sctx,
            void ** mech_data )  [extern]
```

### 5.7.2.4 _gsasl_cram_md5_server_step()

```
int _gsasl_cram_md5_server_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

Definition at line 65 of file cram-md5/server.c.

## 5.7.3 Variable Documentation

### 5.7.3.1 _gsasl_cram_md5_mechanism

```
Gsasl_mechanism _gsasl_cram_md5_mechanism  [extern]
```

Definition at line 27 of file cram-md5/mechinfo.c.

## 5.8 cram-md5.h

Go to the documentation of this file.

```
00001 /* cram-md5.h --- Prototypes for CRAM-MD5 mechanism as defined in RFC 2195.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef CRAM_MD5_H
00023 # define CRAM_MD5_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_CRAM_MD5_NAME "CRAM-MD5"
00028
00029 extern Gsasl_mechanism _gsasl_cram_md5_mechanism;
00030
00031 extern int _gsasl_cram_md5_client_step (Gsasl_session * sctx,
00032                                         void *mech_data,
00033                                         const char *input, size_t input_len,
00034                                         char **output, size_t *output_len);
00035
00036 extern int _gsasl_cram_md5_server_start (Gsasl_session * sctx,
00037                                          void **mech_data);
00038 extern int _gsasl_cram_md5_server_step (Gsasl_session * sctx,
00039                                         void *mech_data,
00040                                         const char *input, size_t input_len,
00041                                         char **output, size_t *output_len);
00042 extern void _gsasl_cram_md5_server_finish (Gsasl_session * sctx,
00043                                            void *mech_data);
00044
00045 #endif /* CRAM_MD5_H */
```

## 5.9 digest.c File Reference

```
#include <config.h>
#include <string.h>
#include "digest.h"
#include "gc.h"
```

### Macros

- #define HEXCHAR(c) ((c & 0x0F) > 9 ? 'a' + (c & 0x0F) - 10 : '0' + (c & 0x0F))

### Functions

- void cram_md5_digest (const char *challenge, size_t challengelen, const char *secret, size_t secretlen, char response[CRAM_MD5_DIGEST_LEN])

### 5.9.1 Macro Definition Documentation

#### 5.9.1.1 HEXCHAR

```
#define HEXCHAR(
              c ) ((c & 0x0F) > 9 ?  'a' + (c & 0x0F) - 10 :  '0' + (c & 0x0F))
```

Definition at line 56 of file digest.c.

### 5.9.2 Function Documentation

#### 5.9.2.1 cram_md5_digest()

```
void cram_md5_digest (
              const char * challenge,
              size_t challengelen,
              const char * secret,
              size_t secretlen,
              char response[CRAM_MD5_DIGEST_LEN] )
```

Definition at line 59 of file digest.c.

## 5.10 digest.c

Go to the documentation of this file.

```
00001 /* digest.c --- Generate a CRAM-MD5 hex encoded HMAC-MD5 response string.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 #include <string.h>
00025
00026 /* Get prototype. */
00027 #include "digest.h"
00028
00029 /* Get gc_hmac_md5. */
00030 #include "gc.h"
00031
00032 /*
00033  * From draft-ietf-sasl-crammd5-02.txt:
00034  *
00035  *   The latter is computed by applying the keyed MD5 algorithm from
00036  *   [KEYED-MD5] where the key is a shared secret and the digested
00037  *   text is the challenge (including angle-brackets). The client
00038  *   MUST NOT interpret or attempt to validate the contents of the
00039  *   challenge in any way.
00040  *
00041  *   This shared secret is a string known only to the client and
```

```
00042  *   server.  The "digest" parameter itself is a 16-octet value which
00043  *   is sent in hexadecimal format, using lower-case US-ASCII
00044  *   characters.
00045  * ...
00046  *   digest     = 32(DIGIT / %x61-66)
00047  *   ; A hexadecimal string using only lower-case
00048  *   ; letters
00049  *
00050  */
00051
00052 #if CRAM_MD5_DIGEST_LEN != 2*GC_MD5_DIGEST_SIZE
00053 # error MD5 length mismatch
00054 #endif
00055
00056 #define HEXCHAR(c) ((c & 0x0F) > 9 ? 'a' + (c & 0x0F) - 10 : '0' + (c & 0x0F))
00057
00058 void
00059 cram_md5_digest (const char *challenge,
00060                  size_t challengelen,
00061                  const char *secret,
00062                  size_t secretlen, char response[CRAM_MD5_DIGEST_LEN])
00063 {
00064   char hash[GC_MD5_DIGEST_SIZE];
00065   size_t i;
00066
00067   gc_hmac_md5 (secret, secretlen ? secretlen : strlen (secret),
00068                challenge, challengelen ? challengelen : strlen (challenge),
00069                hash);
00070
00071   for (i = 0; i < GC_MD5_DIGEST_SIZE; i++)
00072     {
00073       *response++ = HEXCHAR (hash[i] >> 4);
00074       *response++ = HEXCHAR (hash[i]);
00075     }
00076 }
```

## 5.11 digest.h File Reference

```
#include <stddef.h>
```

### Macros

- #define CRAM_MD5_DIGEST_LEN 32

### Functions

- void cram_md5_digest (const char *challenge, size_t challengelen, const char *secret, size_t secretlen, char response[CRAM_MD5_DIGEST_LEN])

### 5.11.1 Macro Definition Documentation

#### 5.11.1.1 CRAM_MD5_DIGEST_LEN

```
#define CRAM_MD5_DIGEST_LEN 32
```

Definition at line 28 of file digest.h.

### 5.11.2 Function Documentation

#### 5.11.2.1 cram_md5_digest()

```
void cram_md5_digest (
            const char * challenge,
            size_t challengelen,
            const char * secret,
            size_t secretlen,
            char response[CRAM_MD5_DIGEST_LEN] )  [extern]
```

Definition at line 59 of file digest.c.

## 5.12 digest.h

Go to the documentation of this file.

```
00001 /* digest.h --- Generate a CRAM-MD5 hex encoded HMAC-MD5 response string.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef DIGEST_H
00023 # define DIGEST_H
00024
00025 /* Get size_t. */
00026 # include <stddef.h>
00027
00028 # define CRAM_MD5_DIGEST_LEN 32
00029
00030 /* Compute hex encoded HMAC-MD5 on the CHALLENGELEN long string
00031    CHALLENGE, keyed with SECRET of length SECRETLEN.  Use a
00032    CHALLENGELEN or SECRETLEN of 0 to indicate that CHALLENGE or
00033    SECRET, respectively, is zero terminated.  The RESPONSE buffer must
00034    be allocated by the caller, and must have room for
00035    CRAM_MD5_DIGEST_LEN characters.*/
00036 extern void cram_md5_digest (const char *challenge,
00037                              size_t challengelen,
00038                              const char *secret,
00039                              size_t secretlen,
00040                              char response[CRAM_MD5_DIGEST_LEN]);
00041
00042 #endif /* DIGEST_H */
```

## 5.13 digest-md5.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_DIGEST_MD5_NAME "DIGEST-MD5"

### Functions

- int _gsasl_digest_md5_client_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_digest_md5_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_digest_md5_client_finish (Gsasl_session *sctx, void *mech_data)
- int _gsasl_digest_md5_client_encode (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- int _gsasl_digest_md5_client_decode (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- int _gsasl_digest_md5_server_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_digest_md5_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_digest_md5_server_finish (Gsasl_session *sctx, void *mech_data)
- int _gsasl_digest_md5_server_encode (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- int _gsasl_digest_md5_server_decode (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)

### Variables

- Gsasl_mechanism _gsasl_digest_md5_mechanism

## 5.13.1 Macro Definition Documentation

### 5.13.1.1 GSASL_DIGEST_MD5_NAME

```
#define GSASL_DIGEST_MD5_NAME "DIGEST-MD5"
```

Definition at line 27 of file digest-md5.h.

## 5.13.2 Function Documentation

### 5.13.2.1 _gsasl_digest_md5_client_decode()

```
int _gsasl_digest_md5_client_decode (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

#### 5.13.2.2 _gsasl_digest_md5_client_encode()

```
int _gsasl_digest_md5_client_encode (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

#### 5.13.2.3 _gsasl_digest_md5_client_finish()

```
void _gsasl_digest_md5_client_finish (
            Gsasl_session * sctx,
            void * mech_data )  [extern]
```

#### 5.13.2.4 _gsasl_digest_md5_client_start()

```
int _gsasl_digest_md5_client_start (
            Gsasl_session * sctx,
            void ** mech_data )  [extern]
```

#### 5.13.2.5 _gsasl_digest_md5_client_step()

```
int _gsasl_digest_md5_client_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

Definition at line 97 of file digest-md5/client.c.

#### 5.13.2.6 _gsasl_digest_md5_server_decode()

```
int _gsasl_digest_md5_server_decode (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

#### 5.13.2.7 _gsasl_digest_md5_server_encode()

```
int _gsasl_digest_md5_server_encode (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

#### 5.13.2.8 _gsasl_digest_md5_server_finish()

```
void _gsasl_digest_md5_server_finish (
             Gsasl_session * sctx,
             void * mech_data )  [extern]
```

#### 5.13.2.9 _gsasl_digest_md5_server_start()

```
int _gsasl_digest_md5_server_start (
             Gsasl_session * sctx,
             void ** mech_data )  [extern]
```

#### 5.13.2.10 _gsasl_digest_md5_server_step()

```
int _gsasl_digest_md5_server_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

Definition at line 145 of file digest-md5/server.c.

### 5.13.3 Variable Documentation

#### 5.13.3.1 _gsasl_digest_md5_mechanism

```
Gsasl_mechanism _gsasl_digest_md5_mechanism  [extern]
```

Definition at line 27 of file digest-md5/mechinfo.c.

## 5.14 digest-md5.h

Go to the documentation of this file.

```
00001 /* digest-md5.h --- Prototypes for DIGEST-MD5 mechanism as defined in RFC 2831.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
```

```
00022 #ifndef DIGEST_MD5_H
00023 # define DIGEST_MD5_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_DIGEST_MD5_NAME "DIGEST-MD5"
00028
00029 extern Gsasl_mechanism _gsasl_digest_md5_mechanism;
00030
00031 extern int _gsasl_digest_md5_client_start (Gsasl_session * sctx,
00032                                            void **mech_data);
00033 extern int _gsasl_digest_md5_client_step (Gsasl_session * sctx,
00034                                           void *mech_data,
00035                                           const char *input, size_t input_len,
00036                                           char **output, size_t *output_len);
00037 extern void _gsasl_digest_md5_client_finish (Gsasl_session * sctx,
00038                                              void *mech_data);
00039 extern int _gsasl_digest_md5_client_encode (Gsasl_session * sctx,
00040                                             void *mech_data,
00041                                             const char *input,
00042                                             size_t input_len,
00043                                             char **output,
00044                                             size_t *output_len);
00045 extern int _gsasl_digest_md5_client_decode (Gsasl_session * sctx,
00046                                             void *mech_data,
00047                                             const char *input,
00048                                             size_t input_len,
00049                                             char **output,
00050                                             size_t *output_len);
00051
00052 extern int _gsasl_digest_md5_server_start (Gsasl_session * sctx,
00053                                            void **mech_data);
00054 extern int _gsasl_digest_md5_server_step (Gsasl_session * sctx,
00055                                           void *mech_data,
00056                                           const char *input, size_t input_len,
00057                                           char **output, size_t *output_len);
00058 extern void _gsasl_digest_md5_server_finish (Gsasl_session * sctx,
00059                                              void *mech_data);
00060 extern int _gsasl_digest_md5_server_encode (Gsasl_session * sctx,
00061                                             void *mech_data,
00062                                             const char *input,
00063                                             size_t input_len,
00064                                             char **output,
00065                                             size_t *output_len);
00066 extern int _gsasl_digest_md5_server_decode (Gsasl_session * sctx,
00067                                             void *mech_data,
00068                                             const char *input,
00069                                             size_t input_len,
00070                                             char **output,
00071                                             size_t *output_len);
00072
00073 #endif /* DIGEST_MD5_H */
```

## 5.15 digesthmac.c File Reference

```
#include <config.h>
#include "digesthmac.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <gc.h>
```

### Macros

- #define HEXCHAR(c) ((c & 0x0F) > 9 ? 'a' + (c & 0x0F) - 10 : '0' + (c & 0x0F))
- #define QOP_AUTH "auth"
- #define QOP_AUTH_INT "auth-int"
- #define QOP_AUTH_CONF "auth-conf"
- #define A2_PRE "AUTHENTICATE:"
- #define A2_POST ":00000000000000000000000000000000"

- #define COLON ":"
- #define MD5LEN 16
- #define DERIVE_CLIENT_INTEGRITY_KEY_STRING "Digest session key to client-to-server signing key magic constant"
- #define DERIVE_CLIENT_INTEGRITY_KEY_STRING_LEN 65
- #define DERIVE_SERVER_INTEGRITY_KEY_STRING "Digest session key to server-to-client signing key magic constant"
- #define DERIVE_SERVER_INTEGRITY_KEY_STRING_LEN 65
- #define DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING "Digest H(A1) to client-to-server sealing key magic constant"
- #define DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING_LEN 59
- #define DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING "Digest H(A1) to server-to-client sealing key magic constant"
- #define DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING_LEN 59

**Functions**

- int digest_md5_hmac (char *output, char secret[MD5LEN], const char *nonce, unsigned long nc, const char *cnonce, digest_md5_qop qop, const char *authzid, const char *digesturi, int rspauth, digest_md5_cipher cipher, char *kic, char *kis, char *kcc, char *kcs)

## 5.15.1 Macro Definition Documentation

### 5.15.1.1 A2_POST

```
#define A2_POST ":00000000000000000000000000000000"
```

Definition at line 46 of file digesthmac.c.

### 5.15.1.2 A2_PRE

```
#define A2_PRE "AUTHENTICATE:"
```

Definition at line 45 of file digesthmac.c.

### 5.15.1.3 COLON

```
#define COLON ":"
```

Definition at line 47 of file digesthmac.c.

### 5.15.1.4 DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING

```
#define DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING  "Digest H(A1) to client-to-server sealing
key magic constant"
```

Definition at line 55 of file digesthmac.c.

#### 5.15.1.5 DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING_LEN

```
#define DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING_LEN 59
```

Definition at line 57 of file digesthmac.c.

#### 5.15.1.6 DERIVE_CLIENT_INTEGRITY_KEY_STRING

```
#define DERIVE_CLIENT_INTEGRITY_KEY_STRING  "Digest session key to client-to-server signing
key magic constant"
```

Definition at line 49 of file digesthmac.c.

#### 5.15.1.7 DERIVE_CLIENT_INTEGRITY_KEY_STRING_LEN

```
#define DERIVE_CLIENT_INTEGRITY_KEY_STRING_LEN 65
```

Definition at line 51 of file digesthmac.c.

#### 5.15.1.8 DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING

```
#define DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING  "Digest H(A1) to server-to-client sealing
key magic constant"
```

Definition at line 58 of file digesthmac.c.

#### 5.15.1.9 DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING_LEN

```
#define DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING_LEN 59
```

Definition at line 60 of file digesthmac.c.

#### 5.15.1.10 DERIVE_SERVER_INTEGRITY_KEY_STRING

```
#define DERIVE_SERVER_INTEGRITY_KEY_STRING  "Digest session key to server-to-client signing
key magic constant"
```

Definition at line 52 of file digesthmac.c.

#### 5.15.1.11 DERIVE_SERVER_INTEGRITY_KEY_STRING_LEN

```
#define DERIVE_SERVER_INTEGRITY_KEY_STRING_LEN 65
```

Definition at line 54 of file digesthmac.c.

#### 5.15.1.12 HEXCHAR

```
#define HEXCHAR(
             c ) ((c & 0x0F) > 9 ?  'a' + (c & 0x0F) - 10 :  '0' + (c & 0x0F))
```

Definition at line 39 of file digesthmac.c.

#### 5.15.1.13 MD5LEN

```
#define MD5LEN 16
```

Definition at line 48 of file digesthmac.c.

#### 5.15.1.14 QOP_AUTH

```
#define QOP_AUTH "auth"
```

Definition at line 41 of file digesthmac.c.

#### 5.15.1.15 QOP_AUTH_CONF

```
#define QOP_AUTH_CONF "auth-conf"
```

Definition at line 43 of file digesthmac.c.

#### 5.15.1.16 QOP_AUTH_INT

```
#define QOP_AUTH_INT "auth-int"
```

Definition at line 42 of file digesthmac.c.

### 5.15.2 Function Documentation

#### 5.15.2.1 digest_md5_hmac()

```
int digest_md5_hmac (
             char * output,
             char secret[MD5LEN],
             const char * nonce,
             unsigned long nc,
             const char * cnonce,
             digest_md5_qop qop,
             const char * authzid,
             const char * digesturi,
             int rspauth,
             digest_md5_cipher cipher,
             char * kic,
             char * kis,
             char * kcc,
             char * kcs )
```

Definition at line 76 of file digesthmac.c.

# 5.16 digesthmac.c

Go to the documentation of this file.

```
00001 /* digesthmac.c --- Compute DIGEST-MD5 response value.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "digesthmac.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get memcpy, strlen. */
00031 #include <string.h>
00032
00033 /* Get sprintf. */
00034 #include <stdio.h>
00035
00036 /* Get gc_md5. */
00037 #include <gc.h>
00038
00039 #define HEXCHAR(c) ((c & 0x0F) > 9 ? 'a' + (c & 0x0F) - 10 : '0' + (c & 0x0F))
00040
00041 #define QOP_AUTH "auth"
00042 #define QOP_AUTH_INT "auth-int"
00043 #define QOP_AUTH_CONF "auth-conf"
00044
00045 #define A2_PRE "AUTHENTICATE:"
00046 #define A2_POST ":00000000000000000000000000000000"
00047 #define COLON ":"
00048 #define MD5LEN 16
00049 #define DERIVE_CLIENT_INTEGRITY_KEY_STRING \
00050   "Digest session key to client-to-server signing key magic constant"
00051 #define DERIVE_CLIENT_INTEGRITY_KEY_STRING_LEN 65
00052 #define DERIVE_SERVER_INTEGRITY_KEY_STRING \
00053   "Digest session key to server-to-client signing key magic constant"
00054 #define DERIVE_SERVER_INTEGRITY_KEY_STRING_LEN 65
00055 #define DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING \
00056   "Digest H(A1) to client-to-server sealing key magic constant"
00057 #define DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING_LEN 59
00058 #define DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING \
00059   "Digest H(A1) to server-to-client sealing key magic constant"
00060 #define DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING_LEN 59
00061
00062 /* Compute in 33 bytes large array OUTPUT the DIGEST-MD5 response
00063    value.  SECRET holds the 16 bytes MD5 hash SS, i.e.,
00064    H(username:realm:passwd).  NONCE is a zero terminated string with
00065    the server nonce.  NC is the nonce-count, typically 1 for initial
00066    authentication.  CNONCE is a zero terminated string with the client
00067    nonce.  QOP is the quality of protection to use.  AUTHZID is a zero
00068    terminated string with the authorization identity.  DIGESTURI is a
00069    zero terminated string with the server principal (e.g.,
00070    imap/mail.example.org).  RSPAUTH is a boolean which indicate
00071    whether to compute a value for the RSPAUTH response or the "real"
00072    authentication.  CIPHER is the cipher to use.  KIC, KIS, KCC, KCS
00073    are either NULL, or points to 16 byte arrays that will hold the
00074    computed keys on output.  Returns 0 on success. */
00075 int
00076 digest_md5_hmac (char *output, char secret[MD5LEN], const char *nonce,
00077                  unsigned long nc, const char *cnonce, digest_md5_qop qop,
00078                  const char *authzid, const char *digesturi, int rspauth,
00079                  digest_md5_cipher cipher,
00080                  char *kic, char *kis, char *kcc, char *kcs)
00081 {
00082   const char *a2string = rspauth ? COLON : A2_PRE;
```

```
00083   char nchex[17];               /* really 9 but 17 for -Werror=format-overflow= */
00084   char a1hexhash[2 * MD5LEN];
00085   char a2hexhash[2 * MD5LEN];
00086   char hash[MD5LEN];
00087   char *tmp, *p;
00088   size_t tmplen;
00089   int rc;
00090   int i;
00091
00092   /* A1 */
00093
00094   tmplen = MD5LEN + strlen (COLON) + strlen (nonce) +
00095     strlen (COLON) + strlen (cnonce);
00096   if (authzid && strlen (authzid) > 0)
00097     tmplen += strlen (COLON) + strlen (authzid);
00098
00099   p = tmp = malloc (tmplen);
00100   if (tmp == NULL)
00101     return -1;
00102
00103   memcpy (p, secret, MD5LEN);
00104   p += MD5LEN;
00105   memcpy (p, COLON, strlen (COLON));
00106   p += strlen (COLON);
00107   memcpy (p, nonce, strlen (nonce));
00108   p += strlen (nonce);
00109   memcpy (p, COLON, strlen (COLON));
00110   p += strlen (COLON);
00111   memcpy (p, cnonce, strlen (cnonce));
00112   p += strlen (cnonce);
00113   if (authzid && strlen (authzid) > 0)
00114     {
00115       memcpy (p, COLON, strlen (COLON));
00116       p += strlen (COLON);
00117       memcpy (p, authzid, strlen (authzid));
00118     }
00119
00120   rc = gc_md5 (tmp, tmplen, hash);
00121   free (tmp);
00122   if (rc)
00123     return rc;
00124
00125   if (kic)
00126     {
00127       char hash2[MD5LEN];
00128       char q[MD5LEN + DERIVE_CLIENT_INTEGRITY_KEY_STRING_LEN];
00129       size_t qlen = MD5LEN + DERIVE_CLIENT_INTEGRITY_KEY_STRING_LEN;
00130
00131       memcpy (q, hash, MD5LEN);
00132       memcpy (q + MD5LEN, DERIVE_CLIENT_INTEGRITY_KEY_STRING,
00133               DERIVE_CLIENT_INTEGRITY_KEY_STRING_LEN);
00134
00135       rc = gc_md5 (q, qlen, hash2);
00136       if (rc)
00137         return rc;
00138
00139       memcpy (kic, hash2, MD5LEN);
00140     }
00141
00142   if (kis)
00143     {
00144       char hash2[MD5LEN];
00145       char q[MD5LEN + DERIVE_SERVER_INTEGRITY_KEY_STRING_LEN];
00146       size_t qlen = MD5LEN + DERIVE_SERVER_INTEGRITY_KEY_STRING_LEN;
00147
00148       memcpy (q, hash, MD5LEN);
00149       memcpy (q + MD5LEN, DERIVE_SERVER_INTEGRITY_KEY_STRING,
00150               DERIVE_SERVER_INTEGRITY_KEY_STRING_LEN);
00151
00152       rc = gc_md5 (q, qlen, hash2);
00153       if (rc)
00154         return rc;
00155
00156       memcpy (kis, hash2, MD5LEN);
00157     }
00158
00159   if (kcc)
00160     {
00161       char hash2[MD5LEN];
00162       int n;
00163       char q[MD5LEN + DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING_LEN];
00164
00165       if (cipher == DIGEST_MD5_CIPHER_RC4_40)
00166         n = 5;
00167       else if (cipher == DIGEST_MD5_CIPHER_RC4_56)
00168         n = 7;
00169       else
```

```
00170         n = MD5LEN;
00171
00172       memcpy (q, hash, n);
00173       memcpy (q + n, DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING,
00174               DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING_LEN);
00175
00176       rc = gc_md5 (q, n + DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING_LEN,
00177                    hash2);
00178       if (rc)
00179         return rc;
00180
00181       memcpy (kcc, hash2, MD5LEN);
00182     }
00183
00184   if (kcs)
00185     {
00186       char hash2[MD5LEN];
00187       int n;
00188       char q[MD5LEN + DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING_LEN];
00189
00190       if (cipher == DIGEST_MD5_CIPHER_RC4_40)
00191         n = 5;
00192       else if (cipher == DIGEST_MD5_CIPHER_RC4_56)
00193         n = 7;
00194       else
00195         n = MD5LEN;
00196
00197       memcpy (q, hash, n);
00198       memcpy (q + n, DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING,
00199               DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING_LEN);
00200
00201       rc = gc_md5 (q, n + DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING_LEN,
00202                    hash2);
00203       if (rc)
00204         return rc;
00205
00206       memcpy (kcs, hash2, MD5LEN);
00207     }
00208
00209   for (i = 0; i < MD5LEN; i++)
00210     {
00211       a1hexhash[2 * i + 1] = HEXCHAR (hash[i]);
00212       a1hexhash[2 * i + 0] = HEXCHAR (hash[i] >> 4);
00213     }
00214
00215   /* A2 */
00216
00217   tmplen = strlen (a2string) + strlen (digesturi);
00218   if (qop & DIGEST_MD5_QOP_AUTH_INT || qop & DIGEST_MD5_QOP_AUTH_CONF)
00219     tmplen += strlen (A2_POST);
00220
00221   p = tmp = malloc (tmplen);
00222   if (tmp == NULL)
00223     return -1;
00224
00225   memcpy (p, a2string, strlen (a2string));
00226   p += strlen (a2string);
00227   memcpy (p, digesturi, strlen (digesturi));
00228   p += strlen (digesturi);
00229   if (qop & DIGEST_MD5_QOP_AUTH_INT || qop & DIGEST_MD5_QOP_AUTH_CONF)
00230     memcpy (p, A2_POST, strlen (A2_POST));
00231
00232   rc = gc_md5 (tmp, tmplen, hash);
00233   free (tmp);
00234   if (rc)
00235     return rc;
00236
00237   for (i = 0; i < MD5LEN; i++)
00238     {
00239       a2hexhash[2 * i + 1] = HEXCHAR (hash[i]);
00240       a2hexhash[2 * i + 0] = HEXCHAR (hash[i] >> 4);
00241     }
00242
00243   /* response_value */
00244
00245   sprintf (nchex, "%08lx", nc);
00246
00247   tmplen = 2 * MD5LEN + strlen (COLON) + strlen (nonce) + strlen (COLON) +
00248     strlen (nchex) + strlen (COLON) + strlen (cnonce) + strlen (COLON);
00249   if (qop & DIGEST_MD5_QOP_AUTH_CONF)
00250     tmplen += strlen (QOP_AUTH_CONF);
00251   else if (qop & DIGEST_MD5_QOP_AUTH_INT)
00252     tmplen += strlen (QOP_AUTH_INT);
00253   else if (qop & DIGEST_MD5_QOP_AUTH)
00254     tmplen += strlen (QOP_AUTH);
00255   tmplen += strlen (COLON) + 2 * MD5LEN;
00256
```

```
00257   p = tmp = malloc (tmplen);
00258   if (tmp == NULL)
00259     return -1;
00260
00261   memcpy (p, a1hexhash, 2 * MD5LEN);
00262   p += 2 * MD5LEN;
00263   memcpy (p, COLON, strlen (COLON));
00264   p += strlen (COLON);
00265   memcpy (p, nonce, strlen (nonce));
00266   p += strlen (nonce);
00267   memcpy (p, COLON, strlen (COLON));
00268   p += strlen (COLON);
00269   memcpy (p, nchex, strlen (nchex));
00270   p += strlen (nchex);
00271   memcpy (p, COLON, strlen (COLON));
00272   p += strlen (COLON);
00273   memcpy (p, cnonce, strlen (cnonce));
00274   p += strlen (cnonce);
00275   memcpy (p, COLON, strlen (COLON));
00276   p += strlen (COLON);
00277   if (qop & DIGEST_MD5_QOP_AUTH_CONF)
00278     {
00279       memcpy (p, QOP_AUTH_CONF, strlen (QOP_AUTH_CONF));
00280       p += strlen (QOP_AUTH_CONF);
00281     }
00282   else if (qop & DIGEST_MD5_QOP_AUTH_INT)
00283     {
00284       memcpy (p, QOP_AUTH_INT, strlen (QOP_AUTH_INT));
00285       p += strlen (QOP_AUTH_INT);
00286     }
00287   else if (qop & DIGEST_MD5_QOP_AUTH)
00288     {
00289       memcpy (p, QOP_AUTH, strlen (QOP_AUTH));
00290       p += strlen (QOP_AUTH);
00291     }
00292   memcpy (p, COLON, strlen (COLON));
00293   p += strlen (COLON);
00294   memcpy (p, a2hexhash, 2 * MD5LEN);
00295
00296   rc = gc_md5 (tmp, tmplen, hash);
00297   free (tmp);
00298   if (rc)
00299     return rc;
00300
00301   for (i = 0; i < MD5LEN; i++)
00302     {
00303       output[2 * i + 1] = HEXCHAR (hash[i]);
00304       output[2 * i + 0] = HEXCHAR (hash[i] >> 4);
00305     }
00306   output[32] = '\0';
00307
00308   return 0;
00309 }
```

## 5.17 digesthmac.h File Reference

```
#include "tokens.h"
```

### Functions

- int digest_md5_hmac (char *output, char secret[DIGEST_MD5_LENGTH], const char *nonce, unsigned long nc, const char *cnonce, digest_md5_qop qop, const char *authzid, const char *digesturi, int rspauth, digest_md5_cipher cipher, char *kic, char *kis, char *kcc, char *kcs)

### 5.17.1 Function Documentation

#### 5.17.1.1 digest_md5_hmac()

```
int digest_md5_hmac (
             char * output,
```

```
            char secret[DIGEST_MD5_LENGTH],
            const char * nonce,
            unsigned long nc,
            const char * cnonce,
            digest_md5_qop qop,
            const char * authzid,
            const char * digesturi,
            int rspauth,
            digest_md5_cipher cipher,
            char * kic,
            char * kis,
            char * kcc,
            char * kcs )  [extern]
```

## 5.18 digesthmac.h

Go to the documentation of this file.

```
00001 /* digesthmac.h --- Compute DIGEST-MD5 response value.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef DIGEST_MD5_DIGESTHMAC_H
00023 # define DIGEST_MD5_DIGESTHMAC_H
00024
00025 /* Get token types. */
00026 # include "tokens.h"
00027
00028 /* Compute in 33 bytes large array OUTPUT the DIGEST-MD5 response
00029    value.  SECRET holds the 16 bytes MD5 hash SS, i.e.,
00030    H(username:realm:passwd).  NONCE is a zero terminated string with
00031    the server nonce.  NC is the nonce-count, typically 1 for initial
00032    authentication.  CNONCE is a zero terminated string with the client
00033    nonce.  QOP is the quality of protection to use.  AUTHZID is a zero
00034    terminated string with the authorization identity.  DIGESTURI is a
00035    zero terminated string with the server principal (e.g.,
00036    imap/mail.example.org).  RSPAUTH is a boolean which indicate
00037    whether to compute a value for the RSPAUTH response or the "real"
00038    authentication.  CIPHER is the cipher to use.  KIC, KIS, KCC, KCS
00039    are either NULL, or points to 16 byte arrays that will hold the
00040    computed keys on output.  Returns 0 on success. */
00041 extern int digest_md5_hmac (char *output, char secret[DIGEST_MD5_LENGTH],
00042                             const char *nonce, unsigned long nc,
00043                             const char *cnonce, digest_md5_qop qop,
00044                             const char *authzid,
00045                             const char *digesturi, int rspauth,
00046                             digest_md5_cipher cipher, char *kic, char *kis,
00047                             char *kcc, char *kcs);
00048
00049 #endif /* DIGEST_MD5_DIGESTHMAC_H */
```

## 5.19 free.h File Reference

```
#include "tokens.h"
```

### Functions

- void digest_md5_free_challenge (digest_md5_challenge *c)
- void digest_md5_free_response (digest_md5_response *r)
- void digest_md5_free_finish (digest_md5_finish *f)

## 5.19.1 Function Documentation

### 5.19.1.1 digest_md5_free_challenge()

```
void digest_md5_free_challenge (
            digest_md5_challenge * c )  [extern]
```

Definition at line 34 of file digest-md5/free.c.

### 5.19.1.2 digest_md5_free_finish()

```
void digest_md5_free_finish (
            digest_md5_finish * f )  [extern]
```

Definition at line 60 of file digest-md5/free.c.

### 5.19.1.3 digest_md5_free_response()

```
void digest_md5_free_response (
            digest_md5_response * r )  [extern]
```

Definition at line 47 of file digest-md5/free.c.

# 5.20 free.h

Go to the documentation of this file.

```
00001 /* free.h --- Free allocated data in DIGEST-MD5 token structures.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef DIGEST_MD5_FREE_H
00023 # define DIGEST_MD5_FREE_H
00024
00025 /* Get token types. */
00026 # include "tokens.h"
00027
00028 extern void digest_md5_free_challenge (digest_md5_challenge * c);
00029
00030 extern void digest_md5_free_response (digest_md5_response * r);
00031
00032 extern void digest_md5_free_finish (digest_md5_finish * f);
00033
00034 #endif /* DIGEST_MD5_FREE_H */
```

# 5.21 getsubopt.c File Reference

```
#include <config.h>
#include "parser.h"
#include <string.h>
```

### Functions

- int digest_md5_getsubopt (char **optionp, const char *const *tokens, char **valuep)

## 5.21.1 Function Documentation

### 5.21.1.1 digest_md5_getsubopt()

```
int digest_md5_getsubopt (
            char ** optionp,
            const char *const * tokens,
            char ** valuep )
```

Definition at line 43 of file getsubopt.c.

# 5.22 getsubopt.c

Go to the documentation of this file.

```
00001 /* getsubopt.c --- Parse comma separate list into words, DIGEST-MD5 style.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  * Copyright (C) 1996, 1997, 1999 Free Software Foundation, Inc.
00004  * From the GNU C Library, under GNU LGPL version 2.1.
00005  * Contributed by Ulrich Drepper <drepper@cygnus.com>, 1996.
00006  * Modified for Libgsasl by Simon Josefsson <simon@josefsson.org>
00007  *
00008  * This file is part of GNU SASL Library.
00009  *
00010  * GNU SASL Library is free software; you can redistribute it and/or
00011  * modify it under the terms of the GNU Lesser General Public License
00012  * as published by the Free Software Foundation; either version 2.1 of
00013  * the License, or (at your option) any later version.
00014  *
00015  * GNU SASL Library is distributed in the hope that it will be useful,
00016  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00017  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00018  * Lesser General Public License for more details.
00019  *
00020  * You should have received a copy of the GNU Lesser General Public
00021  * License along with GNU SASL Library; if not, see
00022  * <https://www.gnu.org/licenses/>.
00023  *
00024  */
00025
00026 #include <config.h>
00027
00028 /* Get prototypes. */
00029 #include "parser.h"
00030
00031 /* Get memchr and memcmp. */
00032 #include <string.h>
00033
00034 /* Parse comma separated suboption from *OPTIONP and match against
00035    strings in TOKENS.  If found return index and set *VALUEP to
00036    optional value introduced by an equal sign.  If the suboption is
00037    not part of TOKENS return in *VALUEP beginning of unknown
00038    suboption.  On exit *OPTIONP is set to the beginning of the next
00039    token or at the terminating NUL character.
```

```
00040
00041    This function is NOT identical to standard getsubopt! */
00042 int
00043 digest_md5_getsubopt (char **optionp,
00044                       const char *const *tokens, char **valuep)
00045 {
00046   char *endp, *vstart;
00047   int cnt;
00048   int inside_quote = 0;
00049
00050   if (**optionp == '\0')
00051     return -1;
00052
00053   /* Find end of next token.  */
00054   endp = *optionp;
00055   while (*endp != '\0' && (inside_quote || (!inside_quote && *endp != ',')))
00056     {
00057       if (*endp == '"')
00058         inside_quote = !inside_quote;
00059       endp++;
00060     }
00061
00062   /* Find start of value.  */
00063   vstart = memchr (*optionp, '=', endp - *optionp);
00064   if (vstart == NULL)
00065     vstart = endp;
00066
00067   /* Try to match the characters between *OPTIONP and VSTART against
00068      one of the TOKENS.  */
00069   for (cnt = 0; tokens[cnt] != NULL; ++cnt)
00070     if (strncmp (*optionp, tokens[cnt], vstart - *optionp) == 0
00071         && tokens[cnt][vstart - *optionp] == '\0')
00072       {
00073         /* We found the current option in TOKENS.  */
00074         *valuep = vstart != endp ? vstart + 1 : (char *) "";
00075
00076         while (*valuep && (**valuep == ' ' ||
00077                            **valuep == '\t' ||
00078                            **valuep == '\r' ||
00079                            **valuep == '\n' || **valuep == '"'))
00080           (*valuep)++;
00081
00082         if (*endp != '\0')
00083           {
00084             *endp = '\0';
00085             *optionp = endp + 1;
00086           }
00087         else
00088           *optionp = endp;
00089         endp--;
00090         while (*endp == ' ' ||
00091                *endp == '\t' ||
00092                *endp == '\r' || *endp == '\n' || *endp == '"')
00093           *endp-- = '\0';
00094         while (**optionp == ' ' ||
00095                **optionp == '\t' || **optionp == '\r' || **optionp == '\n')
00096           (*optionp)++;
00097
00098         return cnt;
00099       }
00100
00101   /* The current suboption does not match any option.  */
00102   *valuep = *optionp;
00103
00104   if (*endp != '\0')
00105     *endp++ = '\0';
00106   *optionp = endp;
00107   while (**optionp == ' ' ||
00108          **optionp == '\t' || **optionp == '\r' || **optionp == '\n')
00109     (*optionp)++;
00110
00111   return -1;
00112 }
```

## 5.23 nonascii.c File Reference

```
#include <config.h>
#include "nonascii.h"
#include <stdlib.h>
#include <string.h>
```

### Functions

- char * latin1toutf8 (const char *str)
- char * utf8tolatin1ifpossible (const char *passwd)

## 5.23.1 Function Documentation

### 5.23.1.1 latin1toutf8()

```
char * latin1toutf8 (
            const char * str )
```

Definition at line 38 of file nonascii.c.

### 5.23.1.2 utf8tolatin1ifpossible()

```
char * utf8tolatin1ifpossible (
            const char * passwd )
```

Definition at line 66 of file nonascii.c.

# 5.24 nonascii.c

Go to the documentation of this file.

```
00001 /* server.c --- DIGEST-MD5 mechanism from RFC 2831, server side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "nonascii.h"
00026
00027 #include <stdlib.h>
00028 #include <string.h>
00029
00030 /* C89 compliant way to cast 'char' to 'unsigned char'. */
00031 static inline unsigned char
00032 to_uchar (char ch)
00033 {
00034   return ch;
00035 }
00036
00037 char *
00038 latin1toutf8 (const char *str)
00039 {
00040   char *p = malloc (2 * strlen (str) + 1);
00041   if (p)
```

```
00042     {
00043       size_t i, j = 0;
00044       for (i = 0; str[i]; i++)
00045         {
00046           if (to_uchar (str[i]) < 0x80)
00047             p[j++] = str[i];
00048           else if (to_uchar (str[i]) < 0xC0)
00049             {
00050               p[j++] = (unsigned char) 0xC2;
00051               p[j++] = str[i];
00052             }
00053           else
00054             {
00055               p[j++] = (unsigned char) 0xC3;
00056               p[j++] = str[i] - 64;
00057             }
00058         }
00059       p[j] = 0x00;
00060     }
00061
00062   return p;
00063 }
00064
00065 char *
00066 utf8tolatin1ifpossible (const char *passwd)
00067 {
00068   char *p;
00069   size_t i;
00070
00071   for (i = 0; passwd[i]; i++)
00072     {
00073       if (to_uchar (passwd[i]) > 0x7F)
00074         {
00075           if (to_uchar (passwd[i]) < 0xC0 || to_uchar (passwd[i]) > 0xC3)
00076             return strdup (passwd);
00077           i++;
00078           if (to_uchar (passwd[i]) < 0x80 || to_uchar (passwd[i]) > 0xBF)
00079             return strdup (passwd);
00080         }
00081     }
00082
00083   p = malloc (strlen (passwd) + 1);
00084   if (p)
00085     {
00086       size_t j = 0;
00087       for (i = 0; passwd[i]; i++)
00088         {
00089           if (to_uchar (passwd[i]) > 0x7F)
00090             {
00091               /* p[i+1] can't be zero here */
00092               p[j++] =
00093                 ((to_uchar (passwd[i]) & 0x3) << 6)
00094                 | (to_uchar (passwd[i + 1]) & 0x3F);
00095               i++;
00096             }
00097           else
00098             p[j++] = passwd[i];
00099         }
00100       p[j] = 0x00;
00101     }
00102   return p;
00103 }
```

## 5.25 nonascii.h File Reference

### Functions

- char * latin1toutf8 (const char *str)
- char * utf8tolatin1ifpossible (const char *passwd)

### 5.25.1 Function Documentation

#### 5.25.1.1 latin1toutf8()

```
char * latin1toutf8 (
            const char * str )  [extern]
```

Definition at line 38 of file nonascii.c.

#### 5.25.1.2 utf8tolatin1ifpossible()

```
char * utf8tolatin1ifpossible (
            const char * passwd )  [extern]
```

Definition at line 66 of file nonascii.c.

## 5.26 nonascii.h

Go to the documentation of this file.

```
00001 /* nonascii.h --- Prototypes for UTF-8 vs Latin-1 conversion for DIGEST-MD5
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef DIGEST_MD5_NONASCII_H
00023 # define DIGEST_MD5_NONASCII_H
00024
00025 extern char *latin1toutf8 (const char *str);
00026
00027 extern char *utf8tolatin1ifpossible (const char *passwd);
00028
00029 #endif /* DIGEST_MD5_NONASCII_H */
```

## 5.27 qop.c File Reference

```
#include <config.h>
#include "qop.h"
#include "tokens.h"
#include "parser.h"
#include <string.h>
#include <stdlib.h>
```

### Functions

- int digest_md5_qopstr2qops (const char *qopstr)
- const char * digest_md5_qops2qopstr (int qops)

### 5.27.1 Function Documentation

#### 5.27.1.1 digest_md5_qops2qopstr()

```
const char * digest_md5_qops2qopstr (
            int qops )
```

Definition at line 89 of file qop.c.

#### 5.27.1.2 digest_md5_qopstr2qops()

```
int digest_md5_qopstr2qops (
            const char * qopstr )
```

Definition at line 34 of file qop.c.

## 5.28 qop.c

Go to the documentation of this file.

```
00001 /* qop.c --- DIGEST-MD5 QOP handling.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get prototypes. */
00025 #include "qop.h"
00026
00027 #include "tokens.h"
00028 #include "parser.h"
00029
00030 #include <string.h>
00031 #include <stdlib.h>
00032
00033 int
00034 digest_md5_qopstr2qops (const char *qopstr)
00035 {
00036   int qops = 0;
00037   enum
00038   {
00039     /* the order must match the following struct */
00040     QOP_AUTH = 0,
00041     QOP_AUTH_INT,
00042     QOP_AUTH_CONF
00043   };
00044   const char *const qop_opts[] = {
00045     /* the order must match the previous enum */
00046     "qop-auth",
00047     "qop-int",
00048     "qop-conf",
00049     NULL
00050   };
00051   char *subsubopts;
00052   char *val;
00053   char *qopdup;
00054
00055   if (!qopstr)
00056     return 0;
00057
00058   qopdup = strdup (qopstr);
00059   if (!qopdup)
00060     return -1;
00061
00062   subsubopts = qopdup;
00063   while (*subsubopts != '\0')
00064     switch (digest_md5_getsubopt (&subsubopts, qop_opts, &val))
00065       {
00066       case QOP_AUTH:
00067         qops |= DIGEST_MD5_QOP_AUTH;
00068         break;
```

```
00069
00070       case QOP_AUTH_INT:
00071         qops |= DIGEST_MD5_QOP_AUTH_INT;
00072         break;
00073
00074       case QOP_AUTH_CONF:
00075         qops |= DIGEST_MD5_QOP_AUTH_CONF;
00076         break;
00077
00078       default:
00079         /* ignore unrecognized options */
00080         break;
00081       }
00082
00083   free (qopdup);
00084
00085   return qops;
00086 }
00087
00088 const char *
00089 digest_md5_qops2qopstr (int qops)
00090 {
00091   const char *qopstr[] = {
00092     /* 0 */ "qop-auth",
00093     /* 1 */ "qop-auth",
00094     /* 2 */ "qop-int",
00095     /* 3 */ "qop-auth, qop-int",
00096     /* 4 */ "qop-conf",
00097     /* 5 */ "qop-auth, qop-conf",
00098     /* 6 */ "qop-int, qop-conf",
00099     /* 7 */ "qop-auth, qop-int, qop-conf"
00100   };
00101
00102   return qopstr[qops & 0x07];
00103 }
```

## 5.29 qop.h File Reference

### Functions

- int digest_md5_qopstr2qops (const char *qopstr)
- const char * digest_md5_qops2qopstr (int qops)

### 5.29.1 Function Documentation

#### 5.29.1.1 digest_md5_qops2qopstr()

```
const char * digest_md5_qops2qopstr (
            int qops )  [extern]
```

Definition at line 89 of file qop.c.

#### 5.29.1.2 digest_md5_qopstr2qops()

```
int digest_md5_qopstr2qops (
            const char * qopstr )  [extern]
```

Definition at line 34 of file qop.c.

## 5.30 qop.h

Go to the documentation of this file.

```
00001 /* qop.h --- Prototypes for DIGEST-MD5 qop handling.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef DIGEST_MD5_QOP_H
00023 # define DIGEST_MD5_QOP_H
00024
00025 extern int digest_md5_qopstr2qops (const char *qopstr);
00026 extern const char *digest_md5_qops2qopstr (int qops);
00027
00028 #endif /* DIGEST_MD5_QOP_H */
```

## 5.31 session.c File Reference

```
#include <config.h>
#include "session.h"
#include <stdlib.h>
#include <string.h>
#include <gc.h>
```

### Macros

- #define MD5LEN 16
- #define SASL_INTEGRITY_PREFIX_LENGTH 4
- #define MAC_DATA_LEN 4
- #define MAC_HMAC_LEN 10
- #define MAC_MSG_TYPE "\x00\x01"
- #define MAC_MSG_TYPE_LEN 2
- #define MAC_SEQNUM_LEN 4
- #define C2I(buf)

### Functions

- int digest_md5_encode (const char *input, size_t input_len, char **output, size_t *output_len, digest_md5_qop qop, unsigned long sendseqnum, char key[DIGEST_MD5_LENGTH])
- int digest_md5_decode (const char *input, size_t input_len, char **output, size_t *output_len, digest_md5_qop qop, unsigned long readseqnum, char key[DIGEST_MD5_LENGTH])

### 5.31.1 Macro Definition Documentation

#### 5.31.1.1 C2I

```
#define C2I(
              buf )
```

**Value:**

```
((buf[3] & 0xFF) |            \
((buf[2] & 0xFF) « 8) |       \
((buf[1] & 0xFF) « 16) |      \
((buf[0] & 0xFF) « 24))
```

Definition at line 113 of file session.c.

#### 5.31.1.2 MAC_DATA_LEN

```
#define MAC_DATA_LEN 4
```

Definition at line 38 of file session.c.

#### 5.31.1.3 MAC_HMAC_LEN

```
#define MAC_HMAC_LEN 10
```

Definition at line 39 of file session.c.

#### 5.31.1.4 MAC_MSG_TYPE

```
#define MAC_MSG_TYPE "\x00\x01"
```

Definition at line 40 of file session.c.

#### 5.31.1.5 MAC_MSG_TYPE_LEN

```
#define MAC_MSG_TYPE_LEN 2
```

Definition at line 41 of file session.c.

#### 5.31.1.6 MAC_SEQNUM_LEN

```
#define MAC_SEQNUM_LEN 4
```

Definition at line 42 of file session.c.

#### 5.31.1.7 MD5LEN

```
#define MD5LEN 16
```

Definition at line 36 of file session.c.

#### 5.31.1.8 SASL_INTEGRITY_PREFIX_LENGTH

```
#define SASL_INTEGRITY_PREFIX_LENGTH 4
```

Definition at line 37 of file session.c.

### 5.31.2 Function Documentation

#### 5.31.2.1 digest_md5_decode()

```
int digest_md5_decode (
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len,
              digest_md5_qop qop,
              unsigned long readseqnum,
              char key[DIGEST_MD5_LENGTH] )
```

Definition at line 119 of file session.c.

#### 5.31.2.2 digest_md5_encode()

```
int digest_md5_encode (
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len,
              digest_md5_qop qop,
              unsigned long sendseqnum,
              char key[DIGEST_MD5_LENGTH] )
```

Definition at line 45 of file session.c.

## 5.32 session.c

Go to the documentation of this file.

```
00001 /* session.c --- Data integrity/privacy protection of DIGEST-MD5.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
```

```
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "session.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get memcpy, strdup, strlen. */
00031 #include <string.h>
00032
00033 /* Get gc_hmac_md5. */
00034 #include <gc.h>
00035
00036 #define MD5LEN 16
00037 #define SASL_INTEGRITY_PREFIX_LENGTH 4
00038 #define MAC_DATA_LEN 4
00039 #define MAC_HMAC_LEN 10
00040 #define MAC_MSG_TYPE "\x00\x01"
00041 #define MAC_MSG_TYPE_LEN 2
00042 #define MAC_SEQNUM_LEN 4
00043
00044 int
00045 digest_md5_encode (const char *input, size_t input_len,
00046                    char **output, size_t *output_len,
00047                    digest_md5_qop qop,
00048                    unsigned long sendseqnum, char key[DIGEST_MD5_LENGTH])
00049 {
00050   int res;
00051
00052   if (qop & DIGEST_MD5_QOP_AUTH_CONF)
00053     {
00054       return -1;
00055     }
00056   else if (qop & DIGEST_MD5_QOP_AUTH_INT)
00057     {
00058       char *seqnumin;
00059       char hash[GC_MD5_DIGEST_SIZE];
00060       size_t len;
00061
00062       seqnumin = malloc (MAC_SEQNUM_LEN + input_len);
00063       if (seqnumin == NULL)
00064         return -1;
00065
00066       seqnumin[0] = (sendseqnum >> 24) & 0xFF;
00067       seqnumin[1] = (sendseqnum >> 16) & 0xFF;
00068       seqnumin[2] = (sendseqnum >> 8) & 0xFF;
00069       seqnumin[3] = sendseqnum & 0xFF;
00070       memcpy (seqnumin + MAC_SEQNUM_LEN, input, input_len);
00071
00072       res = gc_hmac_md5 (key, MD5LEN,
00073                          seqnumin, MAC_SEQNUM_LEN + input_len, hash);
00074       free (seqnumin);
00075       if (res)
00076         return -1;
00077
00078       *output_len = MAC_DATA_LEN + input_len + MAC_HMAC_LEN +
00079         MAC_MSG_TYPE_LEN + MAC_SEQNUM_LEN;
00080       *output = malloc (*output_len);
00081       if (!*output)
00082         return -1;
00083
00084       len = MAC_DATA_LEN;
00085       memcpy (*output + len, input, input_len);
00086       len += input_len;
00087       memcpy (*output + len, hash, MAC_HMAC_LEN);
00088       len += MAC_HMAC_LEN;
00089       memcpy (*output + len, MAC_MSG_TYPE, MAC_MSG_TYPE_LEN);
00090       len += MAC_MSG_TYPE_LEN;
00091       (*output + len)[0] = (sendseqnum >> 24) & 0xFF;
00092       (*output + len)[1] = (sendseqnum >> 16) & 0xFF;
00093       (*output + len)[2] = (sendseqnum >> 8) & 0xFF;
00094       (*output + len)[3] = sendseqnum & 0xFF;
00095       len += MAC_SEQNUM_LEN;
00096       (*output)[0] = ((len - MAC_DATA_LEN) >> 24) & 0xFF;
00097       (*output)[1] = ((len - MAC_DATA_LEN) >> 16) & 0xFF;
00098       (*output)[2] = ((len - MAC_DATA_LEN) >> 8) & 0xFF;
00099       (*output)[3] = (len - MAC_DATA_LEN) & 0xFF;
00100     }
00101   else
00102     {
00103       *output_len = input_len;
00104       *output = malloc (input_len);
00105       if (!*output)
00106         return -1;
00107       memcpy (*output, input, input_len);
00108     }
```

```
00109
00110   return 0;
00111 }
00112
00113 #define C2I(buf) ((buf[3] & 0xFF) |             \
00114                   ((buf[2] & 0xFF) << 8) |      \
00115                   ((buf[1] & 0xFF) << 16) |     \
00116                   ((buf[0] & 0xFF) << 24))
00117
00118 int
00119 digest_md5_decode (const char *input, size_t input_len,
00120                    char **output, size_t *output_len,
00121                    digest_md5_qop qop,
00122                    unsigned long readseqnum, char key[DIGEST_MD5_LENGTH])
00123 {
00124   if (qop & DIGEST_MD5_QOP_AUTH_CONF)
00125     {
00126       return -1;
00127     }
00128   else if (qop & DIGEST_MD5_QOP_AUTH_INT)
00129     {
00130       char *seqnumin;
00131       char hash[GC_MD5_DIGEST_SIZE];
00132       unsigned long len;
00133       char tmpbuf[SASL_INTEGRITY_PREFIX_LENGTH];
00134       int res;
00135
00136       if (input_len < SASL_INTEGRITY_PREFIX_LENGTH)
00137         return -2;
00138
00139       len = C2I (input);
00140
00141       if (input_len < SASL_INTEGRITY_PREFIX_LENGTH + len)
00142         return -2;
00143
00144       len -= MAC_HMAC_LEN + MAC_MSG_TYPE_LEN + MAC_SEQNUM_LEN;
00145
00146       seqnumin = malloc (SASL_INTEGRITY_PREFIX_LENGTH + len);
00147       if (seqnumin == NULL)
00148         return -1;
00149
00150       tmpbuf[0] = (readseqnum >> 24) & 0xFF;
00151       tmpbuf[1] = (readseqnum >> 16) & 0xFF;
00152       tmpbuf[2] = (readseqnum >> 8) & 0xFF;
00153       tmpbuf[3] = readseqnum & 0xFF;
00154
00155       memcpy (seqnumin, tmpbuf, SASL_INTEGRITY_PREFIX_LENGTH);
00156       memcpy (seqnumin + SASL_INTEGRITY_PREFIX_LENGTH,
00157               input + MAC_DATA_LEN, len);
00158
00159       res = gc_hmac_md5 (key, MD5LEN, seqnumin, MAC_SEQNUM_LEN + len, hash);
00160       free (seqnumin);
00161       if (res)
00162         return -1;
00163
00164       if (memcmp
00165           (hash,
00166            input + input_len - MAC_SEQNUM_LEN - MAC_MSG_TYPE_LEN -
00167            MAC_HMAC_LEN, MAC_HMAC_LEN) == 0
00168           && memcmp (MAC_MSG_TYPE,
00169                      input + input_len - MAC_SEQNUM_LEN - MAC_MSG_TYPE_LEN,
00170                      MAC_MSG_TYPE_LEN) == 0
00171           && memcmp (tmpbuf, input + input_len - MAC_SEQNUM_LEN,
00172                      MAC_SEQNUM_LEN) == 0)
00173         {
00174           *output_len = len;
00175           *output = malloc (*output_len);
00176           if (!*output)
00177             return -1;
00178           memcpy (*output, input + MAC_DATA_LEN, len);
00179         }
00180       else
00181         return -1;
00182     }
00183   else
00184     {
00185       *output_len = input_len;
00186       *output = malloc (input_len);
00187       if (!*output)
00188         return -1;
00189       memcpy (*output, input, input_len);
00190     }
00191
00192   return 0;
00193 }
```

## 5.33 session.h File Reference

```
#include "tokens.h"
```

### Functions

- int digest_md5_encode (const char *input, size_t input_len, char **output, size_t *output_len, digest_md5_qop qop, unsigned long sendseqnum, char key[DIGEST_MD5_LENGTH])
- int digest_md5_decode (const char *input, size_t input_len, char **output, size_t *output_len, digest_md5_qop qop, unsigned long readseqnum, char key[DIGEST_MD5_LENGTH])

### 5.33.1 Function Documentation

#### 5.33.1.1 digest_md5_decode()

```
int digest_md5_decode (
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len,
            digest_md5_qop qop,
            unsigned long readseqnum,
            char key[DIGEST_MD5_LENGTH] )  [extern]
```

Definition at line 119 of file session.c.

#### 5.33.1.2 digest_md5_encode()

```
int digest_md5_encode (
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len,
            digest_md5_qop qop,
            unsigned long sendseqnum,
            char key[DIGEST_MD5_LENGTH] )  [extern]
```

Definition at line 45 of file session.c.

## 5.34 session.h

Go to the documentation of this file.

```
00001 /* session.h --- Data integrity/privacy protection of DIGEST-MD5.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef DIGEST_MD5_SESSION_H
00023 # define DIGEST_MD5_SESSION_H
00024
00025 /* Get token types. */
00026 # include "tokens.h"
00027
00028 extern int digest_md5_encode (const char *input, size_t input_len,
00029                               char **output, size_t *output_len,
00030                               digest_md5_qop qop,
00031                               unsigned long sendseqnum,
00032                               char key[DIGEST_MD5_LENGTH]);
00033
00034 extern int digest_md5_decode (const char *input, size_t input_len,
00035                               char **output, size_t *output_len,
00036                               digest_md5_qop qop,
00037                               unsigned long readseqnum,
00038                               char key[DIGEST_MD5_LENGTH]);
00039
00040 #endif /* DIGEST_MD5_SESSION_H */
```

## 5.35 test-parser.c File Reference

```
#include <config.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "free.h"
#include "parser.h"
#include "printer.h"
#include "digesthmac.h"
#include "gc.h"
```

### Functions

- int main (void)

### 5.35.1 Function Documentation

#### 5.35.1.1 main()

```
int main (
            void  )
```

Definition at line 36 of file test-parser.c.

# 5.36 test-parser.c

Go to the documentation of this file.

```
00001 /* test-parser.c --- Self tests of DIGEST-MD5 parser & printer.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 #include <stdio.h>
00025 #include <stdlib.h>
00026 #include <string.h>
00027
00028 #include "free.h"
00029 #include "parser.h"
00030 #include "printer.h"
00031 #include "digesthmac.h"
00032
00033 #include "gc.h"
00034
00035 int
00036 main (void)
00037 {
00038   digest_md5_challenge c;
00039   digest_md5_response r;
00040   digest_md5_finish f;
00041   char buf32[33];
00042   char buf16[16];
00043   int rc;
00044   char *tmp;
00045
00046   {
00047     const char *token = "nonce=4711, foo=bar, algorithm=md5-sess";
00048
00049     printf ("challenge `%s': ", token);
00050     rc = digest_md5_parse_challenge (token, 0, &c);
00051     if (rc != 0)
00052       abort ();
00053     printf ("nonce `%s': %s", c.nonce,
00054             strcmp ("4711", c.nonce) == 0 ? "PASS" : "FAILURE");
00055     printf ("\n");
00056     tmp = digest_md5_print_challenge (&c);
00057     if (!tmp)
00058       abort ();
00059     printf ("printed `%s' PASS\n", tmp);
00060     free (tmp);
00061     digest_md5_free_challenge (&c);
00062   }
00063
00064   {
00065     const char *token =
00066       "qop=\"auth, auth-conf\", nonce=42, algorithm=md5-sess";
00067
00068     printf ("challenge `%s': ", token);
00069     rc = digest_md5_parse_challenge (token, 0, &c);
00070     if (rc == 0)
00071       abort ();
00072     digest_md5_free_challenge (&c);
00073     printf ("PASS\n");
00074   }
00075
00076   {
00077     const char *token = "cipher=\"des\", nonce=42, algorithm=md5-sess";
00078
00079     printf ("challenge `%s': ", token);
00080     rc = digest_md5_parse_challenge (token, 0, &c);
00081     if (rc == 0)
00082       abort ();
```

```
00083     digest_md5_free_challenge (&c);
00084     printf ("PASS\n");
00085   }
00086
00087   {
00088     const char *token = "qop=\"auth, auth-conf\", nonce=42, "
00089       "algorithm=md5-sess, cipher=\"des\"";
00090
00091     printf ("challenge `%s': ", token);
00092     rc = digest_md5_parse_challenge (token, 0, &c);
00093     if (rc != 0)
00094       abort ();
00095     printf ("qop %02x ciphers %02x: %s\n",
00096             (unsigned) c.qops, (unsigned) c.ciphers,
00097             (c.qops == 5 && c.ciphers == 1) ? "PASS" : "FAILURE");
00098     tmp = digest_md5_print_challenge (&c);
00099     if (!tmp)
00100       abort ();
00101     printf ("printed `%s' PASS\n", tmp);
00102     free (tmp);
00103     digest_md5_free_challenge (&c);
00104   }
00105
00106   {
00107     const char *token = "bar=foo, foo=bar";
00108
00109     printf ("challenge `%s': ", token);
00110     rc = digest_md5_parse_challenge (token, 0, &c);
00111     if (rc == 0)
00112       abort ();
00113     printf ("PASS\n");
00114   }
00115
00116   {
00117     const char *token = "realm=foo, realm=bar, nonce=42, algorithm=md5-sess";
00118
00119     printf ("challenge `%s': ", token);
00120     rc = digest_md5_parse_challenge (token, 0, &c);
00121     if (rc != 0)
00122       abort ();
00123     if (c.nrealms != 2)
00124       abort ();
00125     printf ("realms `%s', `%s': PASS\n", c.realms[0], c.realms[1]);
00126     tmp = digest_md5_print_challenge (&c);
00127     if (!tmp)
00128       abort ();
00129     printf ("printed `%s' PASS\n", tmp);
00130     free (tmp);
00131     digest_md5_free_challenge (&c);
00132   }
00133
00134   /* Response */
00135
00136   {
00137     const char *token = "bar=foo, foo=bar";
00138
00139     printf ("response `%s': ", token);
00140     rc = digest_md5_parse_response (token, 0, &r);
00141     if (rc == 0)
00142       abort ();
00143     printf ("PASS\n");
00144     digest_md5_free_response (&r);
00145   }
00146
00147   {
00148     const char *token = "username=jas, nonce=42, cnonce=4711, nc=00000001, "
00149       "digest-uri=foo, response=01234567890123456789012345678901";
00150
00151     printf ("response `%s': ", token);
00152     rc = digest_md5_parse_response (token, 0, &r);
00153     if (rc != 0)
00154       abort ();
00155     printf ("username `%s', nonce `%s', cnonce `%s',"
00156             " nc %08lx, digest-uri `%s', response `%s': PASS\n",
00157             r.username, r.nonce, r.cnonce, r.nc, r.digesturi, r.response);
00158     tmp = digest_md5_print_response (&r);
00159     if (!tmp)
00160       abort ();
00161     printf ("printed `%s' PASS\n", tmp);
00162     free (tmp);
00163     digest_md5_free_response (&r);
00164   }
00165
00166   /* Auth-response, finish. */
00167
00168   {
00169     const char *token = "rspauth=\"6a204da26b9888ee40bb3052ff056a67\"";
```

```
00170
00171     printf ("finish `%s': ", token);
00172     rc = digest_md5_parse_finish (token, 0, &f);
00173     if (rc != 0)
00174       abort ();
00175     printf ("`%s'? %s\n", f.rspauth,
00176             strcmp ("6a204da26b9888ee40bb3052ff056a67", f.rspauth) == 0
00177             ? "ok" : "FAILURE");
00178     digest_md5_free_response (&r);
00179   }
00180
00181   {
00182     const char *token = "bar=foo, foo=bar";
00183
00184     printf ("finish `%s': ", token);
00185     rc = digest_md5_parse_finish (token, 0, &f);
00186     if (rc == 0)
00187       abort ();
00188     printf ("invalid? PASS\n");
00189     digest_md5_free_finish (&f);
00190   }
00191
00192   rc = gc_init ();
00193   if (rc != 0)
00194     {
00195       printf ("gc_init error %d\n", rc);
00196       abort ();
00197     }
00198
00199   memset (buf16, 'Q', 16);
00200
00201   rc = digest_md5_hmac (buf32, buf16, "nonce", 1, "cnonce",
00202                         DIGEST_MD5_QOP_AUTH, "authzid", "digesturi",
00203                         1, 0, NULL, NULL, NULL, NULL);
00204   if (rc != 0)
00205     abort ();
00206   buf32[32] = '\0';
00207   if (strcmp (buf32, "6a204da26b9888ee40bb3052ff056a67") != 0)
00208     abort ();
00209   printf ("digest: `%s': PASS\n", buf32);
00210
00211   rc = digest_md5_hmac (buf32, buf16, "nonce", 1, "cnonce",
00212                         DIGEST_MD5_QOP_AUTH, "authzid", "digesturi", 0, 0,
00213                         NULL, NULL, NULL, NULL);
00214   if (rc != 0)
00215     abort ();
00216   buf32[32] = '\0';
00217   if (strcmp (buf32, "6c1f58bfa46e9c225b93745c84204efd") != 0)
00218     abort ();
00219   printf ("digest: `%s': PASS\n", buf32);
00220
00221   return 0;
00222 }
```

## 5.37 external.h File Reference

`#include <gsasl.h>`

### Macros

- #define GSASL_EXTERNAL_NAME "EXTERNAL"

### Functions

- int _gsasl_external_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- int _gsasl_external_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)

**Variables**

- Gsasl_mechanism _gsasl_external_mechanism

## 5.37.1 Macro Definition Documentation

### 5.37.1.1 GSASL_EXTERNAL_NAME

```
#define GSASL_EXTERNAL_NAME "EXTERNAL"
```

Definition at line 27 of file external.h.

## 5.37.2 Function Documentation

### 5.37.2.1 _gsasl_external_client_step()

```
int _gsasl_external_client_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

### 5.37.2.2 _gsasl_external_server_step()

```
int _gsasl_external_server_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

## 5.37.3 Variable Documentation

### 5.37.3.1 _gsasl_external_mechanism

```
Gsasl_mechanism _gsasl_external_mechanism  [extern]
```

Definition at line 27 of file external/mechinfo.c.

## 5.38 external.h

Go to the documentation of this file.

```
00001 /* external.h --- Prototypes for EXTERNAL mechanism as defined in RFC 2222.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef EXTERNAL_H
00023 # define EXTERNAL_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_EXTERNAL_NAME "EXTERNAL"
00028
00029 extern Gsasl_mechanism _gsasl_external_mechanism;
00030
00031 extern int _gsasl_external_client_step (Gsasl_session * sctx,
00032                                         void *mech_data,
00033                                         const char *input, size_t input_len,
00034                                         char **output, size_t *output_len);
00035
00036 extern int _gsasl_external_server_step (Gsasl_session * sctx,
00037                                         void *mech_data,
00038                                         const char *input, size_t input_len,
00039                                         char **output, size_t *output_len);
00040
00041 #endif /* EXTERNAL_H */
```

## 5.39 gs2.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_GS2_KRB5_NAME "GS2-KRB5"

### Functions

- int _gsasl_gs2_client_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_gs2_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_gs2_client_finish (Gsasl_session *sctx, void *mech_data)
- int _gsasl_gs2_server_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_gs2_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_gs2_server_finish (Gsasl_session *sctx, void *mech_data)

**Variables**

- Gsasl_mechanism _gsasl_gs2_krb5_mechanism

## 5.39.1 Macro Definition Documentation

### 5.39.1.1 GSASL_GS2_KRB5_NAME

```
#define GSASL_GS2_KRB5_NAME "GS2-KRB5"
```

Definition at line 27 of file gs2.h.

## 5.39.2 Function Documentation

### 5.39.2.1 _gsasl_gs2_client_finish()

```
void _gsasl_gs2_client_finish (
             Gsasl_session * sctx,
             void * mech_data )  [extern]
```

Definition at line 315 of file gs2/client.c.

### 5.39.2.2 _gsasl_gs2_client_start()

```
int _gsasl_gs2_client_start (
             Gsasl_session * sctx,
             void ** mech_data )  [extern]
```

Definition at line 51 of file gs2/client.c.

### 5.39.2.3 _gsasl_gs2_client_step()

```
int _gsasl_gs2_client_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

Definition at line 234 of file gs2/client.c.

### 5.39.2.4 _gsasl_gs2_server_finish()

```
void _gsasl_gs2_server_finish (
             Gsasl_session * sctx,
             void * mech_data )  [extern]
```

Definition at line 298 of file gs2/server.c.

#### 5.39.2.5 _gsasl_gs2_server_start()

```
int _gsasl_gs2_server_start (
             Gsasl_session * sctx,
             void ** mech_data )  [extern]
```

Definition at line 118 of file gs2/server.c.

#### 5.39.2.6 _gsasl_gs2_server_step()

```
int _gsasl_gs2_server_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

Definition at line 161 of file gs2/server.c.

### 5.39.3 Variable Documentation

#### 5.39.3.1 _gsasl_gs2_krb5_mechanism

```
Gsasl_mechanism _gsasl_gs2_krb5_mechanism  [extern]
```

Definition at line 27 of file gs2/mechinfo.c.

## 5.40 gs2.h

Go to the documentation of this file.

```
00001 /* gs2.h --- Prototypes for SASL mechanism GS2.
00002  * Copyright (C) 2006-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef GS2_H
00023 # define GS2_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_GS2_KRB5_NAME "GS2-KRB5"
00028
00029 extern Gsasl_mechanism _gsasl_gs2_krb5_mechanism;
00030
```

```
00031 extern int _gsasl_gs2_client_start (Gsasl_session * sctx, void **mech_data);
00032 extern int _gsasl_gs2_client_step (Gsasl_session * sctx,
00033                                    void *mech_data,
00034                                    const char *input, size_t input_len,
00035                                    char **output, size_t *output_len);
00036 extern void _gsasl_gs2_client_finish (Gsasl_session * sctx, void *mech_data);
00037
00038 extern int _gsasl_gs2_server_start (Gsasl_session * sctx, void **mech_data);
00039 extern int _gsasl_gs2_server_step (Gsasl_session * sctx,
00040                                    void *mech_data,
00041                                    const char *input, size_t input_len,
00042                                    char **output, size_t *output_len);
00043 extern void _gsasl_gs2_server_finish (Gsasl_session * sctx, void *mech_data);
00044
00045 #endif /* GS2_H */
```

## 5.41 gs2helper.c File Reference

```
#include <config.h>
#include <string.h>
#include <stdlib.h>
#include "gs2helper.h"
```

### Functions

- int gs2_get_oid (Gsasl_session *sctx, gss_OID *mech_oid)

### 5.41.1 Function Documentation

#### 5.41.1.1 gs2_get_oid()

```
int gs2_get_oid (
            Gsasl_session * sctx,
            gss_OID * mech_oid )
```

Definition at line 37 of file gs2helper.c.

## 5.42 gs2helper.c

Go to the documentation of this file.

```
00001 /* gs2helper.c --- GS2 helper functions common to client and server.
00002  * Copyright (C) 2010-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
```

```
00021
00022 #include <config.h>
00023
00024 /* Get strcmp. */
00025 #include <string.h>
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get specification. */
00031 #include "gs2helper.h"
00032
00033 /* Populate mech_oid with OID for the current SASL mechanism name.  A
00034    bit silly given that we only support Kerberos V5 today, but will be
00035    useful when that changes.  */
00036 int
00037 gs2_get_oid (Gsasl_session *sctx, gss_OID *mech_oid)
00038 {
00039   gss_buffer_desc sasl_mech_name;
00040   OM_uint32 maj_stat, min_stat;
00041
00042   sasl_mech_name.value = (void *) gsasl_mechanism_name (sctx);
00043   if (!sasl_mech_name.value)
00044     return GSASL_AUTHENTICATION_ERROR;
00045   sasl_mech_name.length = strlen (sasl_mech_name.value);
00046
00047   maj_stat = gss_inquire_mech_for_saslname (&min_stat, &sasl_mech_name,
00048                                             mech_oid);
00049   if (GSS_ERROR (maj_stat))
00050     return GSASL_GSSAPI_INQUIRE_MECH_FOR_SASLNAME_ERROR;
00051
00052   return GSASL_OK;
00053 }
```

## 5.43 gs2helper.h File Reference

```
#include "gss-extra.h"
#include <gsasl.h>
```

### Functions

- int gs2_get_oid (Gsasl_session *sctx, gss_OID *mech_oid)

### 5.43.1 Function Documentation

#### 5.43.1.1 gs2_get_oid()

```
int gs2_get_oid (
            Gsasl_session * sctx,
            gss_OID * mech_oid )  [extern]
```

Definition at line 37 of file gs2helper.c.

## 5.44 gs2helper.h

Go to the documentation of this file.

```
00001 /* gs2helper.h --- GS2 helper functions common to client and server.
00002  * Copyright (C) 2010-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef GS2_HELPER_H
00023 # define GS2_HELPER_H
00024
00025 /* For GSS-API types. */
00026 # include "gss-extra.h"
00027
00028 /* Get gsasl functions and types. */
00029 # include <gsasl.h>
00030
00031 extern int gs2_get_oid (Gsasl_session * sctx, gss_OID * mech_oid);
00032
00033 #endif /* GS2_HELPER_H */
```

## 5.45 x-gssapi.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_GSSAPI_NAME "GSSAPI"

### Functions

- int _gsasl_gssapi_client_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_gssapi_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_gssapi_client_finish (Gsasl_session *sctx, void *mech_data)
- int _gsasl_gssapi_client_encode (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- int _gsasl_gssapi_client_decode (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- int _gsasl_gssapi_server_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_gssapi_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_gssapi_server_finish (Gsasl_session *sctx, void *mech_data)

**Variables**

- Gsasl_mechanism _gsasl_gssapi_mechanism

## 5.45.1 Macro Definition Documentation

### 5.45.1.1 GSASL_GSSAPI_NAME

```
#define GSASL_GSSAPI_NAME "GSSAPI"
```

Definition at line 27 of file x-gssapi.h.

## 5.45.2 Function Documentation

### 5.45.2.1 _gsasl_gssapi_client_decode()

```
int _gsasl_gssapi_client_decode (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

Definition at line 321 of file gssapi/client.c.

### 5.45.2.2 _gsasl_gssapi_client_encode()

```
int _gsasl_gssapi_client_encode (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

Definition at line 266 of file gssapi/client.c.

### 5.45.2.3 _gsasl_gssapi_client_finish()

```
void _gsasl_gssapi_client_finish (
             Gsasl_session * sctx,
             void * mech_data )  [extern]
```

Definition at line 248 of file gssapi/client.c.

#### 5.45.2.4 _gsasl_gssapi_client_start()

```
int _gsasl_gssapi_client_start (
             Gsasl_session * sctx,
             void ** mech_data )  [extern]
```

Definition at line 46 of file gssapi/client.c.

#### 5.45.2.5 _gsasl_gssapi_client_step()

```
int _gsasl_gssapi_client_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

Definition at line 65 of file gssapi/client.c.

#### 5.45.2.6 _gsasl_gssapi_server_finish()

```
void _gsasl_gssapi_server_finish (
             Gsasl_session * sctx,
             void * mech_data )  [extern]
```

Definition at line 265 of file gssapi/server.c.

#### 5.45.2.7 _gsasl_gssapi_server_start()

```
int _gsasl_gssapi_server_start (
             Gsasl_session * sctx,
             void ** mech_data )  [extern]
```

Definition at line 46 of file gssapi/server.c.

#### 5.45.2.8 _gsasl_gssapi_server_step()

```
int _gsasl_gssapi_server_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

Definition at line 64 of file gssapi/server.c.

### 5.45.3 Variable Documentation

#### 5.45.3.1 _gsasl_gssapi_mechanism

```
Gsasl_mechanism _gsasl_gssapi_mechanism  [extern]
```

Definition at line 27 of file gssapi/mechinfo.c.

## 5.46 x-gssapi.h

Go to the documentation of this file.

```
00001 /* x-gssapi.h --- Prototypes for SASL mechanism GSSAPI as defined in RFC 2222.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef X_GSSAPI_H
00023 # define X_GSSAPI_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_GSSAPI_NAME "GSSAPI"
00028
00029 extern Gsasl_mechanism _gsasl_gssapi_mechanism;
00030
00031 extern int _gsasl_gssapi_client_start (Gsasl_session * sctx,
00032                                        void **mech_data);
00033 extern int _gsasl_gssapi_client_step (Gsasl_session * sctx,
00034                                       void *mech_data,
00035                                       const char *input, size_t input_len,
00036                                       char **output, size_t *output_len);
00037 extern void _gsasl_gssapi_client_finish (Gsasl_session * sctx,
00038                                          void *mech_data);
00039 extern int _gsasl_gssapi_client_encode (Gsasl_session * sctx,
00040                                         void *mech_data,
00041                                         const char *input, size_t input_len,
00042                                         char **output, size_t *output_len);
00043 extern int _gsasl_gssapi_client_decode (Gsasl_session * sctx,
00044                                         void *mech_data,
00045                                         const char *input, size_t input_len,
00046                                         char **output, size_t *output_len);
00047
00048 extern int _gsasl_gssapi_server_start (Gsasl_session * sctx,
00049                                        void **mech_data);
00050 extern int _gsasl_gssapi_server_step (Gsasl_session * sctx,
00051                                       void *mech_data,
00052                                       const char *input, size_t input_len,
00053                                       char **output, size_t *output_len);
00054 extern void _gsasl_gssapi_server_finish (Gsasl_session * sctx,
00055                                          void *mech_data);
00056
00057 #endif /* X_GSSAPI_H */
```

## 5.47 login.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_LOGIN_NAME "LOGIN"

### Functions

- int _gsasl_login_client_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_login_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_login_client_finish (Gsasl_session *sctx, void *mech_data)
- int _gsasl_login_server_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_login_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_login_server_finish (Gsasl_session *sctx, void *mech_data)

### Variables

- Gsasl_mechanism _gsasl_login_mechanism

## 5.47.1 Macro Definition Documentation

### 5.47.1.1 GSASL_LOGIN_NAME

```
#define GSASL_LOGIN_NAME "LOGIN"
```

Definition at line 27 of file login.h.

## 5.47.2 Function Documentation

### 5.47.2.1 _gsasl_login_client_finish()

```
void _gsasl_login_client_finish (
            Gsasl_session * sctx,
            void * mech_data )  [extern]
```

### 5.47.2.2 _gsasl_login_client_start()

```
int _gsasl_login_client_start (
            Gsasl_session * sctx,
            void ** mech_data )  [extern]
```

### 5.47.2.3 _gsasl_login_client_step()

```
int _gsasl_login_client_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

#### 5.47.2.4 _gsasl_login_server_finish()

```
void _gsasl_login_server_finish (
            Gsasl_session * sctx,
            void * mech_data )  [extern]
```

#### 5.47.2.5 _gsasl_login_server_start()

```
int _gsasl_login_server_start (
            Gsasl_session * sctx,
            void ** mech_data )  [extern]
```

#### 5.47.2.6 _gsasl_login_server_step()

```
int _gsasl_login_server_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

Definition at line 58 of file login/server.c.

### 5.47.3 Variable Documentation

#### 5.47.3.1 _gsasl_login_mechanism

```
Gsasl_mechanism _gsasl_login_mechanism  [extern]
```

Definition at line 27 of file login/mechinfo.c.

## 5.48 login.h

Go to the documentation of this file.

```
00001 /* login.h --- Prototypes for non-standard SASL mechanism LOGIN.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
```

```
00022 #ifndef LOGIN_H
00023 # define LOGIN_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_LOGIN_NAME "LOGIN"
00028
00029 extern Gsasl_mechanism _gsasl_login_mechanism;
00030
00031 extern int _gsasl_login_client_start (Gsasl_session * sctx, void **mech_data);
00032 extern int _gsasl_login_client_step (Gsasl_session * sctx,
00033                                      void *mech_data,
00034                                      const char *input, size_t input_len,
00035                                      char **output, size_t *output_len);
00036 extern void _gsasl_login_client_finish (Gsasl_session * sctx,
00037                                         void *mech_data);
00038
00039 extern int _gsasl_login_server_start (Gsasl_session * sctx, void **mech_data);
00040 extern int _gsasl_login_server_step (Gsasl_session * sctx,
00041                                      void *mech_data,
00042                                      const char *input, size_t input_len,
00043                                      char **output, size_t *output_len);
00044 extern void _gsasl_login_server_finish (Gsasl_session * sctx,
00045                                         void *mech_data);
00046
00047 #endif /* LOGIN_H */
```

## 5.49 ntlm.c File Reference

```
#include <config.h>
#include <stdlib.h>
#include <string.h>
#include "x-ntlm.h"
#include <ntlm.h>
```

### Data Structures

- struct _Gsasl_ntlm_state

### Typedefs

- typedef struct _Gsasl_ntlm_state _Gsasl_ntlm_state

### Functions

- int _gsasl_ntlm_client_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
- int _gsasl_ntlm_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_ntlm_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)

### 5.49.1 Typedef Documentation

#### 5.49.1.1 _Gsasl_ntlm_state

```
typedef struct _Gsasl_ntlm_state _Gsasl_ntlm_state
```

Definition at line 39 of file ntlm.c.

### 5.49.2 Function Documentation

#### 5.49.2.1 _gsasl_ntlm_client_finish()

```
void _gsasl_ntlm_client_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 164 of file ntlm.c.

#### 5.49.2.2 _gsasl_ntlm_client_start()

```
int _gsasl_ntlm_client_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 42 of file ntlm.c.

#### 5.49.2.3 _gsasl_ntlm_client_step()

```
int _gsasl_ntlm_client_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 58 of file ntlm.c.

## 5.50 ntlm.c

Go to the documentation of this file.

```
00001 /* ntlm.c --- Implementation of non-standard SASL mechanism NTLM, client side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get malloc, free. */
00025 #include <stdlib.h>
00026
00027 /* Get memcpy. */
```

```
00028 #include <string.h>
00029
00030 /* Get specification. */
00031 #include "x-ntlm.h"
00032
00033 #include <ntlm.h>
00034
00035 struct _Gsasl_ntlm_state
00036 {
00037   int step;
00038 };
00039 typedef struct _Gsasl_ntlm_state _Gsasl_ntlm_state;
00040
00041 int
00042 _gsasl_ntlm_client_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
00043 {
00044   _Gsasl_ntlm_state *state;
00045
00046   state = (_Gsasl_ntlm_state *) malloc (sizeof (*state));
00047   if (state == NULL)
00048     return GSASL_MALLOC_ERROR;
00049
00050   state->step = 0;
00051
00052   *mech_data = state;
00053
00054   return GSASL_OK;
00055 }
00056
00057 int
00058 _gsasl_ntlm_client_step (Gsasl_session *sctx,
00059                          void *mech_data,
00060                          const char *input, size_t input_len,
00061                          char **output, size_t *output_len)
00062 {
00063   _Gsasl_ntlm_state *state = mech_data;
00064   const char *domain = gsasl_property_get (sctx, GSASL_REALM);
00065   const char *authid = gsasl_property_get (sctx, GSASL_AUTHID);
00066   const char *password;
00067   int res;
00068
00069   if (!authid)
00070     return GSASL_NO_AUTHID;
00071
00072   switch (state->step)
00073     {
00074     case 0:
00075       {
00076         tSmbNtlmAuthRequest *request;
00077
00078         request = malloc (sizeof (*request));
00079         if (!request)
00080           return GSASL_MALLOC_ERROR;
00081
00082         buildSmbNtlmAuthRequest (request, authid, domain);
00083
00084         *output_len = SmbLength (request);
00085         *output = malloc (*output_len);
00086         if (!*output)
00087           {
00088             free (request);
00089             return GSASL_MALLOC_ERROR;
00090           }
00091         memcpy (*output, request, *output_len);
00092
00093         free (request);
00094
00095         /* dumpSmbNtlmAuthRequest(stdout, &request); */
00096
00097         state->step++;
00098         res = GSASL_NEEDS_MORE;
00099         break;
00100       }
00101
00102     case 1:
00103       {
00104         tSmbNtlmAuthChallenge *challenge;
00105         tSmbNtlmAuthResponse *response;
00106
00107         if (input_len > sizeof (*challenge))
00108           return GSASL_MECHANISM_PARSE_ERROR;
00109
00110         challenge = malloc (sizeof (*challenge));
00111         if (!challenge)
00112           return GSASL_MALLOC_ERROR;
00113
00114         /* Hand crafted challenge for parser testing:
```

```
00115             TlRMTVNTUAAAAAAAAAAAAAAAAAAAAGFiY2RlZmdoMDEyMzQ1Njc4ODY2NDQwMTIz */
00116
00117         memcpy (challenge, input, input_len);
00118
00119         password = gsasl_property_get (sctx, GSASL_PASSWORD);
00120         if (!password)
00121           {
00122             free (challenge);
00123             return GSASL_NO_PASSWORD;
00124           }
00125
00126         response = malloc (sizeof (*response));
00127         if (!response)
00128           {
00129             free (challenge);
00130             return GSASL_MALLOC_ERROR;
00131           }
00132
00133         buildSmbNtlmAuthResponse (challenge, response, authid, password);
00134
00135         free (challenge);
00136
00137         *output_len = SmbLength (response);
00138         *output = malloc (*output_len);
00139         if (!*output)
00140           {
00141             free (response);
00142             return GSASL_MALLOC_ERROR;
00143           }
00144         memcpy (*output, response, *output_len);
00145
00146         free (response);
00147
00148         /* dumpSmbNtlmAuthResponse(stdout, &response); */
00149
00150         state->step++;
00151         res = GSASL_OK;
00152         break;
00153       }
00154
00155     default:
00156       res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00157       break;
00158     }
00159
00160   return res;
00161 }
00162
00163 void
00164 _gsasl_ntlm_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)
00165 {
00166   _Gsasl_ntlm_state *state = mech_data;
00167
00168   free (state);
00169 }
```

## 5.51 x-ntlm.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_NTLM_NAME "NTLM"

### Functions

- int _gsasl_ntlm_client_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_ntlm_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_ntlm_client_finish (Gsasl_session *sctx, void *mech_data)

**Variables**

- Gsasl_mechanism _gsasl_ntlm_mechanism

## 5.51.1 Macro Definition Documentation

### 5.51.1.1 GSASL_NTLM_NAME

```
#define GSASL_NTLM_NAME "NTLM"
```

Definition at line 27 of file x-ntlm.h.

## 5.51.2 Function Documentation

### 5.51.2.1 _gsasl_ntlm_client_finish()

```
void _gsasl_ntlm_client_finish (
             Gsasl_session * sctx,
             void * mech_data )  [extern]
```

### 5.51.2.2 _gsasl_ntlm_client_start()

```
int _gsasl_ntlm_client_start (
             Gsasl_session * sctx,
             void ** mech_data )  [extern]
```

### 5.51.2.3 _gsasl_ntlm_client_step()

```
int _gsasl_ntlm_client_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

Definition at line 58 of file ntlm.c.

## 5.51.3 Variable Documentation

### 5.51.3.1 _gsasl_ntlm_mechanism

```
Gsasl_mechanism _gsasl_ntlm_mechanism  [extern]
```

Definition at line 27 of file ntlm/mechinfo.c.

## 5.52 x-ntlm.h

Go to the documentation of this file.

```
00001 /* x-ntlm.h --- Prototypes for non-standard SASL mechanism NTLM.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef X_NTLM_H
00023 # define X_NTLM_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_NTLM_NAME "NTLM"
00028
00029 extern Gsasl_mechanism _gsasl_ntlm_mechanism;
00030
00031 extern int _gsasl_ntlm_client_start (Gsasl_session * sctx, void **mech_data);
00032 extern int _gsasl_ntlm_client_step (Gsasl_session * sctx,
00033                                     void *mech_data,
00034                                     const char *input, size_t input_len,
00035                                     char **output, size_t *output_len);
00036 extern void _gsasl_ntlm_client_finish (Gsasl_session * sctx, void *mech_data);
00037
00038 #endif /* X_NTLM_H */
```

## 5.53 openid20.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_OPENID20_NAME "OPENID20"

### Functions

- int _gsasl_openid20_client_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_openid20_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_openid20_client_finish (Gsasl_session *sctx, void *mech_data)
- int _gsasl_openid20_server_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_openid20_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_openid20_server_finish (Gsasl_session *sctx, void *mech_data)

### Variables

- Gsasl_mechanism _gsasl_openid20_mechanism

### 5.53.1 Macro Definition Documentation

#### 5.53.1.1 GSASL_OPENID20_NAME

```
#define GSASL_OPENID20_NAME "OPENID20"
```

Definition at line 27 of file openid20.h.

### 5.53.2 Function Documentation

#### 5.53.2.1 _gsasl_openid20_client_finish()

```
void _gsasl_openid20_client_finish (
             Gsasl_session * sctx,
             void * mech_data )  [extern]
```

#### 5.53.2.2 _gsasl_openid20_client_start()

```
int _gsasl_openid20_client_start (
             Gsasl_session * sctx,
             void ** mech_data )  [extern]
```

#### 5.53.2.3 _gsasl_openid20_client_step()

```
int _gsasl_openid20_client_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

Definition at line 60 of file openid20/client.c.

#### 5.53.2.4 _gsasl_openid20_server_finish()

```
void _gsasl_openid20_server_finish (
             Gsasl_session * sctx,
             void * mech_data )  [extern]
```

#### 5.53.2.5 _gsasl_openid20_server_start()

```
int _gsasl_openid20_server_start (
             Gsasl_session * sctx,
             void ** mech_data )  [extern]
```

#### 5.53.2.6 _gsasl_openid20_server_step()

```
int _gsasl_openid20_server_step (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )  [extern]
```

Definition at line 58 of file openid20/server.c.

### 5.53.3 Variable Documentation

#### 5.53.3.1 _gsasl_openid20_mechanism

```
Gsasl_mechanism _gsasl_openid20_mechanism  [extern]
```

Definition at line 27 of file openid20/mechinfo.c.

## 5.54 openid20.h

Go to the documentation of this file.

```
00001 /* openid20.h --- Prototypes for OPENID20.
00002  * Copyright (C) 2011-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef OPENID20_H
00023 # define OPENID20_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_OPENID20_NAME "OPENID20"
00028
00029 extern Gsasl_mechanism _gsasl_openid20_mechanism;
00030
00031 extern int _gsasl_openid20_client_start (Gsasl_session * sctx,
00032                                          void **mech_data);
00033
00034 extern int _gsasl_openid20_client_step (Gsasl_session * sctx,
00035                                         void *mech_data,
00036                                         const char *input, size_t input_len,
00037                                         char **output, size_t *output_len);
00038
00039 extern void _gsasl_openid20_client_finish (Gsasl_session * sctx,
00040                                            void *mech_data);
00041
00042 extern int _gsasl_openid20_server_start (Gsasl_session * sctx,
00043                                          void **mech_data);
00044
00045 extern int _gsasl_openid20_server_step (Gsasl_session * sctx,
```

```
00046                                       void *mech_data,
00047                                       const char *input, size_t input_len,
00048                                       char **output, size_t *output_len);
00049
00050 extern void _gsasl_openid20_server_finish (Gsasl_session * sctx,
00051                                            void *mech_data);
00052
00053 #endif /* OPENID20_H */
```

## 5.55 plain.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_PLAIN_NAME "PLAIN"

### Functions

- int _gsasl_plain_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- int _gsasl_plain_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)

### Variables

- Gsasl_mechanism _gsasl_plain_mechanism

### 5.55.1 Macro Definition Documentation

#### 5.55.1.1 GSASL_PLAIN_NAME

```
#define GSASL_PLAIN_NAME "PLAIN"
```

Definition at line 27 of file plain.h.

### 5.55.2 Function Documentation

#### 5.55.2.1 _gsasl_plain_client_step()

```
int _gsasl_plain_client_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

#### 5.55.2.2 _gsasl_plain_server_step()

```
int _gsasl_plain_server_step (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )  [extern]
```

### 5.55.3 Variable Documentation

#### 5.55.3.1 _gsasl_plain_mechanism

```
Gsasl_mechanism _gsasl_plain_mechanism  [extern]
```

Definition at line 27 of file plain/mechinfo.c.

## 5.56 plain.h

Go to the documentation of this file.

```
00001 /* plain.h --- Prototypes for SASL mechanism PLAIN as defined in RFC 2595.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef PLAIN_H
00023 # define PLAIN_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_PLAIN_NAME "PLAIN"
00028
00029 extern Gsasl_mechanism _gsasl_plain_mechanism;
00030
00031 extern int _gsasl_plain_client_step (Gsasl_session * sctx,
00032                                      void *mech_data,
00033                                      const char *input, size_t input_len,
00034                                      char **output, size_t *output_len);
00035
00036 extern int _gsasl_plain_server_step (Gsasl_session * sctx,
00037                                      void *mech_data,
00038                                      const char *input, size_t input_len,
00039                                      char **output, size_t *output_len);
00040
00041 #endif /* PLAIN_H */
```

## 5.57 anonymous/client.c File Reference

```
#include <config.h>
#include "anonymous.h"
#include <string.h>
```

### Functions

- int _gsasl_anonymous_client_step (Gsasl_session *sctx, void *mech_data _GL_UNUSED, const char *input _GL_UNUSED, size_t input_len _GL_UNUSED, char **output, size_t *output_len)

### 5.57.1 Function Documentation

#### 5.57.1.1 _gsasl_anonymous_client_step()

```
int _gsasl_anonymous_client_step (
            Gsasl_session * sctx,
            void *mech_data _GL_UNUSED,
            const char *input _GL_UNUSED,
            size_t input_len _GL_UNUSED,
            char ** output,
            size_t * output_len )
```

Definition at line 31 of file anonymous/client.c.

## 5.58 anonymous/client.c

Go to the documentation of this file.

```
00001 /* client.c --- ANONYMOUS mechanism as defined in RFC 2245, client side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "anonymous.h"
00026
00027 /* Get strdup, strlen. */
00028 #include <string.h>
00029
00030 int
00031 _gsasl_anonymous_client_step (Gsasl_session *sctx,
00032                               void *mech_data _GL_UNUSED,
00033                               const char *input _GL_UNUSED,
```

```
00034                                size_t input_len _GL_UNUSED,
00035                                char **output, size_t *output_len)
00036 {
00037   const char *p;
00038
00039   p = gsasl_property_get (sctx, GSASL_ANONYMOUS_TOKEN);
00040   if (!p)
00041     return GSASL_NO_ANONYMOUS_TOKEN;
00042
00043   *output = strdup (p);
00044   if (!*output)
00045     return GSASL_MALLOC_ERROR;
00046   *output_len = strlen (p);
00047
00048   return GSASL_OK;
00049 }
```

## 5.59 cram-md5/client.c File Reference

```
#include <config.h>
#include "cram-md5.h"
#include <stdlib.h>
#include <string.h>
#include "digest.h"
```

### Functions

- int _gsasl_cram_md5_client_step (Gsasl_session *sctx, void *mech_data _GL_UNUSED, const char *input, size_t input_len, char **output, size_t *output_len)

### 5.59.1 Function Documentation

#### 5.59.1.1 _gsasl_cram_md5_client_step()

```
int _gsasl_cram_md5_client_step (
             Gsasl_session * sctx,
             void *mech_data _GL_UNUSED,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 37 of file cram-md5/client.c.

## 5.60 cram-md5/client.c

Go to the documentation of this file.

```
00001 /* client.c --- SASL CRAM-MD5 client side functions.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
```

```
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "cram-md5.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get memcpy, strlen. */
00031 #include <string.h>
00032
00033 /* Get cram_md5_digest. */
00034 #include "digest.h"
00035
00036 int
00037 _gsasl_cram_md5_client_step (Gsasl_session *sctx,
00038                              void *mech_data _GL_UNUSED,
00039                              const char *input, size_t input_len,
00040                              char **output, size_t *output_len)
00041 {
00042   char response[CRAM_MD5_DIGEST_LEN];
00043   const char *p;
00044   size_t len;
00045   char *tmp;
00046   char *authid;
00047   int rc;
00048
00049   if (input_len == 0)
00050     {
00051       *output_len = 0;
00052       *output = NULL;
00053       return GSASL_NEEDS_MORE;
00054     }
00055
00056   p = gsasl_property_get (sctx, GSASL_AUTHID);
00057   if (!p)
00058     return GSASL_NO_AUTHID;
00059
00060   /* XXX Use query strings here?  Specification is unclear. */
00061   rc = gsasl_saslprep (p, GSASL_ALLOW_UNASSIGNED, &authid, NULL);
00062   if (rc != GSASL_OK)
00063     return rc;
00064
00065   p = gsasl_property_get (sctx, GSASL_PASSWORD);
00066   if (!p)
00067     {
00068       free (authid);
00069       return GSASL_NO_PASSWORD;
00070     }
00071
00072   /* XXX Use query strings here?  Specification is unclear. */
00073   rc = gsasl_saslprep (p, GSASL_ALLOW_UNASSIGNED, &tmp, NULL);
00074   if (rc != GSASL_OK)
00075     {
00076       free (authid);
00077       return rc;
00078     }
00079
00080   cram_md5_digest (input, input_len, tmp, strlen (tmp), response);
00081
00082   free (tmp);
00083
00084   len = strlen (authid);
00085
00086   *output_len = len + strlen (" ") + CRAM_MD5_DIGEST_LEN;
00087   *output = malloc (*output_len);
00088   if (!*output)
00089     {
00090       free (authid);
00091       return GSASL_MALLOC_ERROR;
00092     }
00093
00094   memcpy (*output, authid, len);
00095   (*output)[len++] = ' ';
00096   memcpy (*output + len, response, CRAM_MD5_DIGEST_LEN);
00097
```

```
00098   free (authid);
00099
00100   return GSASL_OK;
00101 }
```

## 5.61 digest-md5/client.c File Reference

```
#include <config.h>
#include "digest-md5.h"
#include <stdlib.h>
#include <string.h>
#include "gc.h"
#include "nonascii.h"
#include "tokens.h"
#include "parser.h"
#include "printer.h"
#include "free.h"
#include "session.h"
#include "digesthmac.h"
#include "qop.h"
#include "mechtools.h"
```

### Data Structures

- struct _Gsasl_digest_md5_client_state

### Macros

- #define CNONCE_ENTROPY_BYTES 16

### Typedefs

- typedef struct _Gsasl_digest_md5_client_state _Gsasl_digest_md5_client_state

### Functions

- int _gsasl_digest_md5_client_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
- int _gsasl_digest_md5_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input↩ _len, char **output, size_t *output_len)
- void _gsasl_digest_md5_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)
- int _gsasl_digest_md5_client_encode (Gsasl_session *sctx _GL_UNUSED, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- int _gsasl_digest_md5_client_decode (Gsasl_session *sctx _GL_UNUSED, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)

### 5.61.1 Macro Definition Documentation

#### 5.61.1.1 CNONCE_ENTROPY_BYTES

```
#define CNONCE_ENTROPY_BYTES 16
```

Definition at line 47 of file digest-md5/client.c.

## 5.61.2 Typedef Documentation

### 5.61.2.1 _Gsasl_digest_md5_client_state

```
typedef struct _Gsasl_digest_md5_client_state _Gsasl_digest_md5_client_state
```

Definition at line 62 of file digest-md5/client.c.

## 5.61.3 Function Documentation

### 5.61.3.1 _gsasl_digest_md5_client_decode()

```
int _gsasl_digest_md5_client_decode (
              Gsasl_session *sctx _GL_UNUSED,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 328 of file digest-md5/client.c.

### 5.61.3.2 _gsasl_digest_md5_client_encode()

```
int _gsasl_digest_md5_client_encode (
              Gsasl_session *sctx _GL_UNUSED,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 304 of file digest-md5/client.c.

### 5.61.3.3 _gsasl_digest_md5_client_finish()

```
void _gsasl_digest_md5_client_finish (
              Gsasl_session *sctx _GL_UNUSED,
              void * mech_data )
```

Definition at line 288 of file digest-md5/client.c.

### 5.61.3.4 _gsasl_digest_md5_client_start()

```
int _gsasl_digest_md5_client_start (
              Gsasl_session *sctx _GL_UNUSED,
              void ** mech_data )
```

Definition at line 65 of file digest-md5/client.c.

#### 5.61.3.5 _gsasl_digest_md5_client_step()

```
int _gsasl_digest_md5_client_step (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 97 of file digest-md5/client.c.

## 5.62 digest-md5/client.c

Go to the documentation of this file.

```
00001 /* client.c --- DIGEST-MD5 mechanism from RFC 2831, client side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "digest-md5.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get memcpy, strlen. */
00031 #include <string.h>
00032
00033 #include "gc.h"
00034
00035 /* Get tools. */
00036 #include "nonascii.h"
00037 #include "tokens.h"
00038 #include "parser.h"
00039 #include "printer.h"
00040 #include "free.h"
00041 #include "session.h"
00042 #include "digesthmac.h"
00043 #include "qop.h"
00044
00045 #include "mechtools.h"
00046
00047 #define CNONCE_ENTROPY_BYTES 16
00048
00049 struct _Gsasl_digest_md5_client_state
00050 {
00051   int step;
00052   unsigned long readseqnum, sendseqnum;
00053   char secret[DIGEST_MD5_LENGTH];
00054   char kic[DIGEST_MD5_LENGTH];
00055   char kcc[DIGEST_MD5_LENGTH];
00056   char kis[DIGEST_MD5_LENGTH];
00057   char kcs[DIGEST_MD5_LENGTH];
00058   digest_md5_challenge challenge;
00059   digest_md5_response response;
00060   digest_md5_finish finish;
```

```
00061 };
00062 typedef struct _Gsasl_digest_md5_client_state _Gsasl_digest_md5_client_state;
00063
00064 int
00065 _gsasl_digest_md5_client_start (Gsasl_session *sctx _GL_UNUSED,
00066                                 void **mech_data)
00067 {
00068   _Gsasl_digest_md5_client_state *state;
00069   char nonce[CNONCE_ENTROPY_BYTES];
00070   char *p;
00071   int rc;
00072
00073   rc = gsasl_nonce (nonce, CNONCE_ENTROPY_BYTES);
00074   if (rc != GSASL_OK)
00075     return rc;
00076
00077   rc = gsasl_base64_to (nonce, CNONCE_ENTROPY_BYTES, &p, NULL);
00078   if (rc != GSASL_OK)
00079     return rc;
00080
00081   state = calloc (1, sizeof (*state));
00082   if (state == NULL)
00083     {
00084       free (p);
00085       return GSASL_MALLOC_ERROR;
00086     }
00087
00088   state->response.cnonce = p;
00089   state->response.nc = 1;
00090
00091   *mech_data = state;
00092
00093   return GSASL_OK;
00094 }
00095
00096 int
00097 _gsasl_digest_md5_client_step (Gsasl_session *sctx,
00098                                void *mech_data,
00099                                const char *input,
00100                                size_t input_len,
00101                                char **output, size_t *output_len)
00102 {
00103   _Gsasl_digest_md5_client_state *state = mech_data;
00104   int rc, res;
00105
00106   *output = NULL;
00107   *output_len = 0;
00108
00109   if (state->step == 0)
00110     {
00111       state->step++;
00112       if (input_len == 0)
00113         return GSASL_NEEDS_MORE;
00114     }
00115
00116   switch (state->step)
00117     {
00118     case 1:
00119       {
00120         if (digest_md5_parse_challenge (input, input_len,
00121                                         &state->challenge) < 0)
00122           return GSASL_MECHANISM_PARSE_ERROR;
00123
00124         /* FIXME: How to let application know of remaining realms?
00125            One idea, add a GSASL_REALM_COUNT property, and have the
00126            GSASL_REALM be that many concatenated zero terminated realm
00127            strings.  Slightly hackish, though.  Another cleaner
00128            approach would be to add gsasl_property_set_array and
00129            gsasl_property_get_array APIs, for those properties that
00130            may be used multiple times. */
00131         if (state->challenge.nrealms > 0)
00132           res = gsasl_property_set (sctx, GSASL_REALM,
00133                                     state->challenge.realms[0]);
00134         else
00135           res = gsasl_property_set (sctx, GSASL_REALM, NULL);
00136         if (res != GSASL_OK)
00137           return res;
00138
00139         /* FIXME: cipher, maxbuf. */
00140
00141         /* Create response token. */
00142         state->response.utf8 = 1;
00143
00144         res = gsasl_property_set (sctx, GSASL_QOPS,
00145                                   digest_md5_qops2qopstr (state->
00146                                                           challenge.qops));
00147         if (res != GSASL_OK)
```

```
00148           return res;
00149
00150         {
00151           const char *qop = gsasl_property_get (sctx, GSASL_QOP);
00152
00153           if (!qop)
00154             state->response.qop = DIGEST_MD5_QOP_AUTH;
00155           else if (strcmp (qop, "qop-int") == 0)
00156             state->response.qop = DIGEST_MD5_QOP_AUTH_INT;
00157           else if (strcmp (qop, "qop-auth") == 0)
00158             state->response.qop = DIGEST_MD5_QOP_AUTH;
00159           else
00160             /* We don't support confidentiality or unknown
00161                keywords. */
00162             return GSASL_AUTHENTICATION_ERROR;
00163         }
00164
00165         state->response.nonce = strdup (state->challenge.nonce);
00166         if (!state->response.nonce)
00167           return GSASL_MALLOC_ERROR;
00168
00169         {
00170           const char *service = gsasl_property_get (sctx, GSASL_SERVICE);
00171           const char *hostname = gsasl_property_get (sctx, GSASL_HOSTNAME);
00172           if (!service)
00173             return GSASL_NO_SERVICE;
00174           if (!hostname)
00175             return GSASL_NO_HOSTNAME;
00176           if (asprintf (&state->response.digesturi, "%s/%s",
00177                         service, hostname) < 0)
00178             return GSASL_MALLOC_ERROR;
00179         }
00180
00181         {
00182           const char *c;
00183           char *tmp, *tmp2;
00184
00185           c = gsasl_property_get (sctx, GSASL_AUTHID);
00186           if (!c)
00187             return GSASL_NO_AUTHID;
00188
00189           state->response.username = strdup (c);
00190           if (!state->response.username)
00191             return GSASL_MALLOC_ERROR;
00192
00193           c = gsasl_property_get (sctx, GSASL_AUTHZID);
00194           if (c)
00195             {
00196               state->response.authzid = strdup (c);
00197               if (!state->response.authzid)
00198                 return GSASL_MALLOC_ERROR;
00199             }
00200
00201           gsasl_callback (NULL, sctx, GSASL_REALM);
00202           c = gsasl_property_fast (sctx, GSASL_REALM);
00203           if (c)
00204             {
00205               state->response.realm = strdup (c);
00206               if (!state->response.realm)
00207                 return GSASL_MALLOC_ERROR;
00208             }
00209
00210           c = gsasl_property_get (sctx, GSASL_PASSWORD);
00211           if (!c)
00212             return GSASL_NO_PASSWORD;
00213
00214           tmp2 = utf8tolatin1ifpossible (c);
00215
00216           rc = asprintf (&tmp, "%s:%s:%s", state->response.username,
00217                          state->response.realm ?
00218                          state->response.realm : "", tmp2);
00219           free (tmp2);
00220           if (rc < 0)
00221             return GSASL_MALLOC_ERROR;
00222
00223           rc = gc_md5 (tmp, strlen (tmp), state->secret);
00224           free (tmp);
00225           if (rc != GC_OK)
00226             return GSASL_CRYPTO_ERROR;
00227         }
00228
00229         rc = digest_md5_hmac (state->response.response,
00230                               state->secret,
00231                               state->response.nonce,
00232                               state->response.nc,
00233                               state->response.cnonce,
00234                               state->response.qop,
```

```
00235                               state->response.authzid,
00236                               state->response.digesturi,
00237                               0,
00238                               state->response.cipher,
00239                               state->kic, state->kis, state->kcc, state->kcs);
00240         if (rc)
00241           return GSASL_CRYPTO_ERROR;
00242
00243         *output = digest_md5_print_response (&state->response);
00244         if (!*output)
00245           return GSASL_AUTHENTICATION_ERROR;
00246
00247         *output_len = strlen (*output);
00248
00249         state->step++;
00250         res = GSASL_NEEDS_MORE;
00251       }
00252       break;
00253
00254     case 2:
00255       {
00256         char check[DIGEST_MD5_RESPONSE_LENGTH + 1];
00257
00258         if (digest_md5_parse_finish (input, input_len, &state->finish) < 0)
00259           return GSASL_MECHANISM_PARSE_ERROR;
00260
00261         res = digest_md5_hmac (check, state->secret,
00262                                state->response.nonce, state->response.nc,
00263                                state->response.cnonce, state->response.qop,
00264                                state->response.authzid,
00265                                state->response.digesturi, 1,
00266                                state->response.cipher, NULL, NULL, NULL,
00267                                NULL);
00268         if (res != GSASL_OK)
00269           break;
00270
00271         if (strcmp (state->finish.rspauth, check) == 0)
00272           res = GSASL_OK;
00273         else
00274           res = GSASL_AUTHENTICATION_ERROR;
00275         state->step++;
00276       }
00277       break;
00278
00279     default:
00280       res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00281       break;
00282     }
00283
00284   return res;
00285 }
00286
00287 void
00288 _gsasl_digest_md5_client_finish (Gsasl_session *sctx _GL_UNUSED,
00289                                  void *mech_data)
00290 {
00291   _Gsasl_digest_md5_client_state *state = mech_data;
00292
00293   if (!state)
00294     return;
00295
00296   digest_md5_free_challenge (&state->challenge);
00297   digest_md5_free_response (&state->response);
00298   digest_md5_free_finish (&state->finish);
00299
00300   free (state);
00301 }
00302
00303 int
00304 _gsasl_digest_md5_client_encode (Gsasl_session *sctx _GL_UNUSED,
00305                                  void *mech_data,
00306                                  const char *input,
00307                                  size_t input_len,
00308                                  char **output, size_t *output_len)
00309 {
00310   _Gsasl_digest_md5_client_state *state = mech_data;
00311   int res;
00312
00313   res = digest_md5_encode (input, input_len, output, output_len,
00314                            state->response.qop,
00315                            state->sendseqnum, state->kic);
00316   if (res)
00317     return res == -2 ? GSASL_NEEDS_MORE : GSASL_INTEGRITY_ERROR;
00318
00319   if (state->sendseqnum == 4294967295UL)
00320     state->sendseqnum = 0;
00321   else
```

```
00322     state->sendseqnum++;
00323
00324   return GSASL_OK;
00325 }
00326
00327 int
00328 _gsasl_digest_md5_client_decode (Gsasl_session *sctx _GL_UNUSED,
00329                                  void *mech_data,
00330                                  const char *input,
00331                                  size_t input_len,
00332                                  char **output, size_t *output_len)
00333 {
00334   _Gsasl_digest_md5_client_state *state = mech_data;
00335   int res;
00336
00337   res = digest_md5_decode (input, input_len, output, output_len,
00338                            state->response.qop,
00339                            state->readseqnum, state->kis);
00340   if (res)
00341     return res == -2 ? GSASL_NEEDS_MORE : GSASL_INTEGRITY_ERROR;
00342
00343   if (state->readseqnum == 4294967295UL)
00344     state->readseqnum = 0;
00345   else
00346     state->readseqnum++;
00347
00348   return GSASL_OK;
00349 }
```

## 5.63 external/client.c File Reference

```
#include <config.h>
#include "external.h"
#include <string.h>
```

### Functions

- int _gsasl_external_client_step (Gsasl_session *sctx, void *mech_data _GL_UNUSED, const char *input _GL_UNUSED, size_t input_len _GL_UNUSED, char **output, size_t *output_len)

### 5.63.1 Function Documentation

#### 5.63.1.1 _gsasl_external_client_step()

```
int _gsasl_external_client_step (
             Gsasl_session * sctx,
             void *mech_data _GL_UNUSED,
             const char *input _GL_UNUSED,
             size_t input_len _GL_UNUSED,
             char ** output,
             size_t * output_len )
```

Definition at line 31 of file external/client.c.

## 5.64 external/client.c

Go to the documentation of this file.

```
00001 /* client.c --- EXTERNAL mechanism as defined in RFC 2222, client side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "external.h"
00026
00027 /* Get strdup, strlen. */
00028 #include <string.h>
00029
00030 int
00031 _gsasl_external_client_step (Gsasl_session *sctx,
00032                              void *mech_data _GL_UNUSED,
00033                              const char *input _GL_UNUSED,
00034                              size_t input_len _GL_UNUSED,
00035                              char **output, size_t *output_len)
00036 {
00037   const char *p;
00038
00039   p = gsasl_property_get (sctx, GSASL_AUTHZID);
00040   if (!p)
00041     p = "";
00042
00043   *output = strdup (p);
00044   if (!*output)
00045     return GSASL_MALLOC_ERROR;
00046   *output_len = strlen (p);
00047
00048   return GSASL_OK;
00049 }
```

## 5.65 gs2/client.c File Reference

```
#include <config.h>
#include "gs2.h"
#include <stdlib.h>
#include <string.h>
#include "gss-extra.h"
#include "gs2helper.h"
```

### Data Structures

- struct _gsasl_gs2_client_state

### Typedefs

- typedef struct _gsasl_gs2_client_state _gsasl_gs2_client_state

### Functions

- int _gsasl_gs2_client_start (Gsasl_session ∗sctx, void ∗∗mech_data)
- int _gsasl_gs2_client_step (Gsasl_session ∗sctx, void ∗mech_data, const char ∗input, size_t input_len, char ∗∗output, size_t ∗output_len)
- void _gsasl_gs2_client_finish (Gsasl_session ∗sctx, void ∗mech_data)

## 5.65.1 Typedef Documentation

### 5.65.1.1 _gsasl_gs2_client_state

```
typedef struct _gsasl_gs2_client_state _gsasl_gs2_client_state
```

Definition at line 46 of file gs2/client.c.

## 5.65.2 Function Documentation

### 5.65.2.1 _gsasl_gs2_client_finish()

```
void _gsasl_gs2_client_finish (
            Gsasl_session * sctx,
            void * mech_data )
```

Definition at line 315 of file gs2/client.c.

### 5.65.2.2 _gsasl_gs2_client_start()

```
int _gsasl_gs2_client_start (
            Gsasl_session * sctx,
            void ** mech_data )
```

Definition at line 51 of file gs2/client.c.

### 5.65.2.3 _gsasl_gs2_client_step()

```
int _gsasl_gs2_client_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )
```

Definition at line 234 of file gs2/client.c.

## 5.66 gs2/client.c

Go to the documentation of this file.

```
00001 /* client.c --- SASL mechanism GS2, client side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "gs2.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get memcpy, strlen. */
00031 #include <string.h>
00032
00033 #include "gss-extra.h"
00034 #include "gs2helper.h"
00035
00036 struct _gsasl_gs2_client_state
00037 {
00038   /* steps: 0 = initial, 1 = first token, 2 = looping, 3 = done */
00039   int step;
00040   gss_name_t service;
00041   gss_ctx_id_t context;
00042   gss_OID mech_oid;
00043   gss_buffer_desc token;
00044   struct gss_channel_bindings_struct cb;
00045 };
00046 typedef struct _gsasl_gs2_client_state _gsasl_gs2_client_state;
00047
00048 /* Initialize GS2 state into MECH_DATA.  Return GSASL_OK if GS2 is
00049    ready and initialization succeeded, or an error code. */
00050 int
00051 _gsasl_gs2_client_start (Gsasl_session *sctx, void **mech_data)
00052 {
00053   _gsasl_gs2_client_state *state;
00054   int res;
00055
00056   state = (_gsasl_gs2_client_state *) malloc (sizeof (*state));
00057   if (state == NULL)
00058     return GSASL_MALLOC_ERROR;
00059
00060   res = gs2_get_oid (sctx, &state->mech_oid);
00061   if (res != GSASL_OK)
00062     {
00063       free (state);
00064       return res;
00065     }
00066
00067   state->step = 0;
00068   state->service = GSS_C_NO_NAME;
00069   state->context = GSS_C_NO_CONTEXT;
00070   state->token.length = 0;
00071   state->token.value = NULL;
00072   /* The initiator-address-type and acceptor-address-type fields of
00073      the GSS-CHANNEL-BINDINGS structure MUST be set to 0.  The
00074      initiator-address and acceptor-address fields MUST be the empty
00075      string. */
00076   state->cb.initiator_addrtype = 0;
00077   state->cb.initiator_address.length = 0;
00078   state->cb.initiator_address.value = NULL;
00079   state->cb.acceptor_addrtype = 0;
00080   state->cb.acceptor_address.length = 0;
00081   state->cb.acceptor_address.value = NULL;
00082   state->cb.application_data.length = 0;
```

```
00083   state->cb.application_data.value = NULL;
00084
00085   *mech_data = state;
00086
00087   return GSASL_OK;
00088 }
00089
00090 /* Return newly allocated copy of STR with all occurrences of ',' 
00091    replaced with =2C and '=' with '=3D', or return NULL on memory
00092    allocation errors.  */
00093 static char *
00094 escape_authzid (const char *str)
00095 {
00096   char *out = malloc (strlen (str) * 3 + 1);
00097   char *p = out;
00098
00099   if (!out)
00100     return NULL;
00101
00102   while (*str)
00103     {
00104       if (*str == ',')
00105         {
00106           memcpy (p, "=2C", 3);
00107           p += 3;
00108         }
00109       else if (*str == '=')
00110         {
00111           memcpy (p, "=3D", 3);
00112           p += 3;
00113         }
00114       else
00115         {
00116           *p = *str;
00117           p++;
00118         }
00119       str++;
00120     }
00121   *p = '\0';
00122
00123   return out;
00124 }
00125
00126 /* Get service, hostname and authorization identity from application,
00127    import the GSS-API name, and initialize the channel binding data.
00128    Return GSASL_OK on success or an error code. */
00129 static int
00130 prepare (Gsasl_session *sctx, _gsasl_gs2_client_state *state)
00131 {
00132   const char *service = gsasl_property_get (sctx, GSASL_SERVICE);
00133   const char *hostname = gsasl_property_get (sctx, GSASL_HOSTNAME);
00134   const char *authzid = gsasl_property_get (sctx, GSASL_AUTHZID);
00135   gss_buffer_desc bufdesc;
00136   OM_uint32 maj_stat, min_stat;
00137
00138   if (!service)
00139     return GSASL_NO_SERVICE;
00140   if (!hostname)
00141     return GSASL_NO_HOSTNAME;
00142
00143   bufdesc.length = asprintf ((char **) &bufdesc.value, "%s@%s",
00144                              service, hostname);
00145   if (bufdesc.length <= 0 || bufdesc.value == NULL)
00146     return GSASL_MALLOC_ERROR;
00147
00148   maj_stat = gss_import_name (&min_stat, &bufdesc,
00149                               GSS_C_NT_HOSTBASED_SERVICE, &state->service);
00150   free (bufdesc.value);
00151   if (GSS_ERROR (maj_stat))
00152     return GSASL_GSSAPI_IMPORT_NAME_ERROR;
00153
00154   if (authzid)
00155     {
00156       char *escaped_authzid = escape_authzid (authzid);
00157
00158       if (!escaped_authzid)
00159         return GSASL_MALLOC_ERROR;
00160
00161       state->cb.application_data.length
00162         = asprintf ((char **) &state->cb.application_data.value,
00163                     "n,a=%s,", escaped_authzid);
00164
00165       free (escaped_authzid);
00166     }
00167   else
00168     {
00169       state->cb.application_data.value = strdup ("n,,");
```

```
00170       state->cb.application_data.length = 3;
00171     }
00172
00173   if (state->cb.application_data.length <= 0
00174       || state->cb.application_data.value == NULL)
00175     return GSASL_MALLOC_ERROR;
00176
00177   return GSASL_OK;
00178 }
00179
00180 /* Copy token to output buffer.  On first round trip, strip context
00181    token header and add channel binding data. For later round trips,
00182    just copy the buffer.  Return GSASL_OK on success or an error
00183    code.  */
00184 static int
00185 token2output (_gsasl_gs2_client_state *state,
00186               const gss_buffer_t token, char **output, size_t *output_len)
00187 {
00188   OM_uint32 maj_stat, min_stat;
00189   gss_buffer_desc bufdesc;
00190
00191   if (state->step == 1)
00192     {
00193       state->step++;
00194
00195       maj_stat = gss_decapsulate_token (token, state->mech_oid, &bufdesc);
00196       if (GSS_ERROR (maj_stat))
00197         return GSASL_GSSAPI_ENCAPSULATE_TOKEN_ERROR;
00198
00199       *output_len = state->cb.application_data.length + bufdesc.length;
00200       *output = malloc (*output_len);
00201       if (!*output)
00202         {
00203           gss_release_buffer (&min_stat, &bufdesc);
00204           return GSASL_MALLOC_ERROR;
00205         }
00206
00207       memcpy (*output, state->cb.application_data.value,
00208               state->cb.application_data.length);
00209       memcpy (*output + state->cb.application_data.length,
00210               bufdesc.value, bufdesc.length);
00211
00212       maj_stat = gss_release_buffer (&min_stat, &bufdesc);
00213       if (GSS_ERROR (maj_stat))
00214         return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00215     }
00216   else
00217     {
00218       *output_len = token->length;
00219       *output = malloc (*output_len);
00220       if (!*output)
00221         return GSASL_MALLOC_ERROR;
00222       if (token->value)
00223         memcpy (*output, token->value, token->length);
00224     }
00225
00226   return GSASL_OK;
00227 }
00228
00229 /* Perform one GS2 step.  GS2 state is in MECH_DATA.  Any data from
00230    server is provided in INPUT/INPUT_LEN and output from client is
00231    expected to be put in newly allocated OUTPUT/OUTPUT_LEN.  Return
00232    GSASL_NEEDS_MORE or GSASL_OK on success, or an error code.  */
00233 int
00234 _gsasl_gs2_client_step (Gsasl_session *sctx,
00235                         void *mech_data,
00236                         const char *input, size_t input_len,
00237                         char **output, size_t *output_len)
00238 {
00239   _gsasl_gs2_client_state *state = mech_data;
00240   gss_buffer_desc bufdesc;
00241   gss_buffer_t buf = GSS_C_NO_BUFFER;
00242   OM_uint32 maj_stat, min_stat, ret_flags;
00243   gss_OID actual_mech_type;
00244   int res;
00245
00246   if (state->step > 2)
00247     return GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00248
00249   if (state->step == 0)
00250     {
00251       res = prepare (sctx, state);
00252       if (res != GSASL_OK)
00253         return res;
00254       state->step++;
00255     }
00256
```

```
00257   if (state->step == 2)
00258     {
00259       bufdesc.length = input_len;
00260       bufdesc.value = (void *) input;
00261       buf = &bufdesc;
00262     }
00263
00264   /* First release memory for token from last round-trip, if any. */
00265   if (state->token.value != NULL)
00266     {
00267       maj_stat = gss_release_buffer (&min_stat, &state->token);
00268       if (GSS_ERROR (maj_stat))
00269         return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00270
00271       state->token.value = NULL;
00272       state->token.length = 0;
00273     }
00274
00275   maj_stat = gss_init_sec_context (&min_stat,
00276                                    GSS_C_NO_CREDENTIAL,
00277                                    &state->context,
00278                                    state->service,
00279                                    state->mech_oid,
00280                                    GSS_C_MUTUAL_FLAG,
00281                                    0,
00282                                    &state->cb,
00283                                    buf,
00284                                    &actual_mech_type,
00285                                    &state->token, &ret_flags, NULL);
00286   if (maj_stat != GSS_S_COMPLETE && maj_stat != GSS_S_CONTINUE_NEEDED)
00287     return GSASL_GSSAPI_INIT_SEC_CONTEXT_ERROR;
00288
00289   if (state->token.length > 0 && state->token.value == NULL)
00290     return GSASL_GSSAPI_INIT_SEC_CONTEXT_ERROR;
00291
00292   res = token2output (state, &state->token, output, output_len);
00293   if (res != GSASL_OK)
00294     return res;
00295
00296   if (maj_stat == GSS_S_CONTINUE_NEEDED)
00297     return GSASL_NEEDS_MORE;
00298
00299   /* The GSS-API layer is done here, check that we established a valid
00300      security context for GS2 purposes. */
00301
00302   if (!(ret_flags & GSS_C_MUTUAL_FLAG))
00303     return GSASL_AUTHENTICATION_ERROR;
00304
00305   if (!gss_oid_equal (state->mech_oid, actual_mech_type))
00306     return GSASL_AUTHENTICATION_ERROR;
00307
00308   state->step++;
00309   return GSASL_OK;
00310 }
00311
00312 /* Cleanup GS2 state context, i.e., release memory associated with
00313    buffers in MECH_DATA state. */
00314 void
00315 _gsasl_gs2_client_finish (Gsasl_session *sctx, void *mech_data)
00316 {
00317   _gsasl_gs2_client_state *state = mech_data;
00318   OM_uint32 min_stat;
00319   (void) sctx;
00320
00321   if (!state)
00322     return;
00323
00324   if (state->token.value != NULL)
00325     gss_release_buffer (&min_stat, &state->token);
00326   if (state->service != GSS_C_NO_NAME)
00327     gss_release_name (&min_stat, &state->service);
00328   if (state->context != GSS_C_NO_CONTEXT)
00329     gss_delete_sec_context (&min_stat, &state->context, GSS_C_NO_BUFFER);
00330
00331   free (state->cb.application_data.value);
00332   free (state);
00333 }
```

## 5.67 gssapi/client.c File Reference

```
#include <config.h>
#include <stdlib.h>
```

```
#include <string.h>
#include "x-gssapi.h"
#include "gss-extra.h"
```

#### Data Structures

- struct _Gsasl_gssapi_client_state

#### Typedefs

- typedef struct _Gsasl_gssapi_client_state _Gsasl_gssapi_client_state

#### Functions

- int _gsasl_gssapi_client_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_gssapi_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_gssapi_client_finish (Gsasl_session *sctx, void *mech_data)
- int _gsasl_gssapi_client_encode (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- int _gsasl_gssapi_client_decode (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)

### 5.67.1 Typedef Documentation

#### 5.67.1.1 _Gsasl_gssapi_client_state

```
typedef struct _Gsasl_gssapi_client_state _Gsasl_gssapi_client_state
```

Definition at line 43 of file gssapi/client.c.

### 5.67.2 Function Documentation

#### 5.67.2.1 _gsasl_gssapi_client_decode()

```
int _gsasl_gssapi_client_decode (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )
```

Definition at line 321 of file gssapi/client.c.

#### 5.67.2.2 _gsasl_gssapi_client_encode()

```
int _gsasl_gssapi_client_encode (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 266 of file gssapi/client.c.

#### 5.67.2.3 _gsasl_gssapi_client_finish()

```
void _gsasl_gssapi_client_finish (
              Gsasl_session * sctx,
              void * mech_data )
```

Definition at line 248 of file gssapi/client.c.

#### 5.67.2.4 _gsasl_gssapi_client_start()

```
int _gsasl_gssapi_client_start (
              Gsasl_session * sctx,
              void ** mech_data )
```

Definition at line 46 of file gssapi/client.c.

#### 5.67.2.5 _gsasl_gssapi_client_step()

```
int _gsasl_gssapi_client_step (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 65 of file gssapi/client.c.

## 5.68 gssapi/client.c

Go to the documentation of this file.

```
00001 /* client.c --- SASL mechanism GSSAPI as defined in RFC 4752, client side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get malloc, free. */
00025 #include <stdlib.h>
00026
00027 /* Get memcpy, strlen. */
00028 #include <string.h>
00029
00030 /* Get specification. */
00031 #include "x-gssapi.h"
00032
00033 /* For GSS-API prototypes. */
00034 #include "gss-extra.h"
00035
00036 struct _Gsasl_gssapi_client_state
00037 {
00038   int step;
00039   gss_name_t service;
00040   gss_ctx_id_t context;
00041   gss_qop_t qop;
00042 };
00043 typedef struct _Gsasl_gssapi_client_state _Gsasl_gssapi_client_state;
00044
00045 int
00046 _gsasl_gssapi_client_start (Gsasl_session *sctx, void **mech_data)
00047 {
00048   _Gsasl_gssapi_client_state *state;
00049
00050   state = (_Gsasl_gssapi_client_state *) malloc (sizeof (*state));
00051   if (state == NULL)
00052     return GSASL_MALLOC_ERROR;
00053
00054   state->context = GSS_C_NO_CONTEXT;
00055   state->service = GSS_C_NO_NAME;
00056   state->step = 0;
00057   state->qop = GSASL_QOP_AUTH; /* FIXME: Should be GSASL_QOP_AUTH_CONF. */
00058
00059   *mech_data = state;
00060
00061   return GSASL_OK;
00062 }
00063
00064 int
00065 _gsasl_gssapi_client_step (Gsasl_session *sctx,
00066                            void *mech_data,
00067                            const char *input, size_t input_len,
00068                            char **output, size_t *output_len)
00069 {
00070   _Gsasl_gssapi_client_state *state = mech_data;
00071   char clientwrap[4];
00072   gss_qop_t serverqop;
00073   gss_buffer_desc bufdesc, bufdesc2;
00074   gss_buffer_t buf = GSS_C_NO_BUFFER;
00075   OM_uint32 maj_stat, min_stat;
00076   int conf_state;
00077   int res;
00078   const char *p;
00079
00080   if (state->service == NULL)
00081     {
00082       const char *service, *hostname;
```

```
00083
00084       service = gsasl_property_get (sctx, GSASL_SERVICE);
00085       if (!service)
00086         return GSASL_NO_SERVICE;
00087
00088       hostname = gsasl_property_get (sctx, GSASL_HOSTNAME);
00089       if (!hostname)
00090         return GSASL_NO_HOSTNAME;
00091
00092       /* FIXME: Use asprintf. */
00093
00094       bufdesc.length = strlen (service) + 1 + strlen (hostname) + 1;
00095       bufdesc.value = malloc (bufdesc.length);
00096       if (bufdesc.value == NULL)
00097         return GSASL_MALLOC_ERROR;
00098
00099       sprintf (bufdesc.value, "%s@%s", service, hostname);
00100
00101       maj_stat = gss_import_name (&min_stat, &bufdesc,
00102                                   GSS_C_NT_HOSTBASED_SERVICE,
00103                                   &state->service);
00104       free (bufdesc.value);
00105       if (GSS_ERROR (maj_stat))
00106         return GSASL_GSSAPI_IMPORT_NAME_ERROR;
00107     }
00108
00109   switch (state->step)
00110     {
00111     case 1:
00112       bufdesc.length = input_len;
00113       bufdesc.value = (void *) input;
00114       buf = &bufdesc;
00115       /* fall through */
00116
00117     case 0:
00118       bufdesc2.length = 0;
00119       bufdesc2.value = NULL;
00120       maj_stat = gss_init_sec_context (&min_stat,
00121                                        GSS_C_NO_CREDENTIAL,
00122                                        &state->context,
00123                                        state->service,
00124                                        GSS_C_NO_OID,
00125                                        GSS_C_MUTUAL_FLAG |
00126                                        GSS_C_REPLAY_FLAG |
00127                                        GSS_C_SEQUENCE_FLAG |
00128                                        GSS_C_INTEG_FLAG |
00129                                        GSS_C_CONF_FLAG,
00130                                        0,
00131                                        GSS_C_NO_CHANNEL_BINDINGS,
00132                                        buf, NULL, &bufdesc2, NULL, NULL);
00133       if (maj_stat != GSS_S_COMPLETE && maj_stat != GSS_S_CONTINUE_NEEDED)
00134         return GSASL_GSSAPI_INIT_SEC_CONTEXT_ERROR;
00135
00136       if (bufdesc2.length > 0 && bufdesc2.value == NULL)
00137         return GSASL_GSSAPI_INIT_SEC_CONTEXT_ERROR;
00138
00139       *output_len = bufdesc2.length;
00140       *output = malloc (*output_len);
00141       if (!*output)
00142         return GSASL_MALLOC_ERROR;
00143       if (bufdesc2.value)
00144         memcpy (*output, bufdesc2.value, bufdesc2.length);
00145
00146       if (maj_stat == GSS_S_COMPLETE)
00147         state->step = 2;
00148       else
00149         state->step = 1;
00150
00151       maj_stat = gss_release_buffer (&min_stat, &bufdesc2);
00152       if (maj_stat != GSS_S_COMPLETE)
00153         return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00154
00155       res = GSASL_NEEDS_MORE;
00156       break;
00157
00158     case 2:
00159       /* [RFC 2222 section 7.2.1]:
00160          The client passes this token to GSS_Unwrap and interprets the
00161          first octet of resulting cleartext as a bit-mask specifying
00162          the security layers supported by the server and the second
00163          through fourth octets as the maximum size output_message to
00164          send to the server.  The client then constructs data, with
00165          the first octet containing the bit-mask specifying the
00166          selected security layer, the second through fourth octets
00167          containing in network byte order the maximum size
00168          output_message the client is able to receive, and the
00169          remaining octets containing the authorization identity.  The
```

```
00170          client passes the data to GSS_Wrap with conf_flag set to
00171          FALSE, and responds with the generated output_message.  The
00172          client can then consider the server authenticated. */
00173
00174       bufdesc.length = input_len;
00175       bufdesc.value = (void *) input;
00176       maj_stat = gss_unwrap (&min_stat, state->context, &bufdesc,
00177                              &bufdesc2, &conf_state, &serverqop);
00178       if (GSS_ERROR (maj_stat))
00179         return GSASL_GSSAPI_UNWRAP_ERROR;
00180
00181       if (bufdesc2.length != 4)
00182         return GSASL_MECHANISM_PARSE_ERROR;
00183
00184       memcpy (clientwrap, bufdesc2.value, 4);
00185
00186       maj_stat = gss_release_buffer (&min_stat, &bufdesc2);
00187       if (GSS_ERROR (maj_stat))
00188         return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00189
00190 #if 0
00191       /* FIXME: Fix qop. */
00192       if (cb_qop)
00193         state->qop = cb_qop (sctx, serverqop);
00194
00195       if ((state->qop & serverqop) == 0)
00196         /*  Server does not support what user wanted. */
00197         return GSASL_GSSAPI_UNSUPPORTED_PROTECTION_ERROR;
00198 #endif
00199
00200       /* FIXME: Fix maxbuf. */
00201
00202       p = gsasl_property_get (sctx, GSASL_AUTHZID);
00203       if (!p)
00204         p = "";
00205
00206       bufdesc.length = 4 + strlen (p);
00207       bufdesc.value = malloc (bufdesc.length);
00208       if (!bufdesc.value)
00209         return GSASL_MALLOC_ERROR;
00210
00211       {
00212         char *q = bufdesc.value;
00213         q[0] = state->qop;
00214         memcpy (q + 1, clientwrap + 1, 3);
00215         memcpy (q + 4, p, strlen (p));
00216       }
00217
00218       maj_stat = gss_wrap (&min_stat, state->context, 0, GSS_C_QOP_DEFAULT,
00219                            &bufdesc, &conf_state, &bufdesc2);
00220       free (bufdesc.value);
00221       if (GSS_ERROR (maj_stat))
00222         return GSASL_GSSAPI_WRAP_ERROR;
00223
00224       *output_len = bufdesc2.length;
00225       *output = malloc (bufdesc2.length);
00226       if (!*output)
00227         return GSASL_MALLOC_ERROR;
00228
00229       memcpy (*output, bufdesc2.value, bufdesc2.length);
00230
00231       maj_stat = gss_release_buffer (&min_stat, &bufdesc2);
00232       if (GSS_ERROR (maj_stat))
00233         return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00234
00235       state->step++;
00236       res = GSASL_OK;
00237       break;
00238
00239     default:
00240       res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00241       break;
00242     }
00243
00244   return res;
00245 }
00246
00247 void
00248 _gsasl_gssapi_client_finish (Gsasl_session *sctx, void *mech_data)
00249 {
00250   _Gsasl_gssapi_client_state *state = mech_data;
00251   OM_uint32 maj_stat, min_stat;
00252
00253   if (!state)
00254     return;
00255
00256   if (state->service != GSS_C_NO_NAME)
```

```
00257     maj_stat = gss_release_name (&min_stat, &state->service);
00258   if (state->context != GSS_C_NO_CONTEXT)
00259     maj_stat = gss_delete_sec_context (&min_stat, &state->context,
00260                                        GSS_C_NO_BUFFER);
00261
00262   free (state);
00263 }
00264
00265 int
00266 _gsasl_gssapi_client_encode (Gsasl_session *sctx,
00267                              void *mech_data,
00268                              const char *input, size_t input_len,
00269                              char **output, size_t *output_len)
00270 {
00271   _Gsasl_gssapi_client_state *state = mech_data;
00272   OM_uint32 min_stat, maj_stat;
00273   gss_buffer_desc foo;
00274   gss_buffer_t input_message_buffer = &foo;
00275   gss_buffer_desc output_message_buffer;
00276
00277   foo.length = input_len;
00278   foo.value = (void *) input;
00279
00280   if (state && state->step == 3 &&
00281       state->qop & (GSASL_QOP_AUTH_INT | GSASL_QOP_AUTH_CONF))
00282     {
00283       maj_stat = gss_wrap (&min_stat,
00284                            state->context,
00285                            state->qop & GSASL_QOP_AUTH_CONF ? 1 : 0,
00286                            GSS_C_QOP_DEFAULT,
00287                            input_message_buffer,
00288                            NULL, &output_message_buffer);
00289       if (GSS_ERROR (maj_stat))
00290         return GSASL_GSSAPI_WRAP_ERROR;
00291       *output_len = output_message_buffer.length;
00292       *output = malloc (output_message_buffer.length);
00293       if (!*output)
00294         {
00295           maj_stat = gss_release_buffer (&min_stat, &output_message_buffer);
00296           return GSASL_MALLOC_ERROR;
00297         }
00298       memcpy (*output, output_message_buffer.value,
00299               output_message_buffer.length);
00300
00301       maj_stat = gss_release_buffer (&min_stat, &output_message_buffer);
00302       if (GSS_ERROR (maj_stat))
00303         {
00304           free (*output);
00305           return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00306         }
00307     }
00308   else
00309     {
00310       *output_len = input_len;
00311       *output = malloc (input_len);
00312       if (!*output)
00313         return GSASL_MALLOC_ERROR;
00314       memcpy (*output, input, input_len);
00315     }
00316
00317   return GSASL_OK;
00318 }
00319
00320 int
00321 _gsasl_gssapi_client_decode (Gsasl_session *sctx,
00322                              void *mech_data,
00323                              const char *input, size_t input_len,
00324                              char **output, size_t *output_len)
00325 {
00326   _Gsasl_gssapi_client_state *state = mech_data;
00327   OM_uint32 min_stat, maj_stat;
00328   gss_buffer_desc foo;
00329   gss_buffer_t input_message_buffer = &foo;
00330   gss_buffer_desc output_message_buffer;
00331
00332   foo.length = input_len;
00333   foo.value = (void *) input;
00334
00335   if (state && state->step == 3 &&
00336       state->qop & (GSASL_QOP_AUTH_INT | GSASL_QOP_AUTH_CONF))
00337     {
00338       maj_stat = gss_unwrap (&min_stat,
00339                              state->context,
00340                              input_message_buffer,
00341                              &output_message_buffer, NULL, NULL);
00342       if (GSS_ERROR (maj_stat))
00343         return GSASL_GSSAPI_UNWRAP_ERROR;
```

```
00344       *output_len = output_message_buffer.length;
00345       *output = malloc (output_message_buffer.length);
00346       if (!*output)
00347         {
00348           maj_stat = gss_release_buffer (&min_stat, &output_message_buffer);
00349           return GSASL_MALLOC_ERROR;
00350         }
00351       memcpy (*output, output_message_buffer.value,
00352               output_message_buffer.length);
00353
00354       maj_stat = gss_release_buffer (&min_stat, &output_message_buffer);
00355       if (GSS_ERROR (maj_stat))
00356         {
00357           free (*output);
00358           return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00359         }
00360     }
00361   else
00362     {
00363       *output_len = input_len;
00364       *output = malloc (input_len);
00365       if (!*output)
00366         return GSASL_MALLOC_ERROR;
00367       memcpy (*output, input, input_len);
00368     }
00369
00370   return GSASL_OK;
00371 }
```

## 5.69 login/client.c File Reference

```
#include <config.h>
#include <stdlib.h>
#include <string.h>
#include "login.h"
```

### Data Structures

- struct _Gsasl_login_client_state

### Functions

- int _gsasl_login_client_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
- int _gsasl_login_client_step (Gsasl_session *sctx _GL_UNUSED, void *mech_data, const char *input _↩ GL_UNUSED, size_t input_len _GL_UNUSED, char **output, size_t *output_len)
- void _gsasl_login_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)

### 5.69.1 Function Documentation

#### 5.69.1.1 _gsasl_login_client_finish()

```
void _gsasl_login_client_finish (
            Gsasl_session *sctx _GL_UNUSED,
            void * mech_data )
```

Definition at line 102 of file login/client.c.

#### 5.69.1.2 _gsasl_login_client_start()

```
int _gsasl_login_client_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 39 of file login/client.c.

#### 5.69.1.3 _gsasl_login_client_step()

```
int _gsasl_login_client_step (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data,
             const char *input _GL_UNUSED,
             size_t input_len _GL_UNUSED,
             char ** output,
             size_t * output_len )
```

Definition at line 55 of file login/client.c.

## 5.70 login/client.c

Go to the documentation of this file.

```
00001 /* client.c --- Non-standard SASL mechanism LOGIN, client side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get malloc, free. */
00025 #include <stdlib.h>
00026
00027 /* Get strlen. */
00028 #include <string.h>
00029
00030 /* Get specification. */
00031 #include "login.h"
00032
00033 struct _Gsasl_login_client_state
00034 {
00035   int step;
00036 };
00037
00038 int
00039 _gsasl_login_client_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
00040 {
00041   struct _Gsasl_login_client_state *state;
00042
00043   state = malloc (sizeof (*state));
00044   if (state == NULL)
00045     return GSASL_MALLOC_ERROR;
00046
```

```
00047   state->step = 0;
00048
00049   *mech_data = state;
00050
00051   return GSASL_OK;
00052 }
00053
00054 int
00055 _gsasl_login_client_step (Gsasl_session *sctx _GL_UNUSED,
00056                           void *mech_data,
00057                           const char *input _GL_UNUSED,
00058                           size_t input_len _GL_UNUSED,
00059                           char **output, size_t *output_len)
00060 {
00061   struct _Gsasl_login_client_state *state = mech_data;
00062   const char *p;
00063   int res;
00064
00065   switch (state->step)
00066     {
00067     case 0:
00068       p = gsasl_property_get (sctx, GSASL_AUTHID);
00069       if (!p)
00070         return GSASL_NO_AUTHID;
00071
00072       *output = strdup (p);
00073       *output_len = strlen (p);
00074
00075       state->step++;
00076       res = GSASL_NEEDS_MORE;
00077       break;
00078
00079     case 1:
00080       p = gsasl_property_get (sctx, GSASL_PASSWORD);
00081       if (!p)
00082         return GSASL_NO_PASSWORD;
00083
00084       *output = strdup (p);
00085       if (!*output)
00086         return GSASL_MALLOC_ERROR;
00087       *output_len = strlen (*output);
00088
00089       state->step++;
00090       res = GSASL_OK;
00091       break;
00092
00093     default:
00094       res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00095       break;
00096     }
00097
00098   return res;
00099 }
00100
00101 void
00102 _gsasl_login_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)
00103 {
00104   struct _Gsasl_login_client_state *state = mech_data;
00105
00106   if (!state)
00107     return;
00108
00109   free (state);
00110 }
```

## 5.71 openid20/client.c File Reference

```
#include <config.h>
#include "openid20.h"
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include "mechtools.h"
```

### Data Structures

- struct openid20_client_state

### Macros

- #define ERR_PREFIX "openid.error="

### Functions

- int _gsasl_openid20_client_start (Gsasl_session ∗sctx _GL_UNUSED, void ∗∗mech_data)
- int _gsasl_openid20_client_step (Gsasl_session ∗sctx, void ∗mech_data, const char ∗input, size_t input_len, char ∗∗output, size_t ∗output_len)
- void _gsasl_openid20_client_finish (Gsasl_session ∗sctx _GL_UNUSED, void ∗mech_data)

## 5.71.1 Macro Definition Documentation

### 5.71.1.1 ERR_PREFIX

```
#define ERR_PREFIX "openid.error="
```

## 5.71.2 Function Documentation

### 5.71.2.1 _gsasl_openid20_client_finish()

```
void _gsasl_openid20_client_finish (
            Gsasl_session *sctx _GL_UNUSED,
            void * mech_data )
```

Definition at line 166 of file openid20/client.c.

### 5.71.2.2 _gsasl_openid20_client_start()

```
int _gsasl_openid20_client_start (
            Gsasl_session *sctx _GL_UNUSED,
            void ** mech_data )
```

Definition at line 45 of file openid20/client.c.

### 5.71.2.3 _gsasl_openid20_client_step()

```
int _gsasl_openid20_client_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )
```

Definition at line 60 of file openid20/client.c.

## 5.72 openid20/client.c

Go to the documentation of this file.

```
00001 /* client.c --- OPENID20 mechanism, client side.
00002  * Copyright (C) 2011-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "openid20.h"
00026
00027 /* Get strdup, strlen. */
00028 #include <string.h>
00029
00030 /* Get calloc, free. */
00031 #include <stdlib.h>
00032
00033 /* Get bool. */
00034 #include <stdbool.h>
00035
00036 /* Get _gsasl_gs2_generate_header. */
00037 #include "mechtools.h"
00038
00039 struct openid20_client_state
00040 {
00041   int step;
00042 };
00043
00044 int
00045 _gsasl_openid20_client_start (Gsasl_session *sctx _GL_UNUSED,
00046                               void **mech_data)
00047 {
00048   struct openid20_client_state *state;
00049
00050   state = (struct openid20_client_state *) calloc (1, sizeof (*state));
00051   if (state == NULL)
00052     return GSASL_MALLOC_ERROR;
00053
00054   *mech_data = state;
00055
00056   return GSASL_OK;
00057 }
00058
00059 int
00060 _gsasl_openid20_client_step (Gsasl_session *sctx,
00061                              void *mech_data,
00062                              const char *input, size_t input_len,
00063                              char **output, size_t *output_len)
00064 {
00065   struct openid20_client_state *state = mech_data;
00066   int res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00067
00068   switch (state->step)
00069     {
00070     case 0:
00071       {
00072         const char *authzid = gsasl_property_get (sctx, GSASL_AUTHZID);
00073         const char *authid = gsasl_property_get (sctx, GSASL_AUTHID);
00074
00075         if (!authid || !*authid)
00076           return GSASL_NO_AUTHID;
00077
00078         res = _gsasl_gs2_generate_header (false, 'n', NULL, authzid,
00079                                           strlen (authid), authid,
00080                                           output, output_len);
00081         if (res != GSASL_OK)
00082           return res;
```

```
00083
00084         res = GSASL_NEEDS_MORE;
00085         state->step++;
00086       }
00087       break;
00088
00089     case 1:
00090       {
00091         res = gsasl_property_set_raw (sctx, GSASL_OPENID20_REDIRECT_URL,
00092                                       input, input_len);
00093         if (res != GSASL_OK)
00094           return res;
00095
00096         res = gsasl_callback (NULL, sctx,
00097                               GSASL_OPENID20_AUTHENTICATE_IN_BROWSER);
00098         if (res != GSASL_OK)
00099           return res;
00100
00101         *output_len = 1;
00102         *output = strdup ("=");
00103         if (!*output)
00104           return GSASL_MALLOC_ERROR;
00105
00106         res = GSASL_OK;
00107         state->step++;
00108       }
00109       break;
00110
00111       /* This step is optional.  The server could have approved
00112          authentication already.  Alternatively, it wanted to send
00113          some SREGs or error data and we end up here. */
00114     case 2:
00115       {
00116         res = gsasl_property_set_raw (sctx, GSASL_OPENID20_OUTCOME_DATA,
00117                                       input, input_len);
00118         if (res != GSASL_OK)
00119           return res;
00120
00121         /* In the case of failures, the response MUST follow this
00122            syntax:
00123
00124            outcome_data = "openid.error" "=" sreg_val *( "," sregp_avp )
00125
00126            [RFC4422] Section 3.6 explicitly prohibits additional information in
00127            an unsuccessful authentication outcome.  Therefore, the openid.error
00128            and openid.error_code are to be sent as an additional challenge in
00129            the event of an unsuccessful outcome.  In this case, as the protocol
00130            is lock step,  the client will follow with an additional exchange
00131            containing "=", after which the server will respond with an
00132            application-level outcome.
00133          */
00134
00135 #define ERR_PREFIX "openid.error="
00136         if (input_len > strlen (ERR_PREFIX)
00137             && strncmp (ERR_PREFIX, input, strlen (ERR_PREFIX)) == 0)
00138           {
00139             *output_len = 1;
00140             *output = strdup ("=");
00141             if (!*output)
00142               return GSASL_MALLOC_ERROR;
00143
00144             res = GSASL_NEEDS_MORE;
00145           }
00146         else
00147           {
00148             *output_len = 0;
00149             *output = NULL;
00150
00151             res = GSASL_OK;
00152           }
00153
00154         state->step++;
00155       }
00156       break;
00157
00158     default:
00159       break;
00160     }
00161
00162   return res;
00163 }
00164
00165 void
00166 _gsasl_openid20_client_finish (Gsasl_session *sctx _GL_UNUSED,
00167                                void *mech_data)
00168 {
00169   struct openid20_client_state *state = mech_data;
```

```
00170
00171   if (!state)
00172     return;
00173
00174   free (state);
00175 }
```

## 5.73 plain/client.c File Reference

```
#include <config.h>
#include "plain.h"
#include <string.h>
#include <stdlib.h>
```

### Functions

- int _gsasl_plain_client_step (Gsasl_session *sctx, void *mech_data _GL_UNUSED, const char *input _↩ GL_UNUSED, size_t input_len _GL_UNUSED, char **output, size_t *output_len)

### 5.73.1 Function Documentation

#### 5.73.1.1 _gsasl_plain_client_step()

```
int _gsasl_plain_client_step (
              Gsasl_session * sctx,
              void *mech_data _GL_UNUSED,
              const char *input _GL_UNUSED,
              size_t input_len _GL_UNUSED,
              char ** output,
              size_t * output_len )
```

Definition at line 34 of file plain/client.c.

## 5.74 plain/client.c

Go to the documentation of this file.

```
00001 /* client.c --- SASL mechanism PLAIN as defined in RFC 2595, client side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
```

```
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "plain.h"
00026
00027 /* Get memcpy, strdup, strlen. */
00028 #include <string.h>
00029
00030 /* Get malloc, free. */
00031 #include <stdlib.h>
00032
00033 int
00034 _gsasl_plain_client_step (Gsasl_session *sctx,
00035                           void *mech_data _GL_UNUSED,
00036                           const char *input _GL_UNUSED,
00037                           size_t input_len _GL_UNUSED,
00038                           char **output, size_t *output_len)
00039 {
00040   const char *authzid = gsasl_property_get (sctx, GSASL_AUTHZID);
00041   const char *authid = gsasl_property_get (sctx, GSASL_AUTHID);
00042   const char *password = gsasl_property_get (sctx, GSASL_PASSWORD);
00043   size_t authzidlen = 0, authidlen = 0, passwordlen = 0;
00044   char *out;
00045
00046   if (authzid)
00047     authzidlen = strlen (authzid);
00048
00049   if (authid)
00050     authidlen = strlen (authid);
00051   else
00052     return GSASL_NO_AUTHID;
00053
00054   if (password)
00055     passwordlen = strlen (password);
00056   else
00057     return GSASL_NO_PASSWORD;
00058
00059   *output_len = authzidlen + 1 + authidlen + 1 + passwordlen;
00060   *output = out = malloc (*output_len);
00061   if (!out)
00062     return GSASL_MALLOC_ERROR;
00063
00064   if (authzid)
00065     {
00066       memcpy (out, authzid, authzidlen);
00067       out += authzidlen;
00068     }
00069
00070   *out++ = '\0';
00071
00072   memcpy (out, authid, authidlen);
00073   out += authidlen;
00074
00075   *out++ = '\0';
00076
00077   memcpy (out, password, passwordlen);
00078
00079   return GSASL_OK;
00080 }
```

## 5.75 saml20/client.c File Reference

```
#include <config.h>
#include "saml20.h"
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include "mechtools.h"
```

### Data Structures

- struct saml20_client_state

**Functions**

- int _gsasl_saml20_client_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
- int _gsasl_saml20_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_saml20_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)

### 5.75.1 Function Documentation

#### 5.75.1.1 _gsasl_saml20_client_finish()

```
void _gsasl_saml20_client_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 119 of file saml20/client.c.

#### 5.75.1.2 _gsasl_saml20_client_start()

```
int _gsasl_saml20_client_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 45 of file saml20/client.c.

#### 5.75.1.3 _gsasl_saml20_client_step()

```
int _gsasl_saml20_client_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 59 of file saml20/client.c.

## 5.76 saml20/client.c

Go to the documentation of this file.

```
00001 /* client.c --- SAML20 mechanism, client side.
00002  * Copyright (C) 2010-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
```

```
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "saml20.h"
00026
00027 /* Get strdup, strlen. */
00028 #include <string.h>
00029
00030 /* Get free. */
00031 #include <stdlib.h>
00032
00033 /* Get bool. */
00034 #include <stdbool.h>
00035
00036 /* Get _gsasl_gs2_generate_header. */
00037 #include "mechtools.h"
00038
00039 struct saml20_client_state
00040 {
00041   int step;
00042 };
00043
00044 int
00045 _gsasl_saml20_client_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
00046 {
00047   struct saml20_client_state *state;
00048
00049   state = (struct saml20_client_state *) calloc (1, sizeof (*state));
00050   if (state == NULL)
00051     return GSASL_MALLOC_ERROR;
00052
00053   *mech_data = state;
00054
00055   return GSASL_OK;
00056 }
00057
00058 int
00059 _gsasl_saml20_client_step (Gsasl_session *sctx,
00060                            void *mech_data,
00061                            const char *input, size_t input_len,
00062                            char **output, size_t *output_len)
00063 {
00064   struct saml20_client_state *state = mech_data;
00065   int res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00066
00067   switch (state->step)
00068     {
00069     case 0:
00070       {
00071         const char *authzid = gsasl_property_get (sctx, GSASL_AUTHZID);
00072         const char *idp =
00073           gsasl_property_get (sctx, GSASL_SAML20_IDP_IDENTIFIER);
00074
00075         if (!idp || !*idp)
00076           return GSASL_NO_SAML20_IDP_IDENTIFIER;
00077
00078         res = _gsasl_gs2_generate_header (false, 'n', NULL, authzid,
00079                                           strlen (idp), idp,
00080                                           output, output_len);
00081         if (res != GSASL_OK)
00082           return res;
00083
00084         res = GSASL_NEEDS_MORE;
00085         state->step++;
00086       }
00087       break;
00088
00089     case 1:
00090       {
00091         res = gsasl_property_set_raw (sctx, GSASL_SAML20_REDIRECT_URL,
00092                                       input, input_len);
00093         if (res != GSASL_OK)
00094           return res;
00095
00096         res = gsasl_callback (NULL, sctx,
00097                               GSASL_SAML20_AUTHENTICATE_IN_BROWSER);
00098         if (res != GSASL_OK)
00099           return res;
00100
00101         *output_len = 1;
00102         *output = strdup ("=");
```

```
00103         if (!*output)
00104           return GSASL_MALLOC_ERROR;
00105
00106         res = GSASL_OK;
00107         state->step++;
00108       }
00109       break;
00110
00111     default:
00112       break;
00113     }
00114
00115   return res;
00116 }
00117
00118 void
00119 _gsasl_saml20_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)
00120 {
00121   struct saml20_client_state *state = mech_data;
00122
00123   if (!state)
00124     return;
00125
00126   free (state);
00127 }
```

## 5.77 scram/client.c File Reference

```
#include <config.h>
#include "scram.h"
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "tokens.h"
#include "parser.h"
#include "printer.h"
#include "gc.h"
#include "memxor.h"
#include "tools.h"
#include "mechtools.h"
```

### Data Structures

- struct scram_client_state

### Macros

- #define CNONCE_ENTROPY_BYTES 18

### Functions

- int _gsasl_scram_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_scram_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)

### 5.77.1 Macro Definition Documentation

#### 5.77.1.1 CNONCE_ENTROPY_BYTES

```
#define CNONCE_ENTROPY_BYTES 18
```

Definition at line 44 of file scram/client.c.

### 5.77.2 Function Documentation

#### 5.77.2.1 _gsasl_scram_client_finish()

```
void _gsasl_scram_client_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 422 of file scram/client.c.

#### 5.77.2.2 _gsasl_scram_client_step()

```
int _gsasl_scram_client_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 124 of file scram/client.c.

## 5.78 scram/client.c

Go to the documentation of this file.

```
00001 /* client.c --- SASL SCRAM client side functions.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "scram.h"
00026
```

```
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get memcpy, strlen, strchr. */
00031 #include <string.h>
00032
00033 /* Get bool. */
00034 #include <stdbool.h>
00035
00036 #include "tokens.h"
00037 #include "parser.h"
00038 #include "printer.h"
00039 #include "gc.h"
00040 #include "memxor.h"
00041 #include "tools.h"
00042 #include "mechtools.h"
00043
00044 #define CNONCE_ENTROPY_BYTES 18
00045
00046 struct scram_client_state
00047 {
00048   bool plus;
00049   Gsasl_hash hash;
00050   int step;
00051   char *cfmb;                       /* client first message bare */
00052   char *serversignature;
00053   char *authmessage;
00054   struct scram_client_first cf;
00055   struct scram_server_first sf;
00056   struct scram_client_final cl;
00057   struct scram_server_final sl;
00058 };
00059
00060 static int
00061 scram_start (Gsasl_session *sctx _GL_UNUSED,
00062              void **mech_data, bool plus, Gsasl_hash hash)
00063 {
00064   struct scram_client_state *state;
00065   char buf[CNONCE_ENTROPY_BYTES];
00066   int rc;
00067
00068   state = (struct scram_client_state *) calloc (1, sizeof (*state));
00069   if (state == NULL)
00070     return GSASL_MALLOC_ERROR;
00071
00072   state->plus = plus;
00073   state->hash = hash;
00074
00075   rc = gsasl_nonce (buf, CNONCE_ENTROPY_BYTES);
00076   if (rc != GSASL_OK)
00077     {
00078       free (state);
00079       return rc;
00080     }
00081
00082   rc = gsasl_base64_to (buf, CNONCE_ENTROPY_BYTES,
00083                         &state->cf.client_nonce, NULL);
00084   if (rc != GSASL_OK)
00085     {
00086       free (state);
00087       return rc;
00088     }
00089
00090   *mech_data = state;
00091
00092   return GSASL_OK;
00093 }
00094
00095 #ifdef USE_SCRAM_SHA1
00096 int
00097 _gsasl_scram_sha1_client_start (Gsasl_session *sctx, void **mech_data)
00098 {
00099   return scram_start (sctx, mech_data, false, GSASL_HASH_SHA1);
00100 }
00101
00102 int
00103 _gsasl_scram_sha1_plus_client_start (Gsasl_session *sctx, void **mech_data)
00104 {
00105   return scram_start (sctx, mech_data, true, GSASL_HASH_SHA1);
00106 }
00107 #endif
00108
00109 #ifdef USE_SCRAM_SHA256
00110 int
00111 _gsasl_scram_sha256_client_start (Gsasl_session *sctx, void **mech_data)
00112 {
00113   return scram_start (sctx, mech_data, false, GSASL_HASH_SHA256);
```

```
00114 }
00115
00116 int
00117 _gsasl_scram_sha256_plus_client_start (Gsasl_session *sctx, void **mech_data)
00118 {
00119   return scram_start (sctx, mech_data, true, GSASL_HASH_SHA256);
00120 }
00121 #endif
00122
00123 int
00124 _gsasl_scram_client_step (Gsasl_session *sctx,
00125                           void *mech_data,
00126                           const char *input, size_t input_len,
00127                           char **output, size_t *output_len)
00128 {
00129   struct scram_client_state *state = mech_data;
00130   int res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00131   int rc;
00132
00133   *output = NULL;
00134   *output_len = 0;
00135
00136   switch (state->step)
00137     {
00138     case 0:
00139       {
00140         const char *p, *b64cb;
00141
00142         p = gsasl_property_get (sctx, GSASL_AUTHID);
00143         if (!p)
00144           return GSASL_NO_AUTHID;
00145
00146         free (state->cf.username);
00147         rc = gsasl_saslprep (p, GSASL_ALLOW_UNASSIGNED,
00148                              &state->cf.username, NULL);
00149         if (rc != GSASL_OK)
00150           return rc;
00151
00152         p = gsasl_property_get (sctx, GSASL_AUTHZID);
00153         if (p)
00154           state->cf.authzid = strdup (p);
00155
00156         b64cb = gsasl_property_get (sctx, GSASL_CB_TLS_EXPORTER);
00157         if (b64cb)
00158           state->cf.cbname = strdup ("tls-exporter");
00159         else
00160           {
00161             b64cb = gsasl_property_get (sctx, GSASL_CB_TLS_UNIQUE);
00162             if (b64cb)
00163               state->cf.cbname = strdup ("tls-unique");
00164           }
00165
00166         if (state->plus)
00167           {
00168             if (!b64cb)
00169               return GSASL_NO_CB_TLS_EXPORTER;
00170
00171             state->cf.cbflag = 'p';
00172           }
00173         else
00174           {
00175             if (b64cb)
00176               state->cf.cbflag = 'y';
00177             else
00178               state->cf.cbflag = 'n';
00179           }
00180
00181         rc = scram_print_client_first (&state->cf, output);
00182         if (rc == -2)
00183           return GSASL_MALLOC_ERROR;
00184         else if (rc != 0)
00185           return GSASL_AUTHENTICATION_ERROR;
00186
00187         *output_len = strlen (*output);
00188
00189         /* Point p to client-first-message-bare. */
00190         p = strchr (*output, ',');
00191         if (!p)
00192           return GSASL_AUTHENTICATION_ERROR;
00193         p++;
00194         p = strchr (p, ',');
00195         if (!p)
00196           return GSASL_AUTHENTICATION_ERROR;
00197         p++;
00198
00199         /* Save "client-first-message-bare" for the next step. */
00200         state->cfmb = strdup (p);
```

```
00201         if (!state->cfmb)
00202           return GSASL_MALLOC_ERROR;
00203
00204         /* Prepare B64("cbind-input") for the next step. */
00205         if (state->plus && b64cb)
00206           {
00207             size_t len;
00208             char *cbind_input;
00209             char *cb;
00210             size_t cblen;
00211
00212             rc = gsasl_base64_from (b64cb, strlen (b64cb), &cb, &cblen);
00213             if (rc != GSASL_OK)
00214               return rc;
00215
00216             len = (p - *output) + cblen;
00217             cbind_input = malloc (len);
00218             if (cbind_input == NULL)
00219               return GSASL_MALLOC_ERROR;
00220
00221             memcpy (cbind_input, *output, p - *output);
00222             memcpy (cbind_input + (p - *output), cb, cblen);
00223             free (cb);
00224             rc = gsasl_base64_to (cbind_input, len, &state->cl.cbind, NULL);
00225             free (cbind_input);
00226           }
00227         else
00228           rc = gsasl_base64_to (*output, p - *output, &state->cl.cbind, NULL);
00229         if (rc != GSASL_OK)
00230           return rc;
00231
00232         /* We are done. */
00233         state->step++;
00234         return GSASL_NEEDS_MORE;
00235         break;
00236       }
00237
00238     case 1:
00239       {
00240         if (scram_parse_server_first (input, input_len, &state->sf) < 0)
00241           return GSASL_MECHANISM_PARSE_ERROR;
00242
00243         if (strlen (state->sf.nonce) < strlen (state->cf.client_nonce) ||
00244             memcmp (state->cf.client_nonce, state->sf.nonce,
00245                     strlen (state->cf.client_nonce)) != 0)
00246           return GSASL_AUTHENTICATION_ERROR;
00247
00248         free (state->cl.nonce);
00249         state->cl.nonce = strdup (state->sf.nonce);
00250         if (!state->cl.nonce)
00251           return GSASL_MALLOC_ERROR;
00252
00253         /* Save salt/iter as properties, so that client callback can
00254            access them. */
00255         {
00256           char *str = NULL;
00257           int n;
00258           n = asprintf (&str, "%zu", state->sf.iter);
00259           if (n < 0 || str == NULL)
00260             return GSASL_MALLOC_ERROR;
00261           rc = gsasl_property_set (sctx, GSASL_SCRAM_ITER, str);
00262           free (str);
00263           if (rc != GSASL_OK)
00264             return rc;
00265         }
00266
00267         rc = gsasl_property_set (sctx, GSASL_SCRAM_SALT, state->sf.salt);
00268         if (rc != GSASL_OK)
00269           return rc;
00270
00271         /* Generate ClientProof. */
00272         {
00273           char saltedpassword[GSASL_HASH_MAX_SIZE];
00274           char clientkey[GSASL_HASH_MAX_SIZE];
00275           char serverkey[GSASL_HASH_MAX_SIZE];
00276           char storedkey[GSASL_HASH_MAX_SIZE];
00277           const char *p;
00278
00279           /* Get SaltedPassword. */
00280
00281           if ((p = gsasl_property_get (sctx, GSASL_SCRAM_SALTED_PASSWORD))
00282               && (strlen (p) == 2 * gsasl_hash_length (state->hash))
00283               && _gsasl_hex_p (p))
00284             {
00285               _gsasl_hex_decode (p, saltedpassword);
00286
00287               rc = gsasl_scram_secrets_from_salted_password (state->hash,
```

```
00288                                                   saltedpassword,
00289                                                   clientkey,
00290                                                   serverkey,
00291                                                   storedkey);
00292             if (rc != 0)
00293               return rc;
00294           }
00295         else if ((p = gsasl_property_get (sctx, GSASL_PASSWORD)) != NULL)
00296           {
00297             char *salt;
00298             size_t saltlen;
00299
00300             rc = gsasl_base64_from (state->sf.salt, strlen (state->sf.salt),
00301                                     &salt, &saltlen);
00302             if (rc != 0)
00303               return rc;
00304
00305             rc = gsasl_scram_secrets_from_password (state->hash,
00306                                                     p,
00307                                                     state->sf.iter,
00308                                                     salt, saltlen,
00309                                                     saltedpassword,
00310                                                     clientkey,
00311                                                     serverkey, storedkey);
00312             if (rc != 0)
00313               return rc;
00314
00315             rc = set_saltedpassword (sctx, state->hash, saltedpassword);
00316             if (rc != GSASL_OK)
00317               return rc;
00318
00319             gsasl_free (salt);
00320           }
00321         else
00322           return GSASL_NO_PASSWORD;
00323
00324         /* Get client-final-message-without-proof. */
00325         {
00326           char *cfmwp;
00327           int n;
00328
00329           state->cl.proof = strdup ("p");
00330           rc = scram_print_client_final (&state->cl, &cfmwp);
00331           if (rc != 0)
00332             return GSASL_MALLOC_ERROR;
00333           free (state->cl.proof);
00334
00335           /* Compute AuthMessage */
00336           n = asprintf (&state->authmessage, "%s,%.*s,%.*s",
00337                         state->cfmb,
00338                         (int) input_len, input,
00339                         (int) (strlen (cfmwp) - 4), cfmwp);
00340           free (cfmwp);
00341           if (n <= 0 || !state->authmessage)
00342             return GSASL_MALLOC_ERROR;
00343         }
00344
00345         {
00346           char clientsignature[GSASL_HASH_MAX_SIZE];
00347           char clientproof[GSASL_HASH_MAX_SIZE];
00348
00349           /* ClientSignature := HMAC(StoredKey, AuthMessage) */
00350           rc = _gsasl_hmac (state->hash,
00351                             storedkey,
00352                             gsasl_hash_length (state->hash),
00353                             state->authmessage,
00354                             strlen (state->authmessage), clientsignature);
00355           if (rc != 0)
00356             return rc;
00357
00358           /* ClientProof := ClientKey XOR ClientSignature */
00359           memcpy (clientproof, clientkey, gsasl_hash_length (state->hash));
00360           memxor (clientproof, clientsignature,
00361                   gsasl_hash_length (state->hash));
00362
00363           rc =
00364             gsasl_base64_to (clientproof, gsasl_hash_length (state->hash),
00365                              &state->cl.proof, NULL);
00366           if (rc != 0)
00367             return rc;
00368         }
00369
00370         /* Generate ServerSignature, for comparison in next step. */
00371         {
00372           char serversignature[GSASL_HASH_MAX_SIZE];
00373
00374           /* ServerSignature := HMAC(ServerKey, AuthMessage) */
```

```
00375             rc = _gsasl_hmac (state->hash,
00376                               serverkey, gsasl_hash_length (state->hash),
00377                               state->authmessage,
00378                               strlen (state->authmessage), serversignature);
00379             if (rc != 0)
00380               return rc;
00381
00382             rc = gsasl_base64_to (serversignature,
00383                                   gsasl_hash_length (state->hash),
00384                                   &state->serversignature, NULL);
00385             if (rc != 0)
00386               return rc;
00387           }
00388         }
00389
00390         rc = scram_print_client_final (&state->cl, output);
00391         if (rc != 0)
00392           return GSASL_MALLOC_ERROR;
00393
00394         *output_len = strlen (*output);
00395
00396         state->step++;
00397         return GSASL_NEEDS_MORE;
00398         break;
00399       }
00400
00401     case 2:
00402       {
00403         if (scram_parse_server_final (input, input_len, &state->sl) < 0)
00404           return GSASL_MECHANISM_PARSE_ERROR;
00405
00406         if (strcmp (state->sl.verifier, state->serversignature) != 0)
00407           return GSASL_AUTHENTICATION_ERROR;
00408
00409         state->step++;
00410         return GSASL_OK;
00411         break;
00412       }
00413
00414     default:
00415       break;
00416     }
00417
00418   return res;
00419 }
00420
00421 void
00422 _gsasl_scram_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)
00423 {
00424   struct scram_client_state *state = mech_data;
00425
00426   if (!state)
00427     return;
00428
00429   free (state->cfmb);
00430   free (state->serversignature);
00431   free (state->authmessage);
00432   scram_free_client_first (&state->cf);
00433   scram_free_server_first (&state->sf);
00434   scram_free_client_final (&state->cl);
00435   scram_free_server_final (&state->sl);
00436
00437   free (state);
00438 }
```

## 5.79 securid/client.c File Reference

```
#include <config.h>
#include "securid.h"
#include <stdlib.h>
#include <string.h>
```

### Macros

- #define PASSCODE "passcode"
- #define PIN "pin"

### Functions

- int _gsasl_securid_client_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
- int _gsasl_securid_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_securid_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)

## 5.79.1 Macro Definition Documentation

### 5.79.1.1 PASSCODE

```
#define PASSCODE "passcode"
```

Definition at line 33 of file securid/client.c.

### 5.79.1.2 PIN

```
#define PIN "pin"
```

Definition at line 34 of file securid/client.c.

## 5.79.2 Function Documentation

### 5.79.2.1 _gsasl_securid_client_finish()

```
void _gsasl_securid_client_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 164 of file securid/client.c.

### 5.79.2.2 _gsasl_securid_client_start()

```
int _gsasl_securid_client_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 37 of file securid/client.c.

### 5.79.2.3 _gsasl_securid_client_step()

```
int _gsasl_securid_client_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 53 of file securid/client.c.

## 5.80 securid/client.c

Go to the documentation of this file.

```
00001 /* client.c --- SASL mechanism SECURID from RFC 2808, client side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "securid.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get strdup, strlen. */
00031 #include <string.h>
00032
00033 #define PASSCODE "passcode"
00034 #define PIN "pin"
00035
00036 int
00037 _gsasl_securid_client_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
00038 {
00039   int *step;
00040
00041   step = (int *) malloc (sizeof (*step));
00042   if (step == NULL)
00043     return GSASL_MALLOC_ERROR;
00044
00045   *step = 0;
00046
00047   *mech_data = step;
00048
00049   return GSASL_OK;
00050 }
00051
00052 int
00053 _gsasl_securid_client_step (Gsasl_session *sctx,
00054                             void *mech_data,
00055                             const char *input,
00056                             size_t input_len,
00057                             char **output, size_t *output_len)
00058 {
00059   int *step = mech_data;
00060   const char *authzid = NULL, *authid = NULL, *passcode = NULL, *pin = NULL;
00061   size_t authzidlen, authidlen, passcodelen, pinlen = 0;
00062   int do_pin = 0;
00063   int res;
00064
00065   switch (*step)
00066     {
00067     case 1:
00068       if (input_len == strlen (PASSCODE) &&
00069           memcmp (input, PASSCODE, strlen (PASSCODE)) == 0)
00070         {
00071           *step = 0;
00072         }
00073       else if (input_len >= strlen (PIN) &&
00074                memcmp (input, PIN, strlen (PIN)) == 0)
00075         {
00076           do_pin = 1;
00077           *step = 0;
00078         }
00079       else
00080         {
00081           *output_len = 0;
00082           res = GSASL_OK;
```

```
00083           break;
00084         }
00085       /* fall through */
00086
00087     case 0:
00088       authzid = gsasl_property_get (sctx, GSASL_AUTHZID);
00089       if (authzid)
00090         authzidlen = strlen (authzid);
00091       else
00092         authzidlen = 0;
00093
00094       authid = gsasl_property_get (sctx, GSASL_AUTHID);
00095       if (!authid)
00096         return GSASL_NO_AUTHID;
00097       authidlen = strlen (authid);
00098
00099       passcode = gsasl_property_get (sctx, GSASL_PASSCODE);
00100       if (!passcode)
00101         return GSASL_NO_PASSCODE;
00102       passcodelen = strlen (passcode);
00103
00104       if (do_pin)
00105         {
00106           if (input_len > strlen (PIN))
00107             {
00108               res = gsasl_property_set_raw (sctx, GSASL_SUGGESTED_PIN,
00109                                             &input[strlen (PIN)],
00110                                             input_len - strlen (PIN));
00111               if (res != GSASL_OK)
00112                 return res;
00113             }
00114
00115           pin = gsasl_property_get (sctx, GSASL_PIN);
00116           if (!pin)
00117             return GSASL_NO_PIN;
00118           pinlen = strlen (pin);
00119         }
00120
00121       *output_len = authzidlen + 1 + authidlen + 1 + passcodelen + 1;
00122       if (do_pin)
00123         *output_len += pinlen + 1;
00124       *output = malloc (*output_len);
00125       if (*output == NULL)
00126         return GSASL_MALLOC_ERROR;
00127
00128       if (authzid)
00129         memcpy (*output, authzid, authzidlen);
00130       (*output)[authzidlen] = '\0';
00131       memcpy (*output + authzidlen + 1, authid, authidlen);
00132       (*output)[authzidlen + 1 + authidlen] = '\0';
00133       memcpy (*output + authzidlen + 1 + authidlen + 1, passcode,
00134               passcodelen);
00135       (*output)[authzidlen + 1 + authidlen + 1 + passcodelen] = '\0';
00136       if (do_pin)
00137         {
00138           memcpy (*output + authzidlen + 1 + authidlen + 1 + passcodelen + 1,
00139                   pin, pinlen);
00140           (*output)[authzidlen + 1 + authidlen + 1 + passcodelen + 1 +
00141                     pinlen] = '\0';
00142         }
00143
00144       (*step)++;
00145       res = GSASL_OK;
00146       break;
00147
00148     case 2:
00149       *output_len = 0;
00150       *output = NULL;
00151       (*step)++;
00152       res = GSASL_OK;
00153       break;
00154
00155     default:
00156       res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00157       break;
00158     }
00159
00160   return res;
00161 }
00162
00163 void
00164 _gsasl_securid_client_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)
00165 {
00166   int *step = mech_data;
00167
00168   free (step);
00169 }
```

## 5.81 anonymous/mechinfo.c File Reference

```
#include <config.h>
#include "anonymous.h"
```

### Variables

- Gsasl_mechanism _gsasl_anonymous_mechanism

### 5.81.1 Variable Documentation

#### 5.81.1.1 _gsasl_anonymous_mechanism

```
Gsasl_mechanism _gsasl_anonymous_mechanism
```

**Initial value:**

```
= {
  GSASL_ANONYMOUS_NAME,
  {
   NULL,
   NULL,
   NULL,



   NULL,

   NULL,
   NULL,
   NULL}
  ,
  {
   NULL,
   NULL,
   NULL,



   NULL,

   NULL,
   NULL,
   NULL}
}
```

Definition at line 27 of file anonymous/mechinfo.c.

## 5.82 anonymous/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of ANONYMOUS mechanism.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
```

```
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "anonymous.h"
00026
00027 Gsasl_mechanism _gsasl_anonymous_mechanism = {
00028   GSASL_ANONYMOUS_NAME,
00029   {
00030    NULL,
00031    NULL,
00032    NULL,
00033 #ifdef USE_CLIENT
00034    _gsasl_anonymous_client_step,
00035 #else
00036    NULL,
00037 #endif
00038    NULL,
00039    NULL,
00040    NULL}
00041   ,
00042   {
00043    NULL,
00044    NULL,
00045    NULL,
00046 #ifdef USE_SERVER
00047    _gsasl_anonymous_server_step,
00048 #else
00049    NULL,
00050 #endif
00051    NULL,
00052    NULL,
00053    NULL}
00054 };
```

## 5.83 cram-md5/mechinfo.c File Reference

```
#include <config.h>
#include "cram-md5.h"
```

### Variables

- Gsasl_mechanism _gsasl_cram_md5_mechanism

### 5.83.1 Variable Documentation

#### 5.83.1.1 _gsasl_cram_md5_mechanism

```
Gsasl_mechanism _gsasl_cram_md5_mechanism
```

Definition at line 27 of file cram-md5/mechinfo.c.

## 5.84 cram-md5/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of CRAM-MD5 mechanism.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "cram-md5.h"
00026
00027 Gsasl_mechanism _gsasl_cram_md5_mechanism = {
00028   GSASL_CRAM_MD5_NAME,
00029   {
00030    NULL,
00031    NULL,
00032    NULL,
00033 #ifdef USE_CLIENT
00034    _gsasl_cram_md5_client_step,
00035 #else
00036    NULL,
00037 #endif
00038    NULL,
00039    NULL,
00040    NULL}
00041   ,
00042   {
00043    NULL,
00044    NULL,
00045 #ifdef USE_SERVER
00046    _gsasl_cram_md5_server_start,
00047 #else
00048    NULL,
00049 #endif
00050 #ifdef USE_SERVER
00051    _gsasl_cram_md5_server_step,
00052 #else
00053    NULL,
00054 #endif
00055 #ifdef USE_SERVER
00056    _gsasl_cram_md5_server_finish,
00057 #else
00058    NULL,
00059 #endif
00060    NULL,
00061    NULL}
00062 };
```

## 5.85 digest-md5/mechinfo.c File Reference

```
#include <config.h>
#include "digest-md5.h"
```

### Variables

- Gsasl_mechanism _gsasl_digest_md5_mechanism

### 5.85.1 Variable Documentation

#### 5.85.1.1 _gsasl_digest_md5_mechanism

```
Gsasl_mechanism _gsasl_digest_md5_mechanism
```

Definition at line 27 of file digest-md5/mechinfo.c.

## 5.86 digest-md5/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of DIGEST-MD5 mechanism.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "digest-md5.h"
00026
00027 Gsasl_mechanism _gsasl_digest_md5_mechanism = {
00028   GSASL_DIGEST_MD5_NAME,
00029   {
00030    NULL,
00031    NULL,
00032 #ifdef USE_CLIENT
00033    _gsasl_digest_md5_client_start,
00034 #else
00035    NULL,
00036 #endif
00037 #ifdef USE_CLIENT
00038    _gsasl_digest_md5_client_step,
00039 #else
00040    NULL,
00041 #endif
00042 #ifdef USE_CLIENT
00043    _gsasl_digest_md5_client_finish,
00044 #else
00045    NULL,
00046 #endif
00047 #ifdef USE_CLIENT
00048    _gsasl_digest_md5_client_encode,
00049 #else
00050    NULL,
00051 #endif
00052 #ifdef USE_CLIENT
00053    _gsasl_digest_md5_client_decode
00054 #else
00055    NULL
00056 #endif
00057    }
00058   ,
00059   {
00060    NULL,
00061    NULL,
00062 #ifdef USE_SERVER
00063    _gsasl_digest_md5_server_start,
00064 #else
00065    NULL,
00066 #endif
```

```
00067 #ifdef USE_SERVER
00068    _gsasl_digest_md5_server_step,
00069 #else
00070    NULL,
00071 #endif
00072 #ifdef USE_SERVER
00073    _gsasl_digest_md5_server_finish,
00074 #else
00075    NULL,
00076 #endif
00077 #ifdef USE_SERVER
00078    _gsasl_digest_md5_server_encode,
00079 #else
00080    NULL,
00081 #endif
00082 #ifdef USE_SERVER
00083    _gsasl_digest_md5_server_decode
00084 #else
00085    NULL
00086 #endif
00087    }
00088 };
```

## 5.87 external/mechinfo.c File Reference

```
#include <config.h>
#include "external.h"
```

### Variables

- Gsasl_mechanism _gsasl_external_mechanism

### 5.87.1 Variable Documentation

#### 5.87.1.1 _gsasl_external_mechanism

```
Gsasl_mechanism _gsasl_external_mechanism
```

**Initial value:**

```
= {
  GSASL_EXTERNAL_NAME,
  {
   NULL,
   NULL,
   NULL,



   NULL,

   NULL,
   NULL,
   NULL}
  ,
  {
   NULL,
   NULL,
   NULL,



   NULL,

   NULL,
   NULL,
   NULL}
}
```

Definition at line 27 of file external/mechinfo.c.

## 5.88 external/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of EXTERNAL mechanism.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "external.h"
00026
00027 Gsasl_mechanism _gsasl_external_mechanism = {
00028   GSASL_EXTERNAL_NAME,
00029   {
00030    NULL,
00031    NULL,
00032    NULL,
00033 #ifdef USE_CLIENT
00034    _gsasl_external_client_step,
00035 #else
00036    NULL,
00037 #endif
00038    NULL,
00039    NULL,
00040    NULL}
00041   ,
00042   {
00043    NULL,
00044    NULL,
00045    NULL,
00046 #ifdef USE_SERVER
00047    _gsasl_external_server_step,
00048 #else
00049    NULL,
00050 #endif
00051    NULL,
00052    NULL,
00053    NULL}
00054 };
```

## 5.89 gs2/mechinfo.c File Reference

```
#include <config.h>
#include "gs2.h"
```

### Variables

- Gsasl_mechanism _gsasl_gs2_krb5_mechanism

### 5.89.1 Variable Documentation

#### 5.89.1.1 _gsasl_gs2_krb5_mechanism

```
Gsasl_mechanism _gsasl_gs2_krb5_mechanism
```

Definition at line 27 of file gs2/mechinfo.c.

## 5.90 gs2/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of GS2 mechanism.
00002  * Copyright (C) 2006-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "gs2.h"
00026
00027 Gsasl_mechanism _gsasl_gs2_krb5_mechanism = {
00028   GSASL_GS2_KRB5_NAME,
00029   {
00030    NULL,
00031    NULL,
00032 #ifdef USE_CLIENT
00033    _gsasl_gs2_client_start,
00034 #else
00035    NULL,
00036 #endif
00037 #ifdef USE_CLIENT
00038    _gsasl_gs2_client_step,
00039 #else
00040    NULL,
00041 #endif
00042 #ifdef USE_CLIENT
00043    _gsasl_gs2_client_finish,
00044 #else
00045    NULL,
00046 #endif
00047    NULL,
00048    NULL}
00049   ,
00050   {
00051    NULL,
00052    NULL,
00053 #ifdef USE_SERVER
00054    _gsasl_gs2_server_start,
00055 #else
00056    NULL,
00057 #endif
00058 #ifdef USE_SERVER
00059    _gsasl_gs2_server_step,
00060 #else
00061    NULL,
00062 #endif
00063 #ifdef USE_SERVER
00064    _gsasl_gs2_server_finish,
00065 #else
00066    NULL,
00067 #endif
00068    NULL,
00069    NULL}
00070 };
```

## 5.91 gssapi/mechinfo.c File Reference

```
#include <config.h>
#include "x-gssapi.h"
```

### Variables

- Gsasl_mechanism _gsasl_gssapi_mechanism

### 5.91.1 Variable Documentation

#### 5.91.1.1 _gsasl_gssapi_mechanism

```
Gsasl_mechanism _gsasl_gssapi_mechanism
```

Definition at line 27 of file gssapi/mechinfo.c.

## 5.92 gssapi/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of GSSAPI mechanism.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "x-gssapi.h"
00026
00027 Gsasl_mechanism _gsasl_gssapi_mechanism = {
00028   GSASL_GSSAPI_NAME,
00029   {
00030     NULL,
00031     NULL,
00032 #ifdef USE_CLIENT
00033     _gsasl_gssapi_client_start,
00034 #else
00035     NULL,
00036 #endif
00037 #ifdef USE_CLIENT
00038     _gsasl_gssapi_client_step,
00039 #else
00040     NULL,
00041 #endif
00042 #ifdef USE_CLIENT
00043     _gsasl_gssapi_client_finish,
00044 #else
00045     NULL,
```

```
00046 #endif
00047 #ifdef USE_CLIENT
00048    _gsasl_gssapi_client_encode,
00049 #else
00050    NULL,
00051 #endif
00052 #ifdef USE_CLIENT
00053    _gsasl_gssapi_client_decode
00054 #else
00055    NULL
00056 #endif
00057    }
00058   ,
00059   {
00060    NULL,
00061    NULL,
00062 #ifdef USE_SERVER
00063    _gsasl_gssapi_server_start,
00064 #else
00065    NULL,
00066 #endif
00067 #ifdef USE_SERVER
00068    _gsasl_gssapi_server_step,
00069 #else
00070    NULL,
00071 #endif
00072 #ifdef USE_SERVER
00073    _gsasl_gssapi_server_finish,
00074 #else
00075    NULL,
00076 #endif
00077    NULL,
00078    NULL}
00079 };
```

## 5.93 login/mechinfo.c File Reference

```
#include <config.h>
#include "login.h"
```

### Variables

- Gsasl_mechanism _gsasl_login_mechanism

### 5.93.1 Variable Documentation

#### 5.93.1.1 _gsasl_login_mechanism

```
Gsasl_mechanism _gsasl_login_mechanism
```

Definition at line 27 of file login/mechinfo.c.

## 5.94 login/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of LOGIN mechanism.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
```

```
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "login.h"
00026
00027 Gsasl_mechanism _gsasl_login_mechanism = {
00028   GSASL_LOGIN_NAME,
00029   {
00030    NULL,
00031    NULL,
00032 #ifdef USE_CLIENT
00033    _gsasl_login_client_start,
00034 #else
00035    NULL,
00036 #endif
00037 #ifdef USE_CLIENT
00038    _gsasl_login_client_step,
00039 #else
00040    NULL,
00041 #endif
00042 #ifdef USE_CLIENT
00043    _gsasl_login_client_finish,
00044 #else
00045    NULL,
00046 #endif
00047    NULL,
00048    NULL}
00049   ,
00050   {
00051    NULL,
00052    NULL,
00053 #ifdef USE_SERVER
00054    _gsasl_login_server_start,
00055 #else
00056    NULL,
00057 #endif
00058 #ifdef USE_SERVER
00059    _gsasl_login_server_step,
00060 #else
00061    NULL,
00062 #endif
00063 #ifdef USE_SERVER
00064    _gsasl_login_server_finish,
00065 #else
00066    NULL,
00067 #endif
00068    NULL,
00069    NULL}
00070 };
```

## 5.95 ntlm/mechinfo.c File Reference

```
#include <config.h>
#include "x-ntlm.h"
```

### Variables

- Gsasl_mechanism _gsasl_ntlm_mechanism

### 5.95.1 Variable Documentation

#### 5.95.1.1 _gsasl_ntlm_mechanism

```
Gsasl_mechanism _gsasl_ntlm_mechanism
```

Definition at line 27 of file ntlm/mechinfo.c.

## 5.96 ntlm/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of NTLM mechanism.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "x-ntlm.h"
00026
00027 Gsasl_mechanism _gsasl_ntlm_mechanism = {
00028   GSASL_NTLM_NAME,
00029   {
00030    NULL,
00031    NULL,
00032 #ifdef USE_CLIENT
00033    _gsasl_ntlm_client_start,
00034 #else
00035    NULL,
00036 #endif
00037 #ifdef USE_CLIENT
00038    _gsasl_ntlm_client_step,
00039 #else
00040    NULL,
00041 #endif
00042 #ifdef USE_CLIENT
00043    _gsasl_ntlm_client_finish,
00044 #else
00045    NULL,
00046 #endif
00047    NULL,
00048    NULL}
00049   ,
00050   {
00051    NULL,
00052    NULL,
00053    NULL,
00054    NULL,
00055    NULL,
00056    NULL,
00057    NULL}
00058 };
```

## 5.97 openid20/mechinfo.c File Reference

```
#include <config.h>
#include "openid20.h"
```

### Variables

- Gsasl_mechanism _gsasl_openid20_mechanism

## 5.97.1 Variable Documentation

### 5.97.1.1 _gsasl_openid20_mechanism

```
Gsasl_mechanism _gsasl_openid20_mechanism
```

Definition at line 27 of file openid20/mechinfo.c.

# 5.98 openid20/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of OPENID20 mechanism.
00002  * Copyright (C) 2011-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "openid20.h"
00026
00027 Gsasl_mechanism _gsasl_openid20_mechanism = {
00028   GSASL_OPENID20_NAME,
00029   {
00030     NULL,
00031     NULL,
00032 #ifdef USE_CLIENT
00033     _gsasl_openid20_client_start,
00034     _gsasl_openid20_client_step,
00035     _gsasl_openid20_client_finish,
00036 #else
00037     NULL,
00038     NULL,
00039     NULL,
00040 #endif
00041     NULL,
00042     NULL}
00043   ,
00044   {
00045     NULL,
00046     NULL,
00047 #ifdef USE_SERVER
00048     _gsasl_openid20_server_start,
00049     _gsasl_openid20_server_step,
00050     _gsasl_openid20_server_finish,
00051 #else
00052     NULL,
00053     NULL,
00054     NULL,
00055 #endif
00056     NULL,
00057     NULL}
00058 };
```

## 5.99 plain/mechinfo.c File Reference

```
#include <config.h>
#include "plain.h"
```

### Variables

- Gsasl_mechanism _gsasl_plain_mechanism

### 5.99.1 Variable Documentation

#### 5.99.1.1 _gsasl_plain_mechanism

```
Gsasl_mechanism _gsasl_plain_mechanism
```

**Initial value:**

```
= {
  GSASL_PLAIN_NAME,
  {
   NULL,
   NULL,
   NULL,


   NULL,

   NULL,
   NULL,
   NULL}
  ,
  {
   NULL,
   NULL,
   NULL,


   NULL,

   NULL,
   NULL,
   NULL}
}
```

Definition at line 27 of file plain/mechinfo.c.

## 5.100 plain/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of PLAIN mechanism.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
```

```
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "plain.h"
00026
00027 Gsasl_mechanism _gsasl_plain_mechanism = {
00028   GSASL_PLAIN_NAME,
00029   {
00030    NULL,
00031    NULL,
00032    NULL,
00033 #ifdef USE_CLIENT
00034    _gsasl_plain_client_step,
00035 #else
00036    NULL,
00037 #endif
00038    NULL,
00039    NULL,
00040    NULL}
00041   ,
00042   {
00043    NULL,
00044    NULL,
00045    NULL,
00046 #ifdef USE_SERVER
00047    _gsasl_plain_server_step,
00048 #else
00049    NULL,
00050 #endif
00051    NULL,
00052    NULL,
00053    NULL}
00054 };
```

## 5.101 saml20/mechinfo.c File Reference

```
#include <config.h>
#include "saml20.h"
```

### Variables

- Gsasl_mechanism _gsasl_saml20_mechanism

### 5.101.1 Variable Documentation

#### 5.101.1.1 _gsasl_saml20_mechanism

```
Gsasl_mechanism _gsasl_saml20_mechanism
```

Definition at line 27 of file saml20/mechinfo.c.

## 5.102 saml20/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of SAML20 mechanism.
00002  * Copyright (C) 2010-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "saml20.h"
00026
00027 Gsasl_mechanism _gsasl_saml20_mechanism = {
00028   GSASL_SAML20_NAME,
00029   {
00030    NULL,
00031    NULL,
00032 #ifdef USE_CLIENT
00033    _gsasl_saml20_client_start,
00034    _gsasl_saml20_client_step,
00035    _gsasl_saml20_client_finish,
00036 #else
00037    NULL,
00038    NULL,
00039    NULL,
00040 #endif
00041    NULL,
00042    NULL}
00043   ,
00044   {
00045    NULL,
00046    NULL,
00047 #ifdef USE_SERVER
00048    _gsasl_saml20_server_start,
00049    _gsasl_saml20_server_step,
00050    _gsasl_saml20_server_finish,
00051 #else
00052    NULL,
00053    NULL,
00054    NULL,
00055 #endif
00056    NULL,
00057    NULL}
00058 };
```

## 5.103 scram/mechinfo.c File Reference

```
#include <config.h>
#include "scram.h"
```

## 5.104 scram/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of SCRAM mechanism.
```

```
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "scram.h"
00026
00027 #ifdef USE_SCRAM_SHA1
00028 Gsasl_mechanism _gsasl_scram_sha1_mechanism = {
00029   GSASL_SCRAM_SHA1_NAME,
00030   {
00031    NULL,
00032    NULL,
00033 # ifdef USE_CLIENT
00034    _gsasl_scram_sha1_client_start,
00035    _gsasl_scram_client_step,
00036    _gsasl_scram_client_finish,
00037 # else
00038    NULL,
00039    NULL,
00040    NULL,
00041 # endif
00042    NULL,
00043    NULL}
00044   ,
00045   {
00046    NULL,
00047    NULL,
00048 # ifdef USE_SERVER
00049    _gsasl_scram_sha1_server_start,
00050    _gsasl_scram_server_step,
00051    _gsasl_scram_server_finish,
00052 # else
00053    NULL,
00054    NULL,
00055    NULL,
00056 # endif
00057    NULL,
00058    NULL}
00059 };
00060 #endif
00061
00062
00063 #ifdef USE_SCRAM_SHA1
00064 Gsasl_mechanism _gsasl_scram_sha1_plus_mechanism = {
00065   GSASL_SCRAM_SHA1_PLUS_NAME,
00066   {
00067    NULL,
00068    NULL,
00069 # ifdef USE_CLIENT
00070    _gsasl_scram_sha1_plus_client_start,
00071    _gsasl_scram_client_step,
00072    _gsasl_scram_client_finish,
00073 # else
00074    NULL,
00075    NULL,
00076    NULL,
00077 # endif
00078    NULL,
00079    NULL}
00080   ,
00081   {
00082    NULL,
00083    NULL,
00084 # ifdef USE_SERVER
00085    _gsasl_scram_sha1_plus_server_start,
00086    _gsasl_scram_server_step,
00087    _gsasl_scram_server_finish,
00088 # else
```

```
00089     NULL,
00090     NULL,
00091     NULL,
00092 # endif
00093     NULL,
00094     NULL}
00095 };
00096 #endif
00097
00098 #ifdef USE_SCRAM_SHA256
00099 Gsasl_mechanism _gsasl_scram_sha256_mechanism = {
00100   GSASL_SCRAM_SHA256_NAME,
00101   {
00102    NULL,
00103    NULL,
00104 # ifdef USE_CLIENT
00105    _gsasl_scram_sha256_client_start,
00106    _gsasl_scram_client_step,
00107    _gsasl_scram_client_finish,
00108 # else
00109    NULL,
00110    NULL,
00111    NULL,
00112 # endif
00113    NULL,
00114    NULL}
00115   ,
00116   {
00117    NULL,
00118    NULL,
00119 # ifdef USE_SERVER
00120    _gsasl_scram_sha256_server_start,
00121    _gsasl_scram_server_step,
00122    _gsasl_scram_server_finish,
00123 # else
00124    NULL,
00125    NULL,
00126    NULL,
00127 # endif
00128    NULL,
00129    NULL}
00130 };
00131 #endif
00132
00133
00134 #ifdef USE_SCRAM_SHA256
00135 Gsasl_mechanism _gsasl_scram_sha256_plus_mechanism = {
00136   GSASL_SCRAM_SHA256_PLUS_NAME,
00137   {
00138    NULL,
00139    NULL,
00140 # ifdef USE_CLIENT
00141    _gsasl_scram_sha256_plus_client_start,
00142    _gsasl_scram_client_step,
00143    _gsasl_scram_client_finish,
00144 # else
00145    NULL,
00146    NULL,
00147    NULL,
00148 # endif
00149    NULL,
00150    NULL}
00151   ,
00152   {
00153    NULL,
00154    NULL,
00155 # ifdef USE_SERVER
00156    _gsasl_scram_sha256_plus_server_start,
00157    _gsasl_scram_server_step,
00158    _gsasl_scram_server_finish,
00159 # else
00160    NULL,
00161    NULL,
00162    NULL,
00163 # endif
00164    NULL,
00165    NULL}
00166 };
00167 #endif
```

# 5.105 securid/mechinfo.c File Reference

```
#include <config.h>
#include "securid.h"
```

### Variables

- Gsasl_mechanism _gsasl_securid_mechanism

## 5.105.1 Variable Documentation

### 5.105.1.1 _gsasl_securid_mechanism

```
Gsasl_mechanism _gsasl_securid_mechanism
```

Definition at line 27 of file securid/mechinfo.c.

# 5.106 securid/mechinfo.c

Go to the documentation of this file.

```
00001 /* mechinfo.c --- Definition of SECURID mechanism.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "securid.h"
00026
00027 Gsasl_mechanism _gsasl_securid_mechanism = {
00028   GSASL_SECURID_NAME,
00029   {
00030     NULL,
00031     NULL,
00032 #ifdef USE_CLIENT
00033     _gsasl_securid_client_start,
00034 #else
00035     NULL,
00036 #endif
00037 #ifdef USE_CLIENT
00038     _gsasl_securid_client_step,
00039 #else
00040     NULL,
00041 #endif
00042 #ifdef USE_CLIENT
00043     _gsasl_securid_client_finish,
00044 #else
00045     NULL,
```

```
00046 #endif
00047    NULL,
00048    NULL}
00049   ,
00050   {
00051    NULL,
00052    NULL,
00053    NULL,
00054 #ifdef USE_SERVER
00055    _gsasl_securid_server_step,
00056 #else
00057    NULL,
00058 #endif
00059    NULL,
00060    NULL,
00061    NULL}
00062 };
```

## 5.107 saml20.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_SAML20_NAME "SAML20"

### Functions

- int _gsasl_saml20_client_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_saml20_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_saml20_client_finish (Gsasl_session *sctx, void *mech_data)
- int _gsasl_saml20_server_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_saml20_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_saml20_server_finish (Gsasl_session *sctx, void *mech_data)

### Variables

- Gsasl_mechanism _gsasl_saml20_mechanism

### 5.107.1 Macro Definition Documentation

#### 5.107.1.1 GSASL_SAML20_NAME

```
#define GSASL_SAML20_NAME "SAML20"
```

Definition at line 27 of file saml20.h.

## 5.107.2 Function Documentation

### 5.107.2.1 _gsasl_saml20_client_finish()

```
void _gsasl_saml20_client_finish (
            Gsasl_session * sctx,
            void * mech_data )  [extern]
```

### 5.107.2.2 _gsasl_saml20_client_start()

```
int _gsasl_saml20_client_start (
            Gsasl_session * sctx,
            void ** mech_data )  [extern]
```

### 5.107.2.3 _gsasl_saml20_client_step()

```
int _gsasl_saml20_client_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

Definition at line 59 of file saml20/client.c.

### 5.107.2.4 _gsasl_saml20_server_finish()

```
void _gsasl_saml20_server_finish (
            Gsasl_session * sctx,
            void * mech_data )  [extern]
```

### 5.107.2.5 _gsasl_saml20_server_start()

```
int _gsasl_saml20_server_start (
            Gsasl_session * sctx,
            void ** mech_data )  [extern]
```

### 5.107.2.6 _gsasl_saml20_server_step()

```
int _gsasl_saml20_server_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )  [extern]
```

Definition at line 56 of file saml20/server.c.

### 5.107.3 Variable Documentation

#### 5.107.3.1 _gsasl_saml20_mechanism

```
Gsasl_mechanism _gsasl_saml20_mechanism  [extern]
```

Definition at line 27 of file saml20/mechinfo.c.

## 5.108 saml20.h

Go to the documentation of this file.

```
00001 /* saml20.h --- Prototypes for SAML20.
00002  * Copyright (C) 2010-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef SAML20_H
00023 # define SAML20_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_SAML20_NAME "SAML20"
00028
00029 extern Gsasl_mechanism _gsasl_saml20_mechanism;
00030
00031 extern int _gsasl_saml20_client_start (Gsasl_session * sctx,
00032                                        void **mech_data);
00033
00034 extern int _gsasl_saml20_client_step (Gsasl_session * sctx,
00035                                       void *mech_data,
00036                                       const char *input, size_t input_len,
00037                                       char **output, size_t *output_len);
00038
00039 extern void _gsasl_saml20_client_finish (Gsasl_session * sctx,
00040                                          void *mech_data);
00041
00042 extern int _gsasl_saml20_server_start (Gsasl_session * sctx,
00043                                        void **mech_data);
00044
00045 extern int _gsasl_saml20_server_step (Gsasl_session * sctx,
00046                                       void *mech_data,
00047                                       const char *input, size_t input_len,
00048                                       char **output, size_t *output_len);
00049
00050 extern void _gsasl_saml20_server_finish (Gsasl_session * sctx,
00051                                          void *mech_data);
00052
00053 #endif /* SAML20_H */
```

## 5.109 anonymous/server.c File Reference

```
#include <config.h>
#include "anonymous.h"
```

### Functions

- int _gsasl_anonymous_server_step (Gsasl_session *sctx, void *mech_data _GL_UNUSED, const char *input, size_t input_len, char **output, size_t *output_len)

## 5.109.1 Function Documentation

### 5.109.1.1 _gsasl_anonymous_server_step()

```
int _gsasl_anonymous_server_step (
              Gsasl_session * sctx,
              void *mech_data _GL_UNUSED,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 28 of file anonymous/server.c.

# 5.110 anonymous/server.c

Go to the documentation of this file.

```
00001 /* server.c --- ANONYMOUS mechanism as defined in RFC 2245, server side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "anonymous.h"
00026
00027 int
00028 _gsasl_anonymous_server_step (Gsasl_session *sctx,
00029                               void *mech_data _GL_UNUSED,
00030                               const char *input, size_t input_len,
00031                               char **output, size_t *output_len)
00032 {
00033   int rc;
00034
00035   *output = NULL;
00036   *output_len = 0;
00037
00038   if (!input)
00039     return GSASL_NEEDS_MORE;
00040
00041   /* token       = 1*255TCHAR
00042      The <token> production is restricted to 255 UTF-8 encoded Unicode
00043      characters.   As the encoding of a characters uses a sequence of 1
00044      to 4 octets, a token may be long as 1020 octets. */
00045   if (input_len == 0 || input_len > 1020)
00046     return GSASL_MECHANISM_PARSE_ERROR;
00047
```

```
00048   /* FIXME: Validate that input is UTF-8. */
00049
00050   rc = gsasl_property_set_raw (sctx, GSASL_ANONYMOUS_TOKEN, input, input_len);
00051   if (rc != GSASL_OK)
00052     return rc;
00053
00054   return gsasl_callback (NULL, sctx, GSASL_VALIDATE_ANONYMOUS);
00055 }
```

## 5.111 cram-md5/server.c File Reference

```
#include <config.h>
#include "cram-md5.h"
#include <stdlib.h>
#include <string.h>
#include "challenge.h"
#include "digest.h"
```

### Macros

- #define MD5LEN 16

### Functions

- int _gsasl_cram_md5_server_start (Gsasl_session ∗sctx _GL_UNUSED, void ∗∗mech_data)
- int _gsasl_cram_md5_server_step (Gsasl_session ∗sctx, void ∗mech_data, const char ∗input, size_t input↩ _len, char ∗∗output, size_t ∗output_len)
- void _gsasl_cram_md5_server_finish (Gsasl_session ∗sctx _GL_UNUSED, void ∗mech_data)

### 5.111.1 Macro Definition Documentation

#### 5.111.1.1 MD5LEN

```
#define MD5LEN 16
```

Definition at line 39 of file cram-md5/server.c.

### 5.111.2 Function Documentation

#### 5.111.2.1 _gsasl_cram_md5_server_finish()

```
void _gsasl_cram_md5_server_finish (
            Gsasl_session *sctx _GL_UNUSED,
            void * mech_data )
```

Definition at line 129 of file cram-md5/server.c.

#### 5.111.2.2 _gsasl_cram_md5_server_start()

```
int _gsasl_cram_md5_server_start (
              Gsasl_session *sctx _GL_UNUSED,
              void ** mech_data )
```

Definition at line 42 of file cram-md5/server.c.

#### 5.111.2.3 _gsasl_cram_md5_server_step()

```
int _gsasl_cram_md5_server_step (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 65 of file cram-md5/server.c.

## 5.112 cram-md5/server.c

Go to the documentation of this file.

```
00001 /* server.c --- SASL CRAM-MD5 server side functions.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "cram-md5.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get memcpy, strdup, strlen. */
00031 #include <string.h>
00032
00033 /* Get cram_md5_challenge. */
00034 #include "challenge.h"
00035
00036 /* Get cram_md5_digest. */
00037 #include "digest.h"
00038
00039 #define MD5LEN 16
00040
00041 int
00042 _gsasl_cram_md5_server_start (Gsasl_session *sctx _GL_UNUSED,
00043                               void **mech_data)
00044 {
00045   char *challenge;
00046   int rc;
```

```
00047
00048   challenge = malloc (CRAM_MD5_CHALLENGE_LEN);
00049   if (challenge == NULL)
00050     return GSASL_MALLOC_ERROR;
00051
00052   rc = cram_md5_challenge (challenge);
00053   if (rc)
00054     {
00055       free (challenge);
00056       return GSASL_CRYPTO_ERROR;
00057     }
00058
00059   *mech_data = challenge;
00060
00061   return GSASL_OK;
00062 }
00063
00064 int
00065 _gsasl_cram_md5_server_step (Gsasl_session *sctx,
00066                              void *mech_data,
00067                              const char *input, size_t input_len,
00068                              char **output, size_t *output_len)
00069 {
00070   char *challenge = mech_data;
00071   char hash[CRAM_MD5_DIGEST_LEN];
00072   const char *password;
00073   char *username = NULL;
00074   int res = GSASL_OK;
00075   char *normkey;
00076
00077   if (input_len == 0)
00078     {
00079       *output_len = strlen (challenge);
00080       *output = strdup (challenge);
00081
00082       return GSASL_NEEDS_MORE;
00083     }
00084
00085   if (input_len <= MD5LEN * 2)
00086     return GSASL_MECHANISM_PARSE_ERROR;
00087
00088   if (input[input_len - MD5LEN * 2 - 1] != ' ')
00089     return GSASL_MECHANISM_PARSE_ERROR;
00090
00091   username = calloc (1, input_len - MD5LEN * 2);
00092   if (username == NULL)
00093     return GSASL_MALLOC_ERROR;
00094
00095   memcpy (username, input, input_len - MD5LEN * 2 - 1);
00096
00097   res = gsasl_property_set (sctx, GSASL_AUTHID, username);
00098   free (username);
00099   if (res != GSASL_OK)
00100     return res;
00101
00102   password = gsasl_property_get (sctx, GSASL_PASSWORD);
00103   if (!password)
00104     return GSASL_NO_PASSWORD;
00105
00106   /* FIXME: Use SASLprep here?  Treat string as storage string?
00107      Specification is unclear. */
00108   res = gsasl_saslprep (password, 0, &normkey, NULL);
00109   if (res != GSASL_OK)
00110     return res;
00111
00112   cram_md5_digest (challenge, strlen (challenge),
00113                    normkey, strlen (normkey), hash);
00114
00115   free (normkey);
00116
00117   if (memcmp (&input[input_len - MD5LEN * 2], hash, 2 * MD5LEN) == 0)
00118     res = GSASL_OK;
00119   else
00120     res = GSASL_AUTHENTICATION_ERROR;
00121
00122   *output_len = 0;
00123   *output = NULL;
00124
00125   return res;
00126 }
00127
00128 void
00129 _gsasl_cram_md5_server_finish (Gsasl_session *sctx _GL_UNUSED,
00130                                void *mech_data)
00131 {
00132   char *challenge = mech_data;
00133
```

```
00134   free (challenge);
00135 }
```

## 5.113 digest-md5/server.c File Reference

```
#include <config.h>
#include "digest-md5.h"
#include <stdlib.h>
#include <string.h>
#include "gc.h"
#include "nonascii.h"
#include "tokens.h"
#include "parser.h"
#include "printer.h"
#include "free.h"
#include "session.h"
#include "digesthmac.h"
#include "validate.h"
#include "qop.h"
#include "mechtools.h"
```

### Data Structures

- struct _Gsasl_digest_md5_server_state

### Macros

- #define NONCE_ENTROPY_BYTES 16

### Typedefs

- typedef struct _Gsasl_digest_md5_server_state _Gsasl_digest_md5_server_state

### Functions

- int _gsasl_digest_md5_server_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
- int _gsasl_digest_md5_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input↩ _len, char **output, size_t *output_len)
- void _gsasl_digest_md5_server_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)
- int _gsasl_digest_md5_server_encode (Gsasl_session *sctx _GL_UNUSED, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- int _gsasl_digest_md5_server_decode (Gsasl_session *sctx _GL_UNUSED, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)

### 5.113.1 Macro Definition Documentation

#### 5.113.1.1 NONCE_ENTROPY_BYTES

```
#define NONCE_ENTROPY_BYTES 16
```

Definition at line 48 of file digest-md5/server.c.

## 5.113.2 Typedef Documentation

### 5.113.2.1 _Gsasl_digest_md5_server_state

```
typedef struct _Gsasl_digest_md5_server_state _Gsasl_digest_md5_server_state
```

Definition at line 63 of file digest-md5/server.c.

## 5.113.3 Function Documentation

### 5.113.3.1 _gsasl_digest_md5_server_decode()

```
int _gsasl_digest_md5_server_decode (
              Gsasl_session *sctx _GL_UNUSED,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 388 of file digest-md5/server.c.

### 5.113.3.2 _gsasl_digest_md5_server_encode()

```
int _gsasl_digest_md5_server_encode (
              Gsasl_session *sctx _GL_UNUSED,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 364 of file digest-md5/server.c.

### 5.113.3.3 _gsasl_digest_md5_server_finish()

```
void _gsasl_digest_md5_server_finish (
              Gsasl_session *sctx _GL_UNUSED,
              void * mech_data )
```

Definition at line 348 of file digest-md5/server.c.

### 5.113.3.4 _gsasl_digest_md5_server_start()

```
int _gsasl_digest_md5_server_start (
              Gsasl_session *sctx _GL_UNUSED,
              void ** mech_data )
```

Definition at line 66 of file digest-md5/server.c.

#### 5.113.3.5 _gsasl_digest_md5_server_step()

```
int _gsasl_digest_md5_server_step (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 145 of file digest-md5/server.c.

## 5.114 digest-md5/server.c

Go to the documentation of this file.

```
00001 /* server.c --- DIGEST-MD5 mechanism from RFC 2831, server side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "digest-md5.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get memcpy, strdup, strlen. */
00031 #include <string.h>
00032
00033 #include "gc.h"
00034
00035 /* Get tools. */
00036 #include "nonascii.h"
00037 #include "tokens.h"
00038 #include "parser.h"
00039 #include "printer.h"
00040 #include "free.h"
00041 #include "session.h"
00042 #include "digesthmac.h"
00043 #include "validate.h"
00044 #include "qop.h"
00045
00046 #include "mechtools.h"
00047
00048 #define NONCE_ENTROPY_BYTES 16
00049
00050 struct _Gsasl_digest_md5_server_state
00051 {
00052   int step;
00053   unsigned long readseqnum, sendseqnum;
00054   char secret[DIGEST_MD5_LENGTH];
00055   char kic[DIGEST_MD5_LENGTH];
00056   char kcc[DIGEST_MD5_LENGTH];
00057   char kis[DIGEST_MD5_LENGTH];
00058   char kcs[DIGEST_MD5_LENGTH];
00059   digest_md5_challenge challenge;
00060   digest_md5_response response;
```

```
00061   digest_md5_finish finish;
00062 };
00063 typedef struct _Gsasl_digest_md5_server_state _Gsasl_digest_md5_server_state;
00064
00065 int
00066 _gsasl_digest_md5_server_start (Gsasl_session *sctx _GL_UNUSED,
00067                                 void **mech_data)
00068 {
00069   _Gsasl_digest_md5_server_state *state;
00070   char nonce[NONCE_ENTROPY_BYTES];
00071   char *p;
00072   int rc;
00073
00074   rc = gsasl_nonce (nonce, NONCE_ENTROPY_BYTES);
00075   if (rc != GSASL_OK)
00076     return rc;
00077
00078   rc = gsasl_base64_to (nonce, NONCE_ENTROPY_BYTES, &p, NULL);
00079   if (rc != GSASL_OK)
00080     return rc;
00081
00082   state = calloc (1, sizeof (*state));
00083   if (state == NULL)
00084     {
00085       free (p);
00086       return GSASL_MALLOC_ERROR;
00087     }
00088
00089   state->challenge.qops = DIGEST_MD5_QOP_AUTH;
00090   state->challenge.ciphers = 0;
00091
00092   state->challenge.nonce = p;
00093   state->challenge.utf8 = 1;
00094
00095   *mech_data = state;
00096
00097   return GSASL_OK;
00098 }
00099
00100 static char
00101 _gsasl_digest_md5_hexdigit_to_char (char hexdigit)
00102 {
00103   /* The hex representation always contains lowercase alphabetic
00104      characters.  See RFC 2831, 1.1. */
00105
00106   if (hexdigit >= '0' && hexdigit <= '9')
00107     return hexdigit - '0';
00108   if (hexdigit >= 'a' && hexdigit <= 'z')
00109     return hexdigit - 'a' + 10;
00110
00111   return -1;
00112 }
00113
00114 static char
00115 _gsasl_digest_md5_hex_to_char (char u, char l)
00116 {
00117   return (char) (((unsigned char) _gsasl_digest_md5_hexdigit_to_char (u)) *
00118                  16 + _gsasl_digest_md5_hexdigit_to_char (l));
00119 }
00120
00121 static int
00122 _gsasl_digest_md5_set_hashed_secret (char *secret, const char *hex_secret)
00123 {
00124   /* Convert the hex string containing the secret to a byte array */
00125   const char *p;
00126   char *s;
00127
00128   if (!hex_secret)
00129     return GSASL_AUTHENTICATION_ERROR;
00130
00131   s = secret;
00132   p = hex_secret;
00133   while (*p)
00134     {
00135       *s = _gsasl_digest_md5_hex_to_char (p[0], p[1]);
00136       s++;
00137
00138       p += 2;
00139     }
00140
00141   return GSASL_OK;
00142 }
00143
00144 int
00145 _gsasl_digest_md5_server_step (Gsasl_session *sctx,
00146                                void *mech_data,
00147                                const char *input,
```

```
00148                                size_t input_len,
00149                                char **output, size_t *output_len)
00150 {
00151   _Gsasl_digest_md5_server_state *state = mech_data;
00152   int rc, res;
00153
00154   *output = NULL;
00155   *output_len = 0;
00156
00157   switch (state->step)
00158     {
00159     case 0:
00160       /* Set realm. */
00161       {
00162         const char *c;
00163         c = gsasl_property_get (sctx, GSASL_REALM);
00164         if (c)
00165           {
00166             state->challenge.nrealms = 1;
00167
00168             state->challenge.realms =
00169               malloc (sizeof (*state->challenge.realms));
00170             if (!state->challenge.realms)
00171               return GSASL_MALLOC_ERROR;
00172
00173             state->challenge.realms[0] = strdup (c);
00174             if (!state->challenge.realms[0])
00175               return GSASL_MALLOC_ERROR;
00176           }
00177       }
00178
00179       /* Set QOP */
00180       {
00181         const char *qopstr = gsasl_property_get (sctx, GSASL_QOPS);
00182
00183         if (qopstr)
00184           {
00185             int qops = digest_md5_qopstr2qops (qopstr);
00186
00187             if (qops == -1)
00188               return GSASL_MALLOC_ERROR;
00189
00190             /* We don't support confidentiality right now. */
00191             if (qops & DIGEST_MD5_QOP_AUTH_CONF)
00192               return GSASL_AUTHENTICATION_ERROR;
00193
00194             if (qops)
00195               state->challenge.qops = qops;
00196           }
00197       }
00198
00199       /* FIXME: cipher, maxbuf, more realms. */
00200
00201       /* Create challenge. */
00202       *output = digest_md5_print_challenge (&state->challenge);
00203       if (!*output)
00204         return GSASL_AUTHENTICATION_ERROR;
00205
00206       *output_len = strlen (*output);
00207       state->step++;
00208       res = GSASL_NEEDS_MORE;
00209       break;
00210
00211     case 1:
00212       if (digest_md5_parse_response (input, input_len, &state->response) < 0)
00213         return GSASL_MECHANISM_PARSE_ERROR;
00214
00215       /* Make sure response is consistent with challenge. */
00216       if (digest_md5_validate (&state->challenge, &state->response) < 0)
00217         return GSASL_MECHANISM_PARSE_ERROR;
00218
00219       /* Store properties, from the client response. */
00220       if (state->response.utf8)
00221         {
00222           res = gsasl_property_set (sctx, GSASL_AUTHID,
00223                                     state->response.username);
00224           if (res != GSASL_OK)
00225             return res;
00226
00227           res = gsasl_property_set (sctx, GSASL_REALM, state->response.realm);
00228           if (res != GSASL_OK)
00229             return res;
00230         }
00231       else
00232         {
00233           /* Client provided username/realm in ISO-8859-1 form,
00234              convert it to UTF-8 since the library is all-UTF-8. */
```

```
00235           char *tmp;
00236
00237           tmp = latin1toutf8 (state->response.username);
00238           if (!tmp)
00239             return GSASL_MALLOC_ERROR;
00240           res = gsasl_property_set (sctx, GSASL_AUTHID, tmp);
00241           free (tmp);
00242           if (res != GSASL_OK)
00243             return res;
00244
00245           tmp = latin1toutf8 (state->response.realm);
00246           if (!tmp)
00247             return GSASL_MALLOC_ERROR;
00248           res = gsasl_property_set (sctx, GSASL_REALM, tmp);
00249           free (tmp);
00250           if (res != GSASL_OK)
00251             return res;
00252         }
00253
00254       res = gsasl_property_set (sctx, GSASL_AUTHZID, state->response.authzid);
00255       if (res != GSASL_OK)
00256         return res;
00257
00258       /* FIXME: cipher, maxbuf.  */
00259
00260       /* Compute secret. */
00261       {
00262         const char *passwd;
00263         const char *hashed_passwd;
00264
00265         hashed_passwd =
00266           gsasl_property_get (sctx, GSASL_DIGEST_MD5_HASHED_PASSWORD);
00267         if (hashed_passwd)
00268           {
00269             if (strlen (hashed_passwd) != (DIGEST_MD5_LENGTH * 2))
00270               return GSASL_AUTHENTICATION_ERROR;
00271
00272             rc = _gsasl_digest_md5_set_hashed_secret (state->secret,
00273                                                       hashed_passwd);
00274             if (rc != GSASL_OK)
00275               return rc;
00276           }
00277         else if ((passwd = gsasl_property_get (sctx, GSASL_PASSWORD)) != NULL)
00278           {
00279             char *tmp, *tmp2;
00280
00281             tmp2 = utf8tolatin1ifpossible (passwd);
00282
00283             rc = asprintf (&tmp, "%s:%s:%s", state->response.username,
00284                            state->response.realm ?
00285                            state->response.realm : "", tmp2);
00286             free (tmp2);
00287             if (rc < 0)
00288               return GSASL_MALLOC_ERROR;
00289
00290             rc = gc_md5 (tmp, strlen (tmp), state->secret);
00291             free (tmp);
00292             if (rc != GC_OK)
00293               return GSASL_CRYPTO_ERROR;
00294           }
00295         else
00296           {
00297             return GSASL_NO_PASSWORD;
00298           }
00299       }
00300
00301       /* Check client response. */
00302       {
00303         char check[DIGEST_MD5_RESPONSE_LENGTH + 1];
00304
00305         rc = digest_md5_hmac (check, state->secret,
00306                               state->response.nonce, state->response.nc,
00307                               state->response.cnonce, state->response.qop,
00308                               state->response.authzid,
00309                               state->response.digesturi, 0,
00310                               state->response.cipher,
00311                               state->kic, state->kis, state->kcc, state->kcs);
00312         if (rc)
00313           return GSASL_AUTHENTICATION_ERROR;
00314
00315         if (strcmp (state->response.response, check) != 0)
00316           return GSASL_AUTHENTICATION_ERROR;
00317       }
00318
00319       /* Create finish token. */
00320       rc = digest_md5_hmac (state->finish.rspauth, state->secret,
00321                             state->response.nonce, state->response.nc,
```

```
00322                             state->response.cnonce, state->response.qop,
00323                             state->response.authzid,
00324                             state->response.digesturi, 1,
00325                             state->response.cipher, NULL, NULL, NULL, NULL);
00326       if (rc)
00327         return GSASL_AUTHENTICATION_ERROR;
00328
00329       *output = digest_md5_print_finish (&state->finish);
00330       if (!*output)
00331         return GSASL_MALLOC_ERROR;
00332
00333       *output_len = strlen (*output);
00334
00335       state->step++;
00336       res = GSASL_OK;
00337       break;
00338
00339     default:
00340       res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00341       break;
00342     }
00343
00344   return res;
00345 }
00346
00347 void
00348 _gsasl_digest_md5_server_finish (Gsasl_session *sctx _GL_UNUSED,
00349                                  void *mech_data)
00350 {
00351   _Gsasl_digest_md5_server_state *state = mech_data;
00352
00353   if (!state)
00354     return;
00355
00356   digest_md5_free_challenge (&state->challenge);
00357   digest_md5_free_response (&state->response);
00358   digest_md5_free_finish (&state->finish);
00359
00360   free (state);
00361 }
00362
00363 int
00364 _gsasl_digest_md5_server_encode (Gsasl_session *sctx _GL_UNUSED,
00365                                  void *mech_data,
00366                                  const char *input,
00367                                  size_t input_len,
00368                                  char **output, size_t *output_len)
00369 {
00370   _Gsasl_digest_md5_server_state *state = mech_data;
00371   int res;
00372
00373   res = digest_md5_encode (input, input_len, output, output_len,
00374                            state->response.qop, state->sendseqnum,
00375                            state->kis);
00376   if (res)
00377     return res == -2 ? GSASL_NEEDS_MORE : GSASL_INTEGRITY_ERROR;
00378
00379   if (state->sendseqnum == 4294967295UL)
00380     state->sendseqnum = 0;
00381   else
00382     state->sendseqnum++;
00383
00384   return GSASL_OK;
00385 }
00386
00387 int
00388 _gsasl_digest_md5_server_decode (Gsasl_session *sctx _GL_UNUSED,
00389                                  void *mech_data,
00390                                  const char *input,
00391                                  size_t input_len,
00392                                  char **output, size_t *output_len)
00393 {
00394   _Gsasl_digest_md5_server_state *state = mech_data;
00395   int res;
00396
00397   res = digest_md5_decode (input, input_len, output, output_len,
00398                            state->response.qop, state->readseqnum,
00399                            state->kic);
00400   if (res)
00401     return res == -2 ? GSASL_NEEDS_MORE : GSASL_INTEGRITY_ERROR;
00402
00403   if (state->readseqnum == 4294967295UL)
00404     state->readseqnum = 0;
00405   else
00406     state->readseqnum++;
00407
00408   return GSASL_OK;
```

```
00409 }
```

## 5.115 external/server.c File Reference

```
#include <config.h>
#include "external.h"
#include <string.h>
```

### Functions

- int _gsasl_external_server_step (Gsasl_session *sctx, void *mech_data _GL_UNUSED, const char *input, size_t input_len, char **output, size_t *output_len)

### 5.115.1 Function Documentation

#### 5.115.1.1 _gsasl_external_server_step()

```
int _gsasl_external_server_step (
             Gsasl_session * sctx,
             void *mech_data _GL_UNUSED,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 31 of file external/server.c.

## 5.116 external/server.c

Go to the documentation of this file.

```
00001 /* server.c --- EXTERNAL mechanism as defined in RFC 2222, server side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "external.h"
00026
00027 /* Get memchr. */
00028 #include <string.h>
```

```
00029
00030 int
00031 _gsasl_external_server_step (Gsasl_session *sctx,
00032                               void *mech_data _GL_UNUSED,
00033                               const char *input, size_t input_len,
00034                               char **output, size_t *output_len)
00035 {
00036   int rc;
00037
00038   *output_len = 0;
00039   *output = NULL;
00040
00041   if (!input)
00042     return GSASL_NEEDS_MORE;
00043
00044   /* Quoting rfc2222bis-09:
00045    * extern-resp       = *( UTF8-char-no-nul )
00046    * UTF8-char-no-nul  = UTF8-1-no-nul / UTF8-2 / UTF8-3 / UTF8-4
00047    * UTF8-1-no-nul     = %x01-7F */
00048   if (memchr (input, '\0', input_len))
00049     return GSASL_MECHANISM_PARSE_ERROR;
00050
00051   /* FIXME: Validate that input is UTF-8. */
00052
00053   if (input_len > 0)
00054     rc = gsasl_property_set_raw (sctx, GSASL_AUTHZID, input, input_len);
00055   else
00056     rc = gsasl_property_set (sctx, GSASL_AUTHZID, NULL);
00057   if (rc != GSASL_OK)
00058     return rc;
00059
00060   return gsasl_callback (NULL, sctx, GSASL_VALIDATE_EXTERNAL);
00061 }
```

## 5.117 gs2/server.c File Reference

```
#include <config.h>
#include "gs2.h"
#include <stdlib.h>
#include <string.h>
#include <attribute.h>
#include "gss-extra.h"
#include "gs2helper.h"
#include "mechtools.h"
```

### Data Structures

- struct _Gsasl_gs2_server_state

### Typedefs

- typedef struct _Gsasl_gs2_server_state _Gsasl_gs2_server_state

### Functions

- int _gsasl_gs2_server_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_gs2_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_gs2_server_finish (Gsasl_session *sctx, void *mech_data)

## 5.117.1 Typedef Documentation

### 5.117.1.1 _Gsasl_gs2_server_state

```
typedef struct _Gsasl_gs2_server_state _Gsasl_gs2_server_state
```

Definition at line 50 of file gs2/server.c.

## 5.117.2 Function Documentation

### 5.117.2.1 _gsasl_gs2_server_finish()

```
void _gsasl_gs2_server_finish (
             Gsasl_session * sctx,
             void * mech_data )
```

Definition at line 298 of file gs2/server.c.

### 5.117.2.2 _gsasl_gs2_server_start()

```
int _gsasl_gs2_server_start (
             Gsasl_session * sctx,
             void ** mech_data )
```

Definition at line 118 of file gs2/server.c.

### 5.117.2.3 _gsasl_gs2_server_step()

```
int _gsasl_gs2_server_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 161 of file gs2/server.c.

# 5.118 gs2/server.c

Go to the documentation of this file.

```
00001 /* server.c --- SASL mechanism GS2, server side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "gs2.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get memcpy, strlen. */
00031 #include <string.h>
00032
00033 /* Get FALLTHROUGH. */
00034 #include <attribute.h>
00035
00036 #include "gss-extra.h"
00037 #include "gs2helper.h"
00038 #include "mechtools.h"
00039
00040 struct _Gsasl_gs2_server_state
00041 {
00042   /* steps: 0 = first state, 1 = initial, 2 = processing, 3 = done */
00043   int step;
00044   gss_name_t client;
00045   gss_cred_id_t cred;
00046   gss_ctx_id_t context;
00047   gss_OID mech_oid;
00048   struct gss_channel_bindings_struct cb;
00049 };
00050 typedef struct _Gsasl_gs2_server_state _Gsasl_gs2_server_state;
00051
00052 /* Populate state->cred with credential to use for connection.  Return
00053    GSASL_OK on success or an error code.  */
00054 static int
00055 gs2_get_cred (Gsasl_session *sctx, _Gsasl_gs2_server_state *state)
00056 {
00057   OM_uint32 maj_stat, min_stat;
00058   gss_buffer_desc bufdesc;
00059   const char *service = gsasl_property_get (sctx, GSASL_SERVICE);
00060   const char *hostname = gsasl_property_get (sctx, GSASL_HOSTNAME);
00061   gss_name_t server;
00062   gss_OID_set_desc oid_set;
00063   gss_OID_set actual_mechs;
00064   int present;
00065
00066   if (!service)
00067     return GSASL_NO_SERVICE;
00068   if (!hostname)
00069     return GSASL_NO_HOSTNAME;
00070
00071   bufdesc.length = asprintf ((char **) &bufdesc.value, "%s@%s",
00072                              service, hostname);
00073   if (bufdesc.length <= 0 || bufdesc.value == NULL)
00074     return GSASL_MALLOC_ERROR;
00075
00076   maj_stat = gss_import_name (&min_stat, &bufdesc,
00077                               GSS_C_NT_HOSTBASED_SERVICE, &server);
00078   free (bufdesc.value);
00079   if (GSS_ERROR (maj_stat))
00080     return GSASL_GSSAPI_IMPORT_NAME_ERROR;
00081
00082   /* Attempt to get a credential for our mechanism.  */
```

```
00083
00084   oid_set.count = 1;
00085   oid_set.elements = state->mech_oid;
00086
00087   maj_stat = gss_acquire_cred (&min_stat, server, 0,
00088                                &oid_set, GSS_C_ACCEPT,
00089                                &state->cred, &actual_mechs, NULL);
00090   gss_release_name (&min_stat, &server);
00091   if (GSS_ERROR (maj_stat))
00092     return GSASL_GSSAPI_ACQUIRE_CRED_ERROR;
00093
00094   /* Now double check that the credential actually was for our
00095      mechanism... */
00096
00097   maj_stat = gss_test_oid_set_member (&min_stat, state->mech_oid,
00098                                       actual_mechs, &present);
00099   if (GSS_ERROR (maj_stat))
00100     {
00101       gss_release_oid_set (&min_stat, &actual_mechs);
00102       return GSASL_GSSAPI_TEST_OID_SET_MEMBER_ERROR;
00103     }
00104
00105   maj_stat = gss_release_oid_set (&min_stat, &actual_mechs);
00106   if (GSS_ERROR (maj_stat))
00107     return GSASL_GSSAPI_RELEASE_OID_SET_ERROR;
00108
00109   if (!present)
00110     return GSASL_GSSAPI_ACQUIRE_CRED_ERROR;
00111
00112   return GSASL_OK;
00113 }
00114
00115 /* Initialize GS2 state into MECH_DATA.  Return GSASL_OK if GS2 is
00116    ready and initialization succeeded, or an error code. */
00117 int
00118 _gsasl_gs2_server_start (Gsasl_session *sctx, void **mech_data)
00119 {
00120   _Gsasl_gs2_server_state *state;
00121   int res;
00122
00123   state = (_Gsasl_gs2_server_state *) malloc (sizeof (*state));
00124   if (state == NULL)
00125     return GSASL_MALLOC_ERROR;
00126
00127   res = gs2_get_oid (sctx, &state->mech_oid);
00128   if (res != GSASL_OK)
00129     {
00130       free (state);
00131       return res;
00132     }
00133
00134   state->step = 0;
00135   state->context = GSS_C_NO_CONTEXT;
00136   state->cred = GSS_C_NO_CREDENTIAL;
00137   state->client = NULL;
00138   /* The initiator-address-type and acceptor-address-type fields of
00139      the GSS-CHANNEL-BINDINGS structure MUST be set to 0.  The
00140      initiator-address and acceptor-address fields MUST be the empty
00141      string. */
00142   state->cb.initiator_addrtype = 0;
00143   state->cb.initiator_address.length = 0;
00144   state->cb.initiator_address.value = NULL;
00145   state->cb.acceptor_addrtype = 0;
00146   state->cb.acceptor_address.length = 0;
00147   state->cb.acceptor_address.value = NULL;
00148   state->cb.application_data.length = 0;
00149   state->cb.application_data.value = NULL;
00150
00151   *mech_data = state;
00152
00153   return GSASL_OK;
00154 }
00155
00156 /* Perform one GS2 step.  GS2 state is in MECH_DATA.  Any data from
00157    client is provided in INPUT/INPUT_LEN and output from server is
00158    expected to be put in newly allocated OUTPUT/OUTPUT_LEN.  Return
00159    GSASL_NEEDS_MORE or GSASL_OK on success, or an error code.  */
00160 int
00161 _gsasl_gs2_server_step (Gsasl_session *sctx,
00162                         void *mech_data,
00163                         const char *input, size_t input_len,
00164                         char **output, size_t *output_len)
00165 {
00166   _Gsasl_gs2_server_state *state = mech_data;
00167   gss_buffer_desc bufdesc1, bufdesc2;
00168   OM_uint32 maj_stat, min_stat;
00169   gss_buffer_desc client_name;
```

```
00170   gss_OID mech_type;
00171   int res;
00172   OM_uint32 ret_flags;
00173   int free_bufdesc1 = 0;
00174
00175   *output = NULL;
00176   *output_len = 0;
00177   bufdesc1.value = (char *) input;
00178   bufdesc1.length = input_len;
00179
00180   switch (state->step)
00181     {
00182     case 0:
00183       res = gs2_get_cred (sctx, state);
00184       if (res != GSASL_OK)
00185         return res;
00186       if (input_len == 0)
00187         {
00188           res = GSASL_NEEDS_MORE;
00189           break;
00190         }
00191       state->step++;
00192       FALLTHROUGH;              /* fall through */
00193
00194     case 1:
00195       {
00196         char *authzid;
00197         size_t headerlen;
00198
00199         res = _gsasl_parse_gs2_header (input, input_len,
00200                                        &authzid, &headerlen);
00201         if (res != GSASL_OK)
00202           return res;
00203
00204         if (authzid)
00205           {
00206             res = gsasl_property_set (sctx, GSASL_AUTHZID, authzid);
00207             free (authzid);
00208             if (res != GSASL_OK)
00209               return res;
00210           }
00211
00212         state->cb.application_data.value = (char *) input;
00213         state->cb.application_data.length = headerlen;
00214
00215         bufdesc2.value = (char *) input + headerlen;
00216         bufdesc2.length = input_len - headerlen;
00217
00218         maj_stat = gss_encapsulate_token (&bufdesc2, state->mech_oid,
00219                                           &bufdesc1);
00220         if (GSS_ERROR (maj_stat))
00221           return GSASL_GSSAPI_ENCAPSULATE_TOKEN_ERROR;
00222
00223         free_bufdesc1 = 1;
00224       }
00225       state->step++;
00226       FALLTHROUGH;              /* fall through */
00227
00228     case 2:
00229       if (state->client)
00230         {
00231           gss_release_name (&min_stat, &state->client);
00232           state->client = GSS_C_NO_NAME;
00233         }
00234
00235       maj_stat = gss_accept_sec_context (&min_stat,
00236                                          &state->context,
00237                                          state->cred,
00238                                          &bufdesc1,
00239                                          &state->cb,
00240                                          &state->client,
00241                                          &mech_type,
00242                                          &bufdesc2, &ret_flags, NULL, NULL);
00243       if (maj_stat != GSS_S_COMPLETE && maj_stat != GSS_S_CONTINUE_NEEDED)
00244         return GSASL_GSSAPI_ACCEPT_SEC_CONTEXT_ERROR;
00245
00246       if (maj_stat == GSS_S_COMPLETE)
00247         {
00248           state->step++;
00249
00250           if (!(ret_flags & GSS_C_MUTUAL_FLAG))
00251             return GSASL_MECHANISM_PARSE_ERROR;
00252
00253           maj_stat = gss_display_name (&min_stat, state->client,
00254                                        &client_name, &mech_type);
00255           if (GSS_ERROR (maj_stat))
00256             return GSASL_GSSAPI_DISPLAY_NAME_ERROR;
```

```
00257
00258           res = gsasl_property_set_raw (sctx, GSASL_GSSAPI_DISPLAY_NAME,
00259                                         client_name.value,
00260                                         client_name.length);
00261           if (res != GSASL_OK)
00262             return res;
00263
00264           res = gsasl_callback (NULL, sctx, GSASL_VALIDATE_GSSAPI);
00265         }
00266       else
00267         res = GSASL_NEEDS_MORE;
00268
00269       if (free_bufdesc1)
00270         {
00271           maj_stat = gss_release_buffer (&min_stat, &bufdesc1);
00272           if (GSS_ERROR (maj_stat))
00273             return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00274         }
00275
00276       *output = malloc (bufdesc2.length);
00277       if (!*output)
00278         return GSASL_MALLOC_ERROR;
00279       memcpy (*output, bufdesc2.value, bufdesc2.length);
00280       *output_len = bufdesc2.length;
00281
00282       maj_stat = gss_release_buffer (&min_stat, &bufdesc2);
00283       if (GSS_ERROR (maj_stat))
00284         return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00285       break;
00286
00287     default:
00288       res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00289       break;
00290     }
00291
00292   return res;
00293 }
00294
00295 /* Cleanup GS2 state context, i.e., release memory associated with
00296    buffers in MECH_DATA state. */
00297 void
00298 _gsasl_gs2_server_finish (Gsasl_session *sctx, void *mech_data)
00299 {
00300   _Gsasl_gs2_server_state *state = mech_data;
00301   OM_uint32 min_stat;
00302   (void) sctx;
00303
00304   if (!state)
00305     return;
00306
00307   if (state->context != GSS_C_NO_CONTEXT)
00308     gss_delete_sec_context (&min_stat, &state->context, GSS_C_NO_BUFFER);
00309
00310   if (state->cred != GSS_C_NO_CREDENTIAL)
00311     gss_release_cred (&min_stat, &state->cred);
00312
00313   if (state->client != GSS_C_NO_NAME)
00314     gss_release_name (&min_stat, &state->client);
00315
00316   free (state);
00317 }
```

## 5.119 gssapi/server.c File Reference

```
#include <config.h>
#include <stdlib.h>
#include <string.h>
#include "x-gssapi.h"
#include "gss-extra.h"
```

### Data Structures

- struct _Gsasl_gssapi_server_state

### Typedefs

- typedef struct _Gsasl_gssapi_server_state _Gsasl_gssapi_server_state

### Functions

- int _gsasl_gssapi_server_start (Gsasl_session ∗sctx, void ∗∗mech_data)
- int _gsasl_gssapi_server_step (Gsasl_session ∗sctx, void ∗mech_data, const char ∗input, size_t input_len, char ∗∗output, size_t ∗output_len)
- void _gsasl_gssapi_server_finish (Gsasl_session ∗sctx, void ∗mech_data)

## 5.119.1 Typedef Documentation

### 5.119.1.1 _Gsasl_gssapi_server_state

```
typedef struct _Gsasl_gssapi_server_state _Gsasl_gssapi_server_state
```

Definition at line 43 of file gssapi/server.c.

## 5.119.2 Function Documentation

### 5.119.2.1 _gsasl_gssapi_server_finish()

```
void _gsasl_gssapi_server_finish (
            Gsasl_session * sctx,
            void * mech_data )
```

Definition at line 265 of file gssapi/server.c.

### 5.119.2.2 _gsasl_gssapi_server_start()

```
int _gsasl_gssapi_server_start (
            Gsasl_session * sctx,
            void ** mech_data )
```

Definition at line 46 of file gssapi/server.c.

### 5.119.2.3 _gsasl_gssapi_server_step()

```
int _gsasl_gssapi_server_step (
            Gsasl_session * sctx,
            void * mech_data,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )
```

Definition at line 64 of file gssapi/server.c.

## 5.120 gssapi/server.c

Go to the documentation of this file.

```
00001 /* server.c --- SASL mechanism GSSAPI as defined in RFC 4752, server side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get malloc, free. */
00025 #include <stdlib.h>
00026
00027 /* Get memcpy, strlen. */
00028 #include <string.h>
00029
00030 /* Get specification. */
00031 #include "x-gssapi.h"
00032
00033 /* For GSS-API prototypes. */
00034 #include "gss-extra.h"
00035
00036 struct _Gsasl_gssapi_server_state
00037 {
00038   int step;
00039   gss_name_t client;
00040   gss_cred_id_t cred;
00041   gss_ctx_id_t context;
00042 };
00043 typedef struct _Gsasl_gssapi_server_state _Gsasl_gssapi_server_state;
00044
00045 int
00046 _gsasl_gssapi_server_start (Gsasl_session *sctx, void **mech_data)
00047 {
00048   _Gsasl_gssapi_server_state *state;
00049
00050   state = (_Gsasl_gssapi_server_state *) malloc (sizeof (*state));
00051   if (state == NULL)
00052     return GSASL_MALLOC_ERROR;
00053
00054   state->step = 0;
00055   state->cred = GSS_C_NO_CREDENTIAL;
00056   state->context = GSS_C_NO_CONTEXT;
00057   state->client = NULL;
00058   *mech_data = state;
00059
00060   return GSASL_OK;
00061 }
00062
00063 int
00064 _gsasl_gssapi_server_step (Gsasl_session *sctx,
00065                            void *mech_data,
00066                            const char *input, size_t input_len,
00067                            char **output, size_t *output_len)
00068 {
00069   _Gsasl_gssapi_server_state *state = mech_data;
00070   gss_buffer_desc bufdesc1, bufdesc2;
00071   OM_uint32 maj_stat, min_stat;
00072   gss_buffer_desc client_name;
00073   gss_OID mech_type;
00074   char tmp[4];
00075   int res;
00076
00077   *output = NULL;
00078   *output_len = 0;
00079
00080   switch (state->step)
00081     {
00082     case 0:
```

```
00083       {
00084         gss_name_t server;
00085         const char *service;
00086         const char *hostname;
00087 
00088         if (input_len == 0)
00089           {
00090             res = GSASL_NEEDS_MORE;
00091             break;
00092           }
00093 
00094         service = gsasl_property_get (sctx, GSASL_SERVICE);
00095         if (!service)
00096           return GSASL_NO_SERVICE;
00097 
00098         hostname = gsasl_property_get (sctx, GSASL_HOSTNAME);
00099         if (!hostname)
00100           return GSASL_NO_HOSTNAME;
00101 
00102         /* FIXME: Use asprintf. */
00103 
00104         bufdesc1.length = strlen (service) + strlen ("@")
00105           + strlen (hostname) + 1;
00106         bufdesc1.value = malloc (bufdesc1.length);
00107         if (bufdesc1.value == NULL)
00108           return GSASL_MALLOC_ERROR;
00109 
00110         sprintf (bufdesc1.value, "%s@%s", service, hostname);
00111 
00112         maj_stat = gss_import_name (&min_stat, &bufdesc1,
00113                                     GSS_C_NT_HOSTBASED_SERVICE, &server);
00114         free (bufdesc1.value);
00115         if (GSS_ERROR (maj_stat))
00116           return GSASL_GSSAPI_IMPORT_NAME_ERROR;
00117 
00118         maj_stat = gss_acquire_cred (&min_stat, server, 0,
00119                                      GSS_C_NULL_OID_SET, GSS_C_ACCEPT,
00120                                      &state->cred, NULL, NULL);
00121         gss_release_name (&min_stat, &server);
00122         if (GSS_ERROR (maj_stat))
00123           return GSASL_GSSAPI_ACQUIRE_CRED_ERROR;
00124       }
00125       state->step++;
00126       /* fall through */
00127 
00128     case 1:
00129       bufdesc1.value = (void *) input;
00130       bufdesc1.length = input_len;
00131       if (state->client)
00132         {
00133           gss_release_name (&min_stat, &state->client);
00134           state->client = GSS_C_NO_NAME;
00135         }
00136 
00137       maj_stat = gss_accept_sec_context (&min_stat,
00138                                          &state->context,
00139                                          state->cred,
00140                                          &bufdesc1,
00141                                          GSS_C_NO_CHANNEL_BINDINGS,
00142                                          &state->client,
00143                                          &mech_type,
00144                                          &bufdesc2, NULL, NULL, NULL);
00145       if (maj_stat != GSS_S_COMPLETE && maj_stat != GSS_S_CONTINUE_NEEDED)
00146         return GSASL_GSSAPI_ACCEPT_SEC_CONTEXT_ERROR;
00147 
00148       if (maj_stat == GSS_S_COMPLETE)
00149         state->step++;
00150 
00151       if (maj_stat == GSS_S_CONTINUE_NEEDED || bufdesc2.length > 0)
00152         {
00153           *output = malloc (bufdesc2.length);
00154           if (!*output)
00155             return GSASL_MALLOC_ERROR;
00156           memcpy (*output, bufdesc2.value, bufdesc2.length);
00157           *output_len = bufdesc2.length;
00158         }
00159 
00160       maj_stat = gss_release_buffer (&min_stat, &bufdesc2);
00161       if (GSS_ERROR (maj_stat))
00162         return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00163 
00164       if (maj_stat == GSS_S_CONTINUE_NEEDED || *output_len > 0)
00165         {
00166           res = GSASL_NEEDS_MORE;
00167           break;
00168         }
00169       /* fall through */
```

```
00170
00171     case 2:
00172       memset (tmp, 0xFF, 4);
00173       tmp[0] = GSASL_QOP_AUTH;
00174       bufdesc1.length = 4;
00175       bufdesc1.value = tmp;
00176       maj_stat = gss_wrap (&min_stat, state->context, 0, GSS_C_QOP_DEFAULT,
00177                            &bufdesc1, NULL, &bufdesc2);
00178       if (GSS_ERROR (maj_stat))
00179         return GSASL_GSSAPI_WRAP_ERROR;
00180
00181       *output = malloc (bufdesc2.length);
00182       if (!*output)
00183         return GSASL_MALLOC_ERROR;
00184       memcpy (*output, bufdesc2.value, bufdesc2.length);
00185       *output_len = bufdesc2.length;
00186
00187       maj_stat = gss_release_buffer (&min_stat, &bufdesc2);
00188       if (GSS_ERROR (maj_stat))
00189         return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00190
00191       state->step++;
00192       res = GSASL_NEEDS_MORE;
00193       break;
00194
00195     case 3:
00196       bufdesc1.value = (void *) input;
00197       bufdesc1.length = input_len;
00198       maj_stat = gss_unwrap (&min_stat, state->context, &bufdesc1,
00199                              &bufdesc2, NULL, NULL);
00200       if (GSS_ERROR (maj_stat))
00201         return GSASL_GSSAPI_UNWRAP_ERROR;
00202
00203       /* [RFC 2222 section 7.2.1]:
00204          The client passes this token to GSS_Unwrap and interprets the
00205          first octet of resulting cleartext as a bit-mask specifying
00206          the security layers supported by the server and the second
00207          through fourth octets as the maximum size output_message to
00208          send to the server.  The client then constructs data, with
00209          the first octet containing the bit-mask specifying the
00210          selected security layer, the second through fourth octets
00211          containing in network byte order the maximum size
00212          output_message the client is able to receive, and the
00213          remaining octets containing the authorization identity.  The
00214          client passes the data to GSS_Wrap with conf_flag set to
00215          FALSE, and responds with the generated output_message.  The
00216          client can then consider the server authenticated. */
00217
00218       if (bufdesc2.length < 4)
00219         return GSASL_AUTHENTICATION_ERROR;
00220
00221       if ((((char *) bufdesc2.value)[0] & GSASL_QOP_AUTH) == 0)
00222         {
00223           /* Integrity or privacy unsupported */
00224           maj_stat = gss_release_buffer (&min_stat, &bufdesc2);
00225           return GSASL_GSSAPI_UNSUPPORTED_PROTECTION_ERROR;
00226         }
00227
00228       if (bufdesc2.length > 4)
00229         gsasl_property_set_raw (sctx, GSASL_AUTHZID,
00230                                 (char *) bufdesc2.value + 4,
00231                                 bufdesc2.length - 4);
00232       else
00233         gsasl_property_set (sctx, GSASL_AUTHZID, NULL);
00234
00235       maj_stat = gss_display_name (&min_stat, state->client,
00236                                    &client_name, &mech_type);
00237       if (GSS_ERROR (maj_stat))
00238         return GSASL_GSSAPI_DISPLAY_NAME_ERROR;
00239
00240       gsasl_property_set_raw (sctx, GSASL_GSSAPI_DISPLAY_NAME,
00241                               client_name.value, client_name.length);
00242
00243       maj_stat = gss_release_buffer (&min_stat, &client_name);
00244       if (GSS_ERROR (maj_stat))
00245         return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00246
00247       maj_stat = gss_release_buffer (&min_stat, &bufdesc2);
00248       if (GSS_ERROR (maj_stat))
00249         return GSASL_GSSAPI_RELEASE_BUFFER_ERROR;
00250
00251       res = gsasl_callback (NULL, sctx, GSASL_VALIDATE_GSSAPI);
00252
00253       state->step++;
00254       break;
00255
00256     default:
```

```
00257       res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00258       break;
00259     }
00260
00261   return res;
00262 }
00263
00264 void
00265 _gsasl_gssapi_server_finish (Gsasl_session *sctx, void *mech_data)
00266 {
00267   _Gsasl_gssapi_server_state *state = mech_data;
00268   OM_uint32 min_stat;
00269
00270   if (!state)
00271     return;
00272
00273   if (state->context != GSS_C_NO_CONTEXT)
00274     gss_delete_sec_context (&min_stat, &state->context, GSS_C_NO_BUFFER);
00275
00276   if (state->cred != GSS_C_NO_CREDENTIAL)
00277     gss_release_cred (&min_stat, &state->cred);
00278
00279   if (state->client != GSS_C_NO_NAME)
00280     gss_release_name (&min_stat, &state->client);
00281
00282   free (state);
00283 }
```

## 5.121 login/server.c File Reference

```
#include <config.h>
#include <stdlib.h>
#include <string.h>
#include "login.h"
```

### Data Structures

- struct _Gsasl_login_server_state

### Macros

- #define CHALLENGE_USERNAME "User Name"
- #define CHALLENGE_PASSWORD "Password"

### Functions

- int _gsasl_login_server_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
- int _gsasl_login_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_login_server_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)

### 5.121.1 Macro Definition Documentation

#### 5.121.1.1 CHALLENGE_PASSWORD

```
#define CHALLENGE_PASSWORD "Password"
```

Definition at line 41 of file login/server.c.

#### 5.121.1.2 CHALLENGE_USERNAME

```
#define CHALLENGE_USERNAME "User Name"
```

Definition at line 40 of file login/server.c.

### 5.121.2 Function Documentation

#### 5.121.2.1 _gsasl_login_server_finish()

```
void _gsasl_login_server_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 147 of file login/server.c.

#### 5.121.2.2 _gsasl_login_server_start()

```
int _gsasl_login_server_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 44 of file login/server.c.

#### 5.121.2.3 _gsasl_login_server_step()

```
int _gsasl_login_server_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 58 of file login/server.c.

## 5.122 login/server.c

Go to the documentation of this file.

```
00001 /* server.c --- Non-standard SASL mechanism LOGIN, server side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
```

```
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get malloc, free. */
00025 #include <stdlib.h>
00026
00027 /* Get strdup, strlen. */
00028 #include <string.h>
00029
00030 /* Get specification. */
00031 #include "login.h"
00032
00033 struct _Gsasl_login_server_state
00034 {
00035   int step;
00036   char *username;
00037   char *password;
00038 };
00039
00040 #define CHALLENGE_USERNAME "User Name"
00041 #define CHALLENGE_PASSWORD "Password"
00042
00043 int
00044 _gsasl_login_server_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
00045 {
00046   struct _Gsasl_login_server_state *state;
00047
00048   state = calloc (1, sizeof (*state));
00049   if (state == NULL)
00050     return GSASL_MALLOC_ERROR;
00051
00052   *mech_data = state;
00053
00054   return GSASL_OK;
00055 }
00056
00057 int
00058 _gsasl_login_server_step (Gsasl_session *sctx,
00059                           void *mech_data,
00060                           const char *input, size_t input_len,
00061                           char **output, size_t *output_len)
00062 {
00063   struct _Gsasl_login_server_state *state = mech_data;
00064   int res;
00065
00066   switch (state->step)
00067     {
00068     case 0:
00069       *output = strdup (CHALLENGE_USERNAME);
00070       if (!*output)
00071         return GSASL_MALLOC_ERROR;
00072       *output_len = strlen (CHALLENGE_USERNAME);
00073
00074       state->step++;
00075       res = GSASL_NEEDS_MORE;
00076       break;
00077
00078     case 1:
00079       if (input_len == 0)
00080         return GSASL_MECHANISM_PARSE_ERROR;
00081
00082       state->username = strndup (input, input_len);
00083       if (state->username == NULL)
00084         return GSASL_MALLOC_ERROR;
00085
00086       if (input_len != strlen (state->username))
00087         return GSASL_MECHANISM_PARSE_ERROR;
00088
00089       *output = strdup (CHALLENGE_PASSWORD);
00090       if (!*output)
00091         return GSASL_MALLOC_ERROR;
00092       *output_len = strlen (CHALLENGE_PASSWORD);
00093
00094       state->step++;
00095       res = GSASL_NEEDS_MORE;
00096       break;
00097
00098     case 2:
00099       if (input_len == 0)
00100         return GSASL_MECHANISM_PARSE_ERROR;
00101
00102       state->password = strndup (input, input_len);
00103       if (state->password == NULL)
```

```
00104         return GSASL_MALLOC_ERROR;
00105
00106       if (input_len != strlen (state->password))
00107         return GSASL_MECHANISM_PARSE_ERROR;
00108
00109       res = gsasl_property_set (sctx, GSASL_AUTHID, state->username);
00110       if (res != GSASL_OK)
00111         return res;
00112       res = gsasl_property_set (sctx, GSASL_PASSWORD, state->password);
00113       if (res != GSASL_OK)
00114         return res;
00115
00116       res = gsasl_callback (NULL, sctx, GSASL_VALIDATE_SIMPLE);
00117       if (res == GSASL_NO_CALLBACK)
00118         {
00119           const char *key;
00120
00121           gsasl_property_free (sctx, GSASL_AUTHZID);
00122           gsasl_property_free (sctx, GSASL_PASSWORD);
00123
00124           key = gsasl_property_get (sctx, GSASL_PASSWORD);
00125
00126           if (key && strlen (state->password) == strlen (key) &&
00127               strcmp (state->password, key) == 0)
00128             res = GSASL_OK;
00129           else
00130             res = GSASL_AUTHENTICATION_ERROR;
00131         }
00132
00133       *output_len = 0;
00134       *output = NULL;
00135       state->step++;
00136       break;
00137
00138     default:
00139       res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00140       break;
00141     }
00142
00143   return res;
00144 }
00145
00146 void
00147 _gsasl_login_server_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)
00148 {
00149   struct _Gsasl_login_server_state *state = mech_data;
00150
00151   if (!state)
00152     return;
00153
00154   free (state->username);
00155   free (state->password);
00156   free (state);
00157 }
```

## 5.123 openid20/server.c File Reference

```
#include <config.h>
#include "openid20.h"
#include <string.h>
#include <stdlib.h>
#include "mechtools.h"
```

### Data Structures

- struct openid20_server_state

### Functions

- int _gsasl_openid20_server_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
- int _gsasl_openid20_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_↩ len, char **output, size_t *output_len)
- void _gsasl_openid20_server_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)

### 5.123.1 Function Documentation

#### 5.123.1.1 _gsasl_openid20_server_finish()

```
void _gsasl_openid20_server_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 190 of file openid20/server.c.

#### 5.123.1.2 _gsasl_openid20_server_start()

```
int _gsasl_openid20_server_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 43 of file openid20/server.c.

#### 5.123.1.3 _gsasl_openid20_server_step()

```
int _gsasl_openid20_server_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 58 of file openid20/server.c.

## 5.124 openid20/server.c

Go to the documentation of this file.

```
00001 /* server.c --- OPENID20 mechanism, server side.
00002  * Copyright (C) 2011-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "openid20.h"
00026
00027 /* Get strdup, strlen. */
```

```
00028 #include <string.h>
00029
00030 /* Get calloc, free. */
00031 #include <stdlib.h>
00032
00033 /* Get _gsasl_parse_gs2_header. */
00034 #include "mechtools.h"
00035
00036 struct openid20_server_state
00037 {
00038   int step;
00039   int allow_error_step;
00040 };
00041
00042 int
00043 _gsasl_openid20_server_start (Gsasl_session *sctx _GL_UNUSED,
00044                               void **mech_data)
00045 {
00046   struct openid20_server_state *state;
00047
00048   state = (struct openid20_server_state *) calloc (1, sizeof (*state));
00049   if (state == NULL)
00050     return GSASL_MALLOC_ERROR;
00051
00052   *mech_data = state;
00053
00054   return GSASL_OK;
00055 }
00056
00057 int
00058 _gsasl_openid20_server_step (Gsasl_session *sctx,
00059                              void *mech_data,
00060                              const char *input, size_t input_len,
00061                              char **output, size_t *output_len)
00062 {
00063   struct openid20_server_state *state = mech_data;
00064   int res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00065
00066   *output_len = 0;
00067   *output = NULL;
00068
00069   switch (state->step)
00070     {
00071     case 0:
00072       {
00073         const char *p;
00074         char *authzid;
00075         size_t headerlen;
00076
00077         if (input_len == 0)
00078           return GSASL_NEEDS_MORE;
00079
00080         res = _gsasl_parse_gs2_header (input, input_len,
00081                                        &authzid, &headerlen);
00082         if (res != GSASL_OK)
00083           return res;
00084
00085         if (authzid)
00086           {
00087             res = gsasl_property_set (sctx, GSASL_AUTHZID, authzid);
00088             free (authzid);
00089             if (res != GSASL_OK)
00090               return res;
00091           }
00092
00093         input += headerlen;
00094         input_len -= headerlen;
00095
00096         res = gsasl_property_set_raw (sctx, GSASL_AUTHID, input, input_len);
00097         if (res != GSASL_OK)
00098           return res;
00099
00100         p = gsasl_property_get (sctx, GSASL_OPENID20_REDIRECT_URL);
00101         if (!p || !*p)
00102           return GSASL_NO_OPENID20_REDIRECT_URL;
00103
00104         *output_len = strlen (p);
00105         *output = malloc (*output_len);
00106         if (!*output)
00107           return GSASL_MALLOC_ERROR;
00108
00109         memcpy (*output, p, *output_len);
00110
00111         res = GSASL_NEEDS_MORE;
00112         state->step++;
00113         break;
00114       }
```

```
00115
00116     case 1:
00117       {
00118         const char *outcome_data;
00119
00120         if (!(input_len == 1 && *input == '='))
00121           return GSASL_MECHANISM_PARSE_ERROR;
00122
00123         res = gsasl_callback (NULL, sctx, GSASL_VALIDATE_OPENID20);
00124         if (res != GSASL_OK)
00125           {
00126             const char *failstr = "openid.error=fail";
00127
00128             *output_len = strlen (failstr);
00129             *output = strdup (failstr);
00130             if (!*output)
00131               return GSASL_MALLOC_ERROR;
00132
00133             /* [RFC4422] Section 3.6 explicitly prohibits additional
00134                information in an unsuccessful authentication outcome.
00135                Therefore, the openid.error and openid.error_code are
00136                to be sent as an additional challenge in the event of
00137                an unsuccessful outcome.  In this case, as the protocol
00138                is lock step, the client will follow with an additional
00139                exchange containing "=", after which the server will
00140                respond with an application-level outcome. */
00141
00142             state->allow_error_step = 1;
00143             state->step++;
00144             return GSASL_NEEDS_MORE;
00145           }
00146
00147         outcome_data = gsasl_property_get (sctx, GSASL_OPENID20_OUTCOME_DATA);
00148         if (outcome_data)
00149           {
00150             *output = strdup (outcome_data);
00151             if (!*output)
00152               return GSASL_MALLOC_ERROR;
00153             *output_len = strlen (*output);
00154           }
00155         else
00156           {
00157             *output = NULL;
00158             *output_len = 0;
00159           }
00160
00161         res = GSASL_OK;
00162         state->step++;
00163       }
00164       break;
00165
00166     case 2:
00167       {
00168         /* We only get here when the previous step signalled an error
00169            to the client.  */
00170
00171         if (!state->allow_error_step)
00172           return GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00173
00174         if (!(input_len == 1 && *input == '='))
00175           return GSASL_MECHANISM_PARSE_ERROR;
00176
00177         res = GSASL_AUTHENTICATION_ERROR;
00178         state->step++;
00179       }
00180       break;
00181
00182     default:
00183       break;
00184     }
00185
00186   return res;
00187 }
00188
00189 void
00190 _gsasl_openid20_server_finish (Gsasl_session *sctx _GL_UNUSED,
00191                                void *mech_data)
00192 {
00193   struct openid20_server_state *state = mech_data;
00194
00195   if (!state)
00196     return;
00197
00198   free (state);
00199 }
```

## 5.125 plain/server.c File Reference

```
#include <config.h>
#include "plain.h"
#include <string.h>
#include <stdlib.h>
```

### Functions

- int _gsasl_plain_server_step (Gsasl_session *sctx, void *mech_data _GL_UNUSED, const char *input, size_t input_len, char **output, size_t *output_len)

### 5.125.1 Function Documentation

#### 5.125.1.1 _gsasl_plain_server_step()

```
int _gsasl_plain_server_step (
             Gsasl_session * sctx,
             void *mech_data _GL_UNUSED,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 34 of file plain/server.c.

## 5.126 plain/server.c

Go to the documentation of this file.

```
00001 /* server.c --- SASL mechanism PLAIN as defined in RFC 2595, server side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "plain.h"
00026
00027 /* Get memcpy, memchr, strlen. */
00028 #include <string.h>
00029
00030 /* Get malloc, free. */
00031 #include <stdlib.h>
00032
```

```
00033 int
00034 _gsasl_plain_server_step (Gsasl_session *sctx,
00035                          void *mech_data _GL_UNUSED,
00036                          const char *input, size_t input_len,
00037                          char **output, size_t *output_len)
00038 {
00039   const char *authzidptr = input;
00040   const char *authidptr = NULL;
00041   const char *passwordptr = NULL;
00042   char *passwdz = NULL, *passprep = NULL, *authidprep = NULL;
00043   int res;
00044
00045   *output_len = 0;
00046   *output = NULL;
00047
00048   if (input_len == 0)
00049     return GSASL_NEEDS_MORE;
00050
00051   /* Parse input. */
00052   {
00053     size_t tmplen;
00054
00055     authidptr = memchr (input, 0, input_len - 1);
00056     if (authidptr)
00057       {
00058         authidptr++;
00059         passwordptr = memchr (authidptr, 0, input_len - strlen (input) - 1);
00060         if (passwordptr)
00061           passwordptr++;
00062         else
00063           return GSASL_MECHANISM_PARSE_ERROR;
00064       }
00065     else
00066       return GSASL_MECHANISM_PARSE_ERROR;
00067
00068     /* As the NUL (U+0000) character is used as a deliminator, the NUL
00069        (U+0000) character MUST NOT appear in authzid, authcid, or passwd
00070        productions. */
00071     tmplen = input_len - (size_t) (passwordptr - input);
00072     if (memchr (passwordptr, 0, tmplen))
00073       return GSASL_MECHANISM_PARSE_ERROR;
00074   }
00075
00076   /* Store authid, after preparing it... */
00077   {
00078     res = gsasl_saslprep (authidptr, GSASL_ALLOW_UNASSIGNED,
00079                           &authidprep, NULL);
00080     if (res != GSASL_OK)
00081       return res;
00082
00083     res = gsasl_property_set (sctx, GSASL_AUTHID, authidprep);
00084     if (res != GSASL_OK)
00085       return res;
00086
00087     /* Store authzid, if absent, use SASLprep(authcid). */
00088     if (*authzidptr == '\0')
00089       res = gsasl_property_set (sctx, GSASL_AUTHZID, authidprep);
00090     else
00091       res = gsasl_property_set (sctx, GSASL_AUTHZID, authzidptr);
00092     if (res != GSASL_OK)
00093       return res;
00094
00095     free (authidprep);
00096   }
00097
00098   /* Store passwd, after preparing it... */
00099   {
00100     size_t passwdzlen = input_len - (size_t) (passwordptr - input);
00101
00102     /* Need to zero terminate password... */
00103     passwdz = malloc (passwdzlen + 1);
00104     if (passwdz == NULL)
00105       return GSASL_MALLOC_ERROR;
00106     memcpy (passwdz, passwordptr, passwdzlen);
00107     passwdz[passwdzlen] = '\0';
00108
00109     res = gsasl_saslprep (passwdz, GSASL_ALLOW_UNASSIGNED, &passprep, NULL);
00110     free (passwdz);
00111     if (res != GSASL_OK)
00112       return res;
00113
00114     res = gsasl_property_set (sctx, GSASL_PASSWORD, passprep);
00115     if (res != GSASL_OK)
00116       return res;
00117   }
00118
00119   /* Authorization.  Let application verify credentials internally,
```

```
00120       but fall back to deal with it locally... */
00121     res = gsasl_callback (NULL, sctx, GSASL_VALIDATE_SIMPLE);
00122     if (res == GSASL_NO_CALLBACK)
00123       {
00124         const char *key;
00125         char *normkey;
00126
00127         gsasl_property_free (sctx, GSASL_PASSWORD);
00128
00129         /* The following will invoke a GSASL_PASSWORD callback. */
00130         key = gsasl_property_get (sctx, GSASL_PASSWORD);
00131         if (!key)
00132           {
00133             free (passprep);
00134             return GSASL_NO_PASSWORD;
00135           }
00136
00137         /* Unassigned code points are not permitted. */
00138         res = gsasl_saslprep (key, 0, &normkey, NULL);
00139         if (res != GSASL_OK)
00140           {
00141             free (passprep);
00142             return res;
00143           }
00144
00145         if (strcmp (normkey, passprep) == 0)
00146           res = GSASL_OK;
00147         else
00148           res = GSASL_AUTHENTICATION_ERROR;
00149
00150         free (normkey);
00151       }
00152     free (passprep);
00153
00154     return res;
00155 }
```

## 5.127 saml20/server.c File Reference

```
#include <config.h>
#include "saml20.h"
#include <string.h>
#include <stdlib.h>
#include "mechtools.h"
```

### Data Structures

- struct saml20_server_state

### Functions

- int _gsasl_saml20_server_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
- int _gsasl_saml20_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_saml20_server_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)

### 5.127.1 Function Documentation

#### 5.127.1.1 _gsasl_saml20_server_finish()

```
void _gsasl_saml20_server_finish (
            Gsasl_session *sctx _GL_UNUSED,
            void * mech_data )
```

Definition at line 140 of file saml20/server.c.

#### 5.127.1.2 _gsasl_saml20_server_start()

```
int _gsasl_saml20_server_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 42 of file saml20/server.c.

#### 5.127.1.3 _gsasl_saml20_server_step()

```
int _gsasl_saml20_server_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 56 of file saml20/server.c.

## 5.128 saml20/server.c

Go to the documentation of this file.

```
00001 /* server.c --- SAML20 mechanism, server side.
00002  * Copyright (C) 2010-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "saml20.h"
00026
00027 /* Get strdup, strlen. */
00028 #include <string.h>
00029
00030 /* Get free. */
00031 #include <stdlib.h>
00032
00033 /* Get _gsasl_parse_gs2_header. */
00034 #include "mechtools.h"
00035
00036 struct saml20_server_state
00037 {
00038   int step;
00039 };
00040
00041 int
00042 _gsasl_saml20_server_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data)
00043 {
00044   struct saml20_server_state *state;
00045
00046   state = (struct saml20_server_state *) calloc (1, sizeof (*state));
```

```
00047   if (state == NULL)
00048     return GSASL_MALLOC_ERROR;
00049
00050   *mech_data = state;
00051
00052   return GSASL_OK;
00053 }
00054
00055 int
00056 _gsasl_saml20_server_step (Gsasl_session *sctx,
00057                            void *mech_data,
00058                            const char *input, size_t input_len,
00059                            char **output, size_t *output_len)
00060 {
00061   struct saml20_server_state *state = mech_data;
00062   int res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00063
00064   *output_len = 0;
00065   *output = NULL;
00066
00067   switch (state->step)
00068     {
00069     case 0:
00070       {
00071         const char *p;
00072         char *authzid;
00073         size_t headerlen;
00074
00075         if (input_len == 0)
00076           return GSASL_NEEDS_MORE;
00077
00078         res = _gsasl_parse_gs2_header (input, input_len,
00079                                        &authzid, &headerlen);
00080         if (res != GSASL_OK)
00081           return res;
00082
00083         if (authzid)
00084           {
00085             res = gsasl_property_set (sctx, GSASL_AUTHZID, authzid);
00086             free (authzid);
00087             if (res != GSASL_OK)
00088               return res;
00089           }
00090
00091         input += headerlen;
00092         input_len -= headerlen;
00093
00094         res = gsasl_property_set_raw (sctx, GSASL_SAML20_IDP_IDENTIFIER,
00095                                       input, input_len);
00096         if (res != GSASL_OK)
00097           return res;
00098
00099         p = gsasl_property_get (sctx, GSASL_SAML20_REDIRECT_URL);
00100         if (!p || !*p)
00101           return GSASL_NO_SAML20_REDIRECT_URL;
00102
00103         *output_len = strlen (p);
00104         *output = malloc (*output_len);
00105         if (!*output)
00106           return GSASL_MALLOC_ERROR;
00107
00108         memcpy (*output, p, *output_len);
00109
00110         res = GSASL_NEEDS_MORE;
00111         state->step++;
00112         break;
00113       }
00114
00115     case 1:
00116       {
00117         if (!(input_len == 1 && *input == '='))
00118           return GSASL_MECHANISM_PARSE_ERROR;
00119
00120         res = gsasl_callback (NULL, sctx, GSASL_VALIDATE_SAML20);
00121         if (res != GSASL_OK)
00122           return res;
00123
00124         *output = NULL;
00125         *output_len = 0;
00126
00127         res = GSASL_OK;
00128         state->step++;
00129         break;
00130       }
00131
00132     default:
00133       break;
```

```
00134     }
00135
00136   return res;
00137 }
00138
00139 void
00140 _gsasl_saml20_server_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)
00141 {
00142   struct saml20_server_state *state = mech_data;
00143
00144   if (!state)
00145     return;
00146
00147   free (state);
00148 }
```

## 5.129 scram/server.c File Reference

```
#include <config.h>
#include "scram.h"
#include <stdlib.h>
#include <limits.h>
#include <string.h>
#include "minmax.h"
#include "tokens.h"
#include "parser.h"
#include "printer.h"
#include "gc.h"
#include "memxor.h"
#include "tools.h"
#include "mechtools.h"
```

### Data Structures

- struct scram_server_state

### Macros

- #define DEFAULT_SALT_BYTES 12
- #define SNONCE_ENTROPY_BYTES 18

### Functions

- int _gsasl_scram_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_scram_server_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)

### 5.129.1 Macro Definition Documentation

#### 5.129.1.1 DEFAULT_SALT_BYTES

```
#define DEFAULT_SALT_BYTES 12
```

Definition at line 47 of file scram/server.c.

#### 5.129.1.2 SNONCE_ENTROPY_BYTES

```
#define SNONCE_ENTROPY_BYTES 18
```

Definition at line 48 of file scram/server.c.

### 5.129.2 Function Documentation

#### 5.129.2.1 _gsasl_scram_server_finish()

```
void _gsasl_scram_server_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 580 of file scram/server.c.

#### 5.129.2.2 _gsasl_scram_server_step()

```
int _gsasl_scram_server_step (
             Gsasl_session * sctx,
             void * mech_data,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

Definition at line 168 of file scram/server.c.

## 5.130 scram/server.c

Go to the documentation of this file.

```
00001 /* server.c --- SASL CRAM-MD5 server side functions.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "scram.h"
00026
00027 /* Get malloc, free, strtoul. */
00028 #include <stdlib.h>
00029
00030 /* Get ULONG_MAX. */
```

```
00031 #include <limits.h>
00032
00033 /* Get memcpy, strdup, strlen. */
00034 #include <string.h>
00035
00036 /* Get MAX. */
00037 #include "minmax.h"
00038
00039 #include "tokens.h"
00040 #include "parser.h"
00041 #include "printer.h"
00042 #include "gc.h"
00043 #include "memxor.h"
00044 #include "tools.h"
00045 #include "mechtools.h"
00046
00047 #define DEFAULT_SALT_BYTES 12
00048 #define SNONCE_ENTROPY_BYTES 18
00049
00050 struct scram_server_state
00051 {
00052   bool plus;
00053   Gsasl_hash hash;
00054   int step;
00055   char *cbind;
00056   char *gs2header;             /* copy of client first gs2-header */
00057   char *cfmb_str;               /* copy of client first message bare */
00058   char *sf_str;                   /* copy of server first message */
00059   char *snonce;
00060   char *clientproof;
00061   char storedkey[GSASL_HASH_MAX_SIZE];
00062   char serverkey[GSASL_HASH_MAX_SIZE];
00063   char *authmessage;
00064   char *cb;
00065   size_t cblen;
00066   struct scram_client_first cf;
00067   struct scram_server_first sf;
00068   struct scram_client_final cl;
00069   struct scram_server_final sl;
00070 };
00071
00072 static int
00073 scram_start (Gsasl_session *sctx _GL_UNUSED, void **mech_data,
00074              bool plus, Gsasl_hash hash)
00075 {
00076   struct scram_server_state *state;
00077   char buf[MAX (SNONCE_ENTROPY_BYTES, DEFAULT_SALT_BYTES)];
00078   int rc;
00079
00080   state = (struct scram_server_state *) calloc (1, sizeof (*state));
00081   if (state == NULL)
00082     return GSASL_MALLOC_ERROR;
00083
00084   state->plus = plus;
00085   state->hash = hash;
00086
00087   rc = gsasl_nonce (buf, SNONCE_ENTROPY_BYTES);
00088   if (rc != GSASL_OK)
00089     goto end;
00090
00091   rc = gsasl_base64_to (buf, SNONCE_ENTROPY_BYTES, &state->snonce, NULL);
00092   if (rc != GSASL_OK)
00093     goto end;
00094
00095   rc = gsasl_nonce (buf, DEFAULT_SALT_BYTES);
00096   if (rc != GSASL_OK)
00097     goto end;
00098
00099   rc = gsasl_base64_to (buf, DEFAULT_SALT_BYTES, &state->sf.salt, NULL);
00100   if (rc != GSASL_OK)
00101     goto end;
00102
00103   *mech_data = state;
00104
00105   return GSASL_OK;
00106
00107 end:
00108   free (state->sf.salt);
00109   free (state->snonce);
00110   free (state);
00111   return rc;
00112 }
00113
00114 #ifdef USE_SCRAM_SHA1
00115 int
00116 _gsasl_scram_sha1_server_start (Gsasl_session *sctx, void **mech_data)
00117 {
```

```
00118   return scram_start (sctx, mech_data, false, GSASL_HASH_SHA1);
00119 }
00120
00121 int
00122 _gsasl_scram_sha1_plus_server_start (Gsasl_session *sctx, void **mech_data)
00123 {
00124   return scram_start (sctx, mech_data, true, GSASL_HASH_SHA1);
00125 }
00126 #endif
00127
00128 #ifdef USE_SCRAM_SHA256
00129 int
00130 _gsasl_scram_sha256_server_start (Gsasl_session *sctx, void **mech_data)
00131 {
00132   return scram_start (sctx, mech_data, false, GSASL_HASH_SHA256);
00133 }
00134
00135 int
00136 _gsasl_scram_sha256_plus_server_start (Gsasl_session *sctx, void **mech_data)
00137 {
00138   return scram_start (sctx, mech_data, true, GSASL_HASH_SHA256);
00139 }
00140 #endif
00141
00142 static int
00143 extract_serverkey (struct scram_server_state *state,
00144                    const char *b64, char *buf)
00145 {
00146   char *bin;
00147   size_t binlen;
00148   int rc;
00149
00150   rc = gsasl_base64_from (b64, strlen (b64), &bin, &binlen);
00151   if (rc != GSASL_OK)
00152     return rc;
00153
00154   if (binlen != gsasl_hash_length (state->hash))
00155     {
00156       free (bin);
00157       return GSASL_AUTHENTICATION_ERROR;
00158     }
00159
00160   memcpy (buf, bin, binlen);
00161
00162   free (bin);
00163
00164   return GSASL_OK;
00165 }
00166
00167 int
00168 _gsasl_scram_server_step (Gsasl_session *sctx,
00169                           void *mech_data,
00170                           const char *input,
00171                           size_t input_len, char **output, size_t *output_len)
00172 {
00173   struct scram_server_state *state = mech_data;
00174   int res = GSASL_MECHANISM_CALLED_TOO_MANY_TIMES;
00175   int rc;
00176
00177   *output = NULL;
00178   *output_len = 0;
00179
00180   switch (state->step)
00181     {
00182     case 0:
00183       {
00184         if (input_len == 0)
00185           return GSASL_NEEDS_MORE;
00186
00187         if (scram_parse_client_first (input, input_len, &state->cf) < 0)
00188           return GSASL_MECHANISM_PARSE_ERROR;
00189
00190         if (state->plus)
00191           {
00192             const char *p;
00193
00194             /* In PLUS server mode, we require use of channel bindings. */
00195             if (state->cf.cbflag != 'p' || state->cf.cbname == NULL)
00196               return GSASL_AUTHENTICATION_ERROR;
00197
00198             if (strcmp (state->cf.cbname, "tls-exporter") == 0)
00199               {
00200                 p = gsasl_property_get (sctx, GSASL_CB_TLS_EXPORTER);
00201                 if (!p)
00202                   return GSASL_NO_CB_TLS_EXPORTER;
00203               }
00204             else if (strcmp (state->cf.cbname, "tls-unique") == 0)
```

```
00205               {
00206                 p = gsasl_property_get (sctx, GSASL_CB_TLS_UNIQUE);
00207                 if (!p)
00208                   return GSASL_NO_CB_TLS_UNIQUE;
00209               }
00210             else
00211               return GSASL_AUTHENTICATION_ERROR;
00212
00213             rc = gsasl_base64_from (p, strlen (p), &state->cb, &state->cblen);
00214             if (rc != GSASL_OK)
00215               return rc;
00216           }
00217         else if (state->cf.cbflag == 'y')
00218           {
00219             const char *p = gsasl_property_get (sctx, GSASL_CB_TLS_EXPORTER);
00220             /* In non-PLUS mode we reject a client 'y' cbflag since we
00221                support channel bindings UNLESS we actually don't have
00222                any channel bindings (application told to libgsasl that
00223                it doesn't want PLUS). */
00224             if (!p)
00225               p = gsasl_property_get (sctx, GSASL_CB_TLS_UNIQUE);
00226             if (p != NULL)
00227               return GSASL_AUTHENTICATION_ERROR;
00228           }
00229
00230         /* Check that username doesn't fail SASLprep. */
00231         {
00232           char *tmp;
00233           rc = gsasl_saslprep (state->cf.username, GSASL_ALLOW_UNASSIGNED,
00234                                &tmp, NULL);
00235           if (rc != GSASL_OK || *tmp == '\0')
00236             return GSASL_AUTHENTICATION_ERROR;
00237           gsasl_free (tmp);
00238         }
00239
00240         {
00241           const char *p;
00242
00243           /* Save "gs2-header" and "message-bare" for next step. */
00244           p = memchr (input, ',', input_len);
00245           if (!p)
00246             return GSASL_AUTHENTICATION_ERROR;
00247           p++;
00248           p = memchr (p, ',', input_len - (p - input));
00249           if (!p)
00250             return GSASL_AUTHENTICATION_ERROR;
00251           p++;
00252
00253           state->gs2header = malloc (p - input + 1);
00254           if (!state->gs2header)
00255             return GSASL_MALLOC_ERROR;
00256           memcpy (state->gs2header, input, p - input);
00257           state->gs2header[p - input] = '\0';
00258
00259           state->cfmb_str = malloc (input_len - (p - input) + 1);
00260           if (!state->cfmb_str)
00261             return GSASL_MALLOC_ERROR;
00262           memcpy (state->cfmb_str, p, input_len - (p - input));
00263           state->cfmb_str[input_len - (p - input)] = '\0';
00264         }
00265
00266         /* Create new nonce. */
00267         {
00268           size_t cnlen = strlen (state->cf.client_nonce);
00269           size_t snlen = strlen (state->snonce);
00270
00271           state->sf.nonce = malloc (cnlen + snlen + 1);
00272           if (!state->sf.nonce)
00273             return GSASL_MALLOC_ERROR;
00274
00275           memcpy (state->sf.nonce, state->cf.client_nonce, cnlen);
00276           memcpy (state->sf.nonce + cnlen, state->snonce, snlen);
00277           state->sf.nonce[cnlen + snlen] = '\0';
00278         }
00279
00280         rc = gsasl_property_set (sctx, GSASL_AUTHID, state->cf.username);
00281         if (rc != GSASL_OK)
00282           return rc;
00283         rc = gsasl_property_set (sctx, GSASL_AUTHZID, state->cf.authzid);
00284         if (rc != GSASL_OK)
00285           return rc;
00286
00287         {
00288           const char *p = gsasl_property_get (sctx, GSASL_SCRAM_ITER);
00289
00290           if (p)
00291             state->sf.iter = strtoul (p, NULL, 10);
```

```
00292           if (!p || state->sf.iter == 0 || state->sf.iter == ULONG_MAX)
00293             state->sf.iter = 4096;
00294
00295           /* Save salt/iter as properties, so that client callback can
00296              access them. */
00297           {
00298             char *str = NULL;
00299             int n;
00300             n = asprintf (&str, "%zu", state->sf.iter);
00301             if (n < 0 || str == NULL)
00302               return GSASL_MALLOC_ERROR;
00303             rc = gsasl_property_set (sctx, GSASL_SCRAM_ITER, str);
00304             free (str);
00305             if (rc != GSASL_OK)
00306               return rc;
00307           }
00308         }
00309
00310         {
00311           const char *p = gsasl_property_get (sctx, GSASL_SCRAM_SALT);
00312           if (p)
00313             {
00314               free (state->sf.salt);
00315               state->sf.salt = strdup (p);
00316             }
00317           else
00318             {
00319               rc =
00320                 gsasl_property_set (sctx, GSASL_SCRAM_SALT, state->sf.salt);
00321               if (rc != GSASL_OK)
00322                 return rc;
00323             }
00324         }
00325
00326         rc = scram_print_server_first (&state->sf, &state->sf_str);
00327         if (rc != 0)
00328           return GSASL_MALLOC_ERROR;
00329
00330         *output = strdup (state->sf_str);
00331         if (!*output)
00332           return GSASL_MALLOC_ERROR;
00333         *output_len = strlen (*output);
00334
00335         state->step++;
00336         return GSASL_NEEDS_MORE;
00337         break;
00338       }
00339
00340     case 1:
00341       {
00342         if (scram_parse_client_final (input, input_len, &state->cl) < 0)
00343           return GSASL_MECHANISM_PARSE_ERROR;
00344
00345         if (strcmp (state->cl.nonce, state->sf.nonce) != 0)
00346           return GSASL_AUTHENTICATION_ERROR;
00347
00348         /* Base64 decode the c= field and check that it matches
00349            client-first.  Also check channel binding data. */
00350         {
00351           size_t len;
00352
00353           free (state->cbind);
00354           rc = gsasl_base64_from (state->cl.cbind, strlen (state->cl.cbind),
00355                                   &state->cbind, &len);
00356           if (rc != 0)
00357             return rc;
00358
00359           if (state->cf.cbflag == 'p')
00360             {
00361               if (len < strlen (state->gs2header))
00362                 return GSASL_AUTHENTICATION_ERROR;
00363
00364               if (memcmp (state->cbind, state->gs2header,
00365                           strlen (state->gs2header)) != 0)
00366                 return GSASL_AUTHENTICATION_ERROR;
00367
00368               if (len - strlen (state->gs2header) != state->cblen)
00369                 return GSASL_AUTHENTICATION_ERROR;
00370
00371               if (memcmp (state->cbind + strlen (state->gs2header),
00372                           state->cb, state->cblen) != 0)
00373                 return GSASL_AUTHENTICATION_ERROR;
00374             }
00375           else
00376             {
00377               if (len != strlen (state->gs2header))
00378                 return GSASL_AUTHENTICATION_ERROR;
```

```
00379
00380               if (memcmp (state->cbind, state->gs2header, len) != 0)
00381                 return GSASL_AUTHENTICATION_ERROR;
00382             }
00383         }
00384
00385         /* Base64 decode client proof and check that length matches
00386            hash size. */
00387         {
00388           size_t len;
00389
00390           free (state->clientproof);
00391           rc = gsasl_base64_from (state->cl.proof, strlen (state->cl.proof),
00392                                   &state->clientproof, &len);
00393           if (rc != 0)
00394             return rc;
00395           if (gsasl_hash_length (state->hash) != len)
00396             return GSASL_MECHANISM_PARSE_ERROR;
00397         }
00398         {
00399           const char *p, *q;
00400
00401           /* Get StoredKey and ServerKey */
00402           if ((p = gsasl_property_get (sctx, GSASL_SCRAM_SERVERKEY))
00403               && (q = gsasl_property_get (sctx, GSASL_SCRAM_STOREDKEY)))
00404             {
00405               rc = extract_serverkey (state, p, state->serverkey);
00406               if (rc != GSASL_OK)
00407                 return rc;
00408               rc = extract_serverkey (state, q, state->storedkey);
00409               if (rc != GSASL_OK)
00410                 return rc;
00411             }
00412           else if ((q = gsasl_property_get (sctx,
00413                                             GSASL_SCRAM_SALTED_PASSWORD))
00414                    || (p = gsasl_property_get (sctx, GSASL_PASSWORD)))
00415             {
00416               char clientkey[GSASL_HASH_MAX_SIZE];
00417               char *b64str;
00418
00419               if (q)
00420                 {
00421                   char *binsaltedpassword;
00422                   size_t outlen;
00423
00424                   rc = gsasl_hex_from (q, &binsaltedpassword, &outlen);
00425                   if (rc != GSASL_OK)
00426                     return rc;
00427
00428                   if (outlen != gsasl_hash_length (state->hash))
00429                     {
00430                       gsasl_free (binsaltedpassword);
00431                       return GSASL_AUTHENTICATION_ERROR;
00432                     }
00433
00434                   rc = gsasl_scram_secrets_from_salted_password
00435                     (state->hash, binsaltedpassword,
00436                      clientkey, state->serverkey, state->storedkey);
00437                   gsasl_free (binsaltedpassword);
00438                   if (rc != GSASL_OK)
00439                     return rc;
00440                 }
00441               else
00442                 {
00443                   char *salt;
00444                   size_t saltlen;
00445                   char saltedpassword[GSASL_HASH_MAX_SIZE];
00446
00447                   rc = gsasl_base64_from (state->sf.salt,
00448                                           strlen (state->sf.salt),
00449                                           &salt, &saltlen);
00450                   if (rc != GSASL_OK)
00451                     return rc;
00452
00453                   rc = gsasl_scram_secrets_from_password (state->hash,
00454                                                           p,
00455                                                           state->sf.iter,
00456                                                           salt, saltlen,
00457                                                           saltedpassword,
00458                                                           clientkey,
00459                                                           state->serverkey,
00460                                                           state->storedkey);
00461                   gsasl_free (salt);
00462                   if (rc != GSASL_OK)
00463                     return rc;
00464
00465                   rc = set_saltedpassword (sctx, state->hash, saltedpassword);
```

```
00466                 if (rc != GSASL_OK)
00467                   return rc;
00468               }
00469
00470             rc = gsasl_base64_to (state->serverkey,
00471                                   gsasl_hash_length (state->hash),
00472                                   &b64str, NULL);
00473             if (rc != GSASL_OK)
00474               return rc;
00475             rc = gsasl_property_set (sctx, GSASL_SCRAM_SERVERKEY, b64str);
00476             free (b64str);
00477             if (rc != GSASL_OK)
00478               return rc;
00479
00480             rc = gsasl_base64_to (state->storedkey,
00481                                   gsasl_hash_length (state->hash),
00482                                   &b64str, NULL);
00483             if (rc != 0)
00484               return rc;
00485             rc = gsasl_property_set (sctx, GSASL_SCRAM_STOREDKEY, b64str);
00486             free (b64str);
00487             if (rc != GSASL_OK)
00488               return rc;
00489           }
00490         else
00491           return GSASL_NO_PASSWORD;
00492
00493         /* Compute AuthMessage */
00494         {
00495           size_t len;
00496           int n;
00497
00498           /* Get client-final-message-without-proof. */
00499           p = memmem (input, input_len, ",p=", 3);
00500           if (!p)
00501             return GSASL_MECHANISM_PARSE_ERROR;
00502           len = p - input;
00503
00504           n = asprintf (&state->authmessage, "%s,%.*s,%.*s",
00505                         state->cfmb_str,
00506                         (int) strlen (state->sf_str), state->sf_str,
00507                         (int) len, input);
00508           if (n <= 0 || !state->authmessage)
00509             return GSASL_MALLOC_ERROR;
00510         }
00511
00512         /* Check client proof. */
00513         {
00514           char clientsignature[GSASL_HASH_MAX_SIZE];
00515           char maybe_storedkey[GSASL_HASH_MAX_SIZE];
00516
00517           /* ClientSignature := HMAC(StoredKey, AuthMessage) */
00518           rc = _gsasl_hmac (state->hash,
00519                             state->storedkey,
00520                             gsasl_hash_length (state->hash),
00521                             state->authmessage, strlen (state->authmessage),
00522                             clientsignature);
00523           if (rc != 0)
00524             return rc;
00525
00526           /* ClientKey := ClientProof XOR ClientSignature */
00527           memxor (clientsignature, state->clientproof,
00528                   gsasl_hash_length (state->hash));
00529
00530           rc = _gsasl_hash (state->hash, clientsignature,
00531                             gsasl_hash_length (state->hash),
00532                             maybe_storedkey);
00533           if (rc != 0)
00534             return rc;
00535
00536           rc = memcmp (state->storedkey, maybe_storedkey,
00537                        gsasl_hash_length (state->hash));
00538           if (rc != 0)
00539             return GSASL_AUTHENTICATION_ERROR;
00540         }
00541
00542         /* Generate server verifier. */
00543         {
00544           char serversignature[GSASL_HASH_MAX_SIZE];
00545
00546           /* ServerSignature := HMAC(ServerKey, AuthMessage) */
00547           rc = _gsasl_hmac (state->hash, state->serverkey,
00548                             gsasl_hash_length (state->hash),
00549                             state->authmessage,
00550                             strlen (state->authmessage), serversignature);
00551           if (rc != 0)
00552             return rc;
```

```
00553
00554             rc = gsasl_base64_to (serversignature,
00555                                   gsasl_hash_length (state->hash),
00556                                   &state->sl.verifier, NULL);
00557             if (rc != 0)
00558               return rc;
00559           }
00560         }
00561
00562         rc = scram_print_server_final (&state->sl, output);
00563         if (rc != 0)
00564           return GSASL_MALLOC_ERROR;
00565         *output_len = strlen (*output);
00566
00567         state->step++;
00568         return GSASL_OK;
00569         break;
00570       }
00571
00572     default:
00573       break;
00574     }
00575
00576   return res;
00577 }
00578
00579 void
00580 _gsasl_scram_server_finish (Gsasl_session *sctx _GL_UNUSED, void *mech_data)
00581 {
00582   struct scram_server_state *state = mech_data;
00583
00584   if (!state)
00585     return;
00586
00587   free (state->cbind);
00588   free (state->gs2header);
00589   free (state->cfmb_str);
00590   free (state->sf_str);
00591   free (state->snonce);
00592   free (state->clientproof);
00593   free (state->authmessage);
00594   free (state->cb);
00595   scram_free_client_first (&state->cf);
00596   scram_free_server_first (&state->sf);
00597   scram_free_client_final (&state->cl);
00598   scram_free_server_final (&state->sl);
00599
00600   free (state);
00601 }
```

## 5.131 securid/server.c File Reference

```
#include <config.h>
#include "securid.h"
#include <stdlib.h>
#include <string.h>
```

### Macros

- #define PASSCODE "passcode"
- #define PIN "pin"

### Functions

- int _gsasl_securid_server_step (Gsasl_session *sctx, void *mech_data _GL_UNUSED, const char *input, size_t input_len, char **output, size_t *output_len)

### 5.131.1 Macro Definition Documentation

#### 5.131.1.1 PASSCODE

```
#define PASSCODE "passcode"
```

Definition at line 33 of file securid/server.c.

#### 5.131.1.2 PIN

```
#define PIN "pin"
```

Definition at line 34 of file securid/server.c.

### 5.131.2 Function Documentation

#### 5.131.2.1 _gsasl_securid_server_step()

```
int _gsasl_securid_server_step (
              Gsasl_session * sctx,
              void *mech_data _GL_UNUSED,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

Definition at line 37 of file securid/server.c.

## 5.132 securid/server.c

Go to the documentation of this file.

```
00001 /* server.c --- SASL mechanism SECURID from RFC 2808, server side.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "securid.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
```

```
00030 /* Get memchr, strdup, strlen. */
00031 #include <string.h>
00032
00033 #define PASSCODE "passcode"
00034 #define PIN "pin"
00035
00036 int
00037 _gsasl_securid_server_step (Gsasl_session *sctx,
00038                             void *mech_data _GL_UNUSED,
00039                             const char *input, size_t input_len,
00040                             char **output, size_t *output_len)
00041 {
00042   const char *authorization_id = NULL;
00043   const char *authentication_id = NULL;
00044   const char *passcode = NULL;
00045   const char *suggestedpin;
00046   const char *pin = NULL;
00047   int res;
00048   size_t len;
00049
00050   if (input_len == 0)
00051     {
00052       *output_len = 0;
00053       *output = NULL;
00054       return GSASL_NEEDS_MORE;
00055     }
00056
00057   authorization_id = input;
00058   authentication_id = memchr (input, '\0', input_len - 1);
00059   if (authentication_id)
00060     {
00061       authentication_id++;
00062       passcode = memchr (authentication_id, '\0',
00063                          input_len - strlen (authorization_id) - 1 - 1);
00064       if (passcode)
00065         {
00066           passcode++;
00067           pin = memchr (passcode, '\0', input_len -
00068                         strlen (authorization_id) - 1 -
00069                         strlen (authentication_id) - 1 - 1);
00070           if (pin)
00071             {
00072               pin++;
00073               if (pin && !*pin)
00074                 pin = NULL;
00075             }
00076         }
00077     }
00078
00079   if (passcode == NULL)
00080     return GSASL_MECHANISM_PARSE_ERROR;
00081
00082   res = gsasl_property_set (sctx, GSASL_AUTHID, authentication_id);
00083   if (res != GSASL_OK)
00084     return res;
00085
00086   res = gsasl_property_set (sctx, GSASL_AUTHZID, authorization_id);
00087   if (res != GSASL_OK)
00088     return res;
00089   res = gsasl_property_set (sctx, GSASL_PASSCODE, passcode);
00090   if (res != GSASL_OK)
00091     return res;
00092
00093   if (pin)
00094     res = gsasl_property_set (sctx, GSASL_PIN, pin);
00095   else
00096     res = gsasl_property_set (sctx, GSASL_PIN, NULL);
00097   if (res != GSASL_OK)
00098     return res;
00099
00100   res = gsasl_callback (NULL, sctx, GSASL_VALIDATE_SECURID);
00101   switch (res)
00102     {
00103     case GSASL_SECURID_SERVER_NEED_ADDITIONAL_PASSCODE:
00104       *output = strdup (PASSCODE);
00105       if (!*output)
00106         return GSASL_MALLOC_ERROR;
00107       *output_len = strlen (PASSCODE);
00108       res = GSASL_NEEDS_MORE;
00109       break;
00110
00111     case GSASL_SECURID_SERVER_NEED_NEW_PIN:
00112       suggestedpin = gsasl_property_get (sctx, GSASL_SUGGESTED_PIN);
00113       if (suggestedpin)
00114         len = strlen (suggestedpin);
00115       else
00116         len = 0;
```

```
00117       *output_len = strlen (PIN) + len;
00118       *output = malloc (*output_len);
00119       if (!*output)
00120         return GSASL_MALLOC_ERROR;
00121       memcpy (*output, PIN, strlen (PIN));
00122       if (suggestedpin)
00123         memcpy (*output + strlen (PIN), suggestedpin, len);
00124       res = GSASL_NEEDS_MORE;
00125       break;
00126
00127     default:
00128       *output_len = 0;
00129       *output = NULL;
00130       break;
00131     }
00132
00133   return res;
00134 }
```

## 5.133 digest-md5/parser.c File Reference

```
#include <config.h>
#include "parser.h"
#include <stdlib.h>
#include <string.h>
#include "validate.h"
```

### Macros

- #define DEFAULT_CHARSET "utf-8"
- #define DEFAULT_ALGORITHM "md5-sess"

### Enumerations

- enum {
  CHALLENGE_REALM = 0 , CHALLENGE_NONCE , CHALLENGE_QOP , CHALLENGE_STALE ,
  CHALLENGE_MAXBUF , CHALLENGE_CHARSET , CHALLENGE_ALGORITHM , CHALLENGE_CIPHER
  }
- enum { QOP_AUTH = 0 , QOP_AUTH_INT , QOP_AUTH_CONF }
- enum {
  CIPHER_DES = 0 , CIPHER_3DES , CIPHER_RC4 , CIPHER_RC4_40 ,
  CIPHER_RC4_56 , CIPHER_AES_CBC }
- enum {
  RESPONSE_USERNAME = 0 , RESPONSE_REALM , RESPONSE_NONCE , RESPONSE_CNONCE ,
  RESPONSE_NC , RESPONSE_QOP , RESPONSE_DIGEST_URI , RESPONSE_RESPONSE ,
  RESPONSE_MAXBUF , RESPONSE_CHARSET , RESPONSE_CIPHER , RESPONSE_AUTHZID }
- enum { RESPONSEAUTH_RSPAUTH = 0 }

### Functions

- int digest_md5_parse_challenge (const char *challenge, size_t len, digest_md5_challenge *out)
- int digest_md5_parse_response (const char *response, size_t len, digest_md5_response *out)
- int digest_md5_parse_finish (const char *finish, size_t len, digest_md5_finish *out)

## 5.133.1 Macro Definition Documentation

### 5.133.1.1 DEFAULT_ALGORITHM

```
#define DEFAULT_ALGORITHM "md5-sess"
```

Definition at line 37 of file digest-md5/parser.c.

### 5.133.1.2 DEFAULT_CHARSET

```
#define DEFAULT_CHARSET "utf-8"
```

Definition at line 36 of file digest-md5/parser.c.

## 5.133.2 Enumeration Type Documentation

### 5.133.2.1 anonymous enum

```
anonymous enum
```

**Enumerator**

| | |
|---|---|
| CHALLENGE_REALM | |
| CHALLENGE_NONCE | |
| CHALLENGE_QOP | |
| CHALLENGE_STALE | |
| CHALLENGE_MAXBUF | |
| CHALLENGE_CHARSET | |
| CHALLENGE_ALGORITHM | |
| CHALLENGE_CIPHER | |

Definition at line 39 of file digest-md5/parser.c.

### 5.133.2.2 anonymous enum

```
anonymous enum
```

**Enumerator**

| | |
|---|---|
| QOP_AUTH | |
| QOP_AUTH_INT | |
| QOP_AUTH_CONF | |

Definition at line 66 of file digest-md5/parser.c.

#### 5.133.2.3 anonymous enum

```
anonymous enum
```

**Enumerator**

| | |
|---|---|
| CIPHER_DES | |
| CIPHER_3DES | |
| CIPHER_RC4 | |
| CIPHER_RC4_40 | |
| CIPHER_RC4_56 | |
| CIPHER_AES_CBC | |

Definition at line 86 of file digest-md5/parser.c.

#### 5.133.2.4 anonymous enum

```
anonymous enum
```

**Enumerator**

| | |
|---|---|
| RESPONSE_USERNAME | |
| RESPONSE_REALM | |
| RESPONSE_NONCE | |
| RESPONSE_CNONCE | |
| RESPONSE_NC | |
| RESPONSE_QOP | |
| RESPONSE_DIGEST_URI | |
| RESPONSE_RESPONSE | |
| RESPONSE_MAXBUF | |
| RESPONSE_CHARSET | |
| RESPONSE_CIPHER | |
| RESPONSE_AUTHZID | |

Definition at line 312 of file digest-md5/parser.c.

#### 5.133.2.5 anonymous enum

```
anonymous enum
```

**Enumerator**

| | |
|---|---|
| RESPONSEAUTH_RSPAUTH | |

Definition at line 518 of file digest-md5/parser.c.

### 5.133.3 Function Documentation

#### 5.133.3.1 digest_md5_parse_challenge()

```
int digest_md5_parse_challenge (
            const char * challenge,
            size_t len,
            digest_md5_challenge * out )
```

Definition at line 566 of file digest-md5/parser.c.

#### 5.133.3.2 digest_md5_parse_finish()

```
int digest_md5_parse_finish (
            const char * finish,
            size_t len,
            digest_md5_finish * out )
```

Definition at line 600 of file digest-md5/parser.c.

#### 5.133.3.3 digest_md5_parse_response()

```
int digest_md5_parse_response (
            const char * response,
            size_t len,
            digest_md5_response * out )
```

Definition at line 583 of file digest-md5/parser.c.

## 5.134 digest-md5/parser.c

Go to the documentation of this file.

```
00001 /* parser.c --- DIGEST-MD5 parser.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get prototypes. */
00025 #include "parser.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
```

```
00030 /* Get memcpy, strlen. */
00031 #include <string.h>
00032
00033 /* Get validator. */
00034 #include "validate.h"
00035
00036 #define DEFAULT_CHARSET "utf-8"
00037 #define DEFAULT_ALGORITHM "md5-sess"
00038
00039 enum
00040 {
00041   /* the order must match the following struct */
00042   CHALLENGE_REALM = 0,
00043   CHALLENGE_NONCE,
00044   CHALLENGE_QOP,
00045   CHALLENGE_STALE,
00046   CHALLENGE_MAXBUF,
00047   CHALLENGE_CHARSET,
00048   CHALLENGE_ALGORITHM,
00049   CHALLENGE_CIPHER
00050 };
00051
00052 static const char *const digest_challenge_opts[] = {
00053   /* the order must match the previous enum */
00054   "realm",
00055   "nonce",
00056   "qop",
00057   "stale",
00058   "maxbuf",
00059   "charset",
00060   "algorithm",
00061   "cipher",
00062   NULL
00063 };
00064
00065 /* qop-value         = "auth" | "auth-int" | "auth-conf" | qop-token */
00066 enum
00067 {
00068   /* the order must match the following struct */
00069   QOP_AUTH = 0,
00070   QOP_AUTH_INT,
00071   QOP_AUTH_CONF
00072 };
00073
00074 static const char *const qop_opts[] = {
00075   /* the order must match the previous enum */
00076   "auth",
00077   "auth-int",
00078   "auth-conf",
00079   NULL
00080 };
00081
00082 /* cipher-value      = "3des" | "des" | "rc4-40" | "rc4" |
00083  *                     "rc4-56" | "aes-cbc" | cipher-token
00084  *                     ;; "des" and "3des" ciphers are obsolete.
00085  */
00086 enum
00087 {
00088   /* the order must match the following struct */
00089   CIPHER_DES = 0,
00090   CIPHER_3DES,
00091   CIPHER_RC4,
00092   CIPHER_RC4_40,
00093   CIPHER_RC4_56,
00094   CIPHER_AES_CBC
00095 };
00096
00097 static const char *const cipher_opts[] = {
00098   /* the order must match the previous enum */
00099   "des",
00100   "3des",
00101   "rc4",
00102   "rc4-40",
00103   "rc4-56",
00104   "aes-cbc",
00105   NULL
00106 };
00107
00108 static int
00109 parse_challenge (char *challenge, digest_md5_challenge *out)
00110 {
00111   int done_algorithm = 0;
00112   int disable_qop_auth_conf = 0;
00113   char *value;
00114
00115   memset (out, 0, sizeof (*out));
00116
```

```
00117   /* The size of a digest-challenge MUST be less than 2048 bytes. */
00118   if (strlen (challenge) >= 2048)
00119     return -1;
00120
00121   while (*challenge != '\0')
00122     switch (digest_md5_getsubopt (&challenge, digest_challenge_opts, &value))
00123       {
00124       case CHALLENGE_REALM:
00125         {
00126           char **tmp;
00127           out->nrealms++;
00128           tmp = realloc (out->realms, out->nrealms * sizeof (*out->realms));
00129           if (!tmp)
00130             return -1;
00131           out->realms = tmp;
00132           out->realms[out->nrealms - 1] = strdup (value);
00133           if (!out->realms[out->nrealms - 1])
00134             return -1;
00135         }
00136         break;
00137
00138       case CHALLENGE_NONCE:
00139         /* This directive is required and MUST appear exactly once; if
00140            not present, or if multiple instances are present, the
00141            client should abort the authentication exchange. */
00142         if (out->nonce)
00143           return -1;
00144         out->nonce = strdup (value);
00145         if (!out->nonce)
00146           return -1;
00147         break;
00148
00149       case CHALLENGE_QOP:
00150         /* «What if this directive is present multiple times? Error,
00151            or take the union of all values?» */
00152         if (out->qops)
00153           return -1;
00154         {
00155           char *subsubopts;
00156           char *val;
00157
00158           subsubopts = value;
00159           while (*subsubopts != '\0')
00160             switch (digest_md5_getsubopt (&subsubopts, qop_opts, &val))
00161               {
00162               case QOP_AUTH:
00163                 out->qops |= DIGEST_MD5_QOP_AUTH;
00164                 break;
00165
00166               case QOP_AUTH_INT:
00167                 out->qops |= DIGEST_MD5_QOP_AUTH_INT;
00168                 break;
00169
00170               case QOP_AUTH_CONF:
00171                 out->qops |= DIGEST_MD5_QOP_AUTH_CONF;
00172                 break;
00173
00174               default:
00175                 /* The client MUST ignore unrecognized options */
00176                 break;
00177               }
00178         }
00179         /* if the client recognizes no cipher, it MUST behave as if
00180            "auth-conf" qop option wasn't provided by the server. */
00181         if (disable_qop_auth_conf)
00182           out->qops &= ~DIGEST_MD5_QOP_AUTH_CONF;
00183         /* if the client recognizes no option, it MUST abort the
00184            authentication exchange. */
00185         if (!out->qops)
00186           return -1;
00187         break;
00188
00189       case CHALLENGE_STALE:
00190         /* This directive may appear at most once; if multiple
00191            instances are present, the client MUST abort the
00192            authentication exchange. */
00193         if (out->stale)
00194           return -1;
00195         out->stale = 1;
00196         break;
00197
00198       case CHALLENGE_MAXBUF:
00199         /* This directive may appear at most once; if multiple
00200            instances are present, or the value is out of range the
00201            client MUST abort the authentication exchange. */
00202         if (out->servermaxbuf)
00203           return -1;
```

```
00204         out->servermaxbuf = strtoul (value, NULL, 10);
00205         /* FIXME: error handling. */
00206         /* The value MUST be bigger than 16 (32 for Confidentiality
00207            protection with the "aes-cbc" cipher) and smaller or equal
00208            to 16777215 (i.e. 2**24-1). */
00209         if (out->servermaxbuf <= 16 || out->servermaxbuf > 16777215)
00210           return -1;
00211         break;
00212
00213       case CHALLENGE_CHARSET:
00214         /* This directive may appear at most once; if multiple
00215            instances are present, the client MUST abort the
00216            authentication exchange. */
00217         if (out->utf8)
00218           return -1;
00219         if (strcmp (DEFAULT_CHARSET, value) != 0)
00220           return -1;
00221         out->utf8 = 1;
00222         break;
00223
00224       case CHALLENGE_ALGORITHM:
00225         /* This directive is required and MUST appear exactly once; if
00226            not present, or if multiple instances are present, the
00227            client SHOULD abort the authentication exchange. */
00228         if (done_algorithm)
00229           return -1;
00230         if (strcmp (DEFAULT_ALGORITHM, value) != 0)
00231           return -1;
00232         done_algorithm = 1;
00233         break;
00234
00235
00236       case CHALLENGE_CIPHER:
00237         /* This directive must be present exactly once if "auth-conf"
00238            is offered in the "qop-options" directive */
00239         if (out->ciphers)
00240           return -1;
00241         {
00242           char *subsubopts;
00243           char *val;
00244
00245           subsubopts = value;
00246           while (*subsubopts != '\0')
00247             switch (digest_md5_getsubopt (&subsubopts, cipher_opts, &val))
00248               {
00249               case CIPHER_DES:
00250                 out->ciphers |= DIGEST_MD5_CIPHER_DES;
00251                 break;
00252
00253               case CIPHER_3DES:
00254                 out->ciphers |= DIGEST_MD5_CIPHER_3DES;
00255                 break;
00256
00257               case CIPHER_RC4:
00258                 out->ciphers |= DIGEST_MD5_CIPHER_RC4;
00259                 break;
00260
00261               case CIPHER_RC4_40:
00262                 out->ciphers |= DIGEST_MD5_CIPHER_RC4_40;
00263                 break;
00264
00265               case CIPHER_RC4_56:
00266                 out->ciphers |= DIGEST_MD5_CIPHER_RC4_56;
00267                 break;
00268
00269               case CIPHER_AES_CBC:
00270                 out->ciphers |= DIGEST_MD5_CIPHER_AES_CBC;
00271                 break;
00272
00273               default:
00274                 /* The client MUST ignore unrecognized ciphers */
00275                 break;
00276               }
00277         }
00278         /* if the client recognizes no cipher, it MUST behave as if
00279            "auth-conf" qop option wasn't provided by the server. */
00280         if (!out->ciphers)
00281           {
00282             disable_qop_auth_conf = 1;
00283             if (out->qops)
00284               {
00285                 /* if the client recognizes no option, it MUST abort the
00286                    authentication exchange. */
00287                 out->qops &= ~DIGEST_MD5_QOP_AUTH_CONF;
00288                 if (!out->qops)
00289                   return -1;
00290               }
```

```
00291           }
00292         break;
00293
00294       default:
00295         /* The client MUST ignore any unrecognized directives. */
00296         break;
00297       }
00298
00299   /* This directive is required and MUST appear exactly once; if
00300      not present, or if multiple instances are present, the
00301      client SHOULD abort the authentication exchange. */
00302   if (!done_algorithm)
00303     return -1;
00304
00305   /* Validate that we have the mandatory fields. */
00306   if (digest_md5_validate_challenge (out) != 0)
00307     return -1;
00308
00309   return 0;
00310 }
00311
00312 enum
00313 {
00314   /* the order must match the following struct */
00315   RESPONSE_USERNAME = 0,
00316   RESPONSE_REALM,
00317   RESPONSE_NONCE,
00318   RESPONSE_CNONCE,
00319   RESPONSE_NC,
00320   RESPONSE_QOP,
00321   RESPONSE_DIGEST_URI,
00322   RESPONSE_RESPONSE,
00323   RESPONSE_MAXBUF,
00324   RESPONSE_CHARSET,
00325   RESPONSE_CIPHER,
00326   RESPONSE_AUTHZID
00327 };
00328
00329 static const char *const digest_response_opts[] = {
00330   /* the order must match the previous enum */
00331   "username",
00332   "realm",
00333   "nonce",
00334   "cnonce",
00335   "nc",
00336   "qop",
00337   "digest-uri",
00338   "response",
00339   "maxbuf",
00340   "charset",
00341   "cipher",
00342   "authzid",
00343   NULL
00344 };
00345
00346 static int
00347 parse_response (char *response, digest_md5_response *out)
00348 {
00349   char *value;
00350
00351   memset (out, 0, sizeof (*out));
00352
00353   /* The size of a digest-response MUST be less than 4096 bytes. */
00354   if (strlen (response) >= 4096)
00355     return -1;
00356
00357   while (*response != '\0')
00358     switch (digest_md5_getsubopt (&response, digest_response_opts, &value))
00359       {
00360       case RESPONSE_USERNAME:
00361         /* This directive is required and MUST be present exactly
00362            once; otherwise, authentication fails. */
00363         if (out->username)
00364           return -1;
00365         out->username = strdup (value);
00366         if (!out->username)
00367           return -1;
00368         break;
00369
00370       case RESPONSE_REALM:
00371         /* This directive is required if the server provided any
00372            realms in the "digest-challenge", in which case it may
00373            appear exactly once and its value SHOULD be one of those
00374            realms. */
00375         if (out->realm)
00376           return -1;
00377         out->realm = strdup (value);
```

```
00378         if (!out->realm)
00379           return -1;
00380         break;
00381
00382       case RESPONSE_NONCE:
00383         /* This directive is required and MUST be present exactly
00384            once; otherwise, authentication fails. */
00385         if (out->nonce)
00386           return -1;
00387         out->nonce = strdup (value);
00388         if (!out->nonce)
00389           return -1;
00390         break;
00391
00392       case RESPONSE_CNONCE:
00393         /* This directive is required and MUST be present exactly once;
00394            otherwise, authentication fails. */
00395         if (out->cnonce)
00396           return -1;
00397         out->cnonce = strdup (value);
00398         if (!out->cnonce)
00399           return -1;
00400         break;
00401
00402       case RESPONSE_NC:
00403         /* This directive is required and MUST be present exactly
00404            once; otherwise, authentication fails. */
00405         if (out->nc)
00406           return -1;
00407         /* nc-value = 8LHEX */
00408         if (strlen (value) != 8)
00409           return -1;
00410         out->nc = strtoul (value, NULL, 16);
00411         /* FIXME: error handling. */
00412         break;
00413
00414       case RESPONSE_QOP:
00415         /* If present, it may appear exactly once and its value MUST
00416            be one of the alternatives in qop-options.  */
00417         if (out->qop)
00418           return -1;
00419         if (strcmp (value, "auth") == 0)
00420           out->qop = DIGEST_MD5_QOP_AUTH;
00421         else if (strcmp (value, "auth-int") == 0)
00422           out->qop = DIGEST_MD5_QOP_AUTH_INT;
00423         else if (strcmp (value, "auth-conf") == 0)
00424           out->qop = DIGEST_MD5_QOP_AUTH_CONF;
00425         else
00426           return -1;
00427         break;
00428
00429       case RESPONSE_DIGEST_URI:
00430         /* This directive is required and MUST be present exactly
00431            once; if multiple instances are present, the client MUST
00432            abort the authentication exchange. */
00433         if (out->digesturi)
00434           return -1;
00435         /* FIXME: sub-parse. */
00436         out->digesturi = strdup (value);
00437         if (!out->digesturi)
00438           return -1;
00439         break;
00440
00441       case RESPONSE_RESPONSE:
00442         /* This directive is required and MUST be present exactly
00443            once; otherwise, authentication fails. */
00444         if (*out->response)
00445           return -1;
00446         /* A string of 32 hex digits */
00447         if (strlen (value) != DIGEST_MD5_RESPONSE_LENGTH)
00448           return -1;
00449         strcpy (out->response, value);
00450         break;
00451
00452       case RESPONSE_MAXBUF:
00453         /* This directive may appear at most once; if multiple
00454            instances are present, the server MUST abort the
00455            authentication exchange. */
00456         if (out->clientmaxbuf)
00457           return -1;
00458         out->clientmaxbuf = strtoul (value, NULL, 10);
00459         /* FIXME: error handling. */
00460         /* If the value is less or equal to 16 («32 for aes-cbc») or
00461            bigger than 16777215 (i.e. 2**24-1), the server MUST abort
00462            the authentication exchange. */
00463         if (out->clientmaxbuf <= 16 || out->clientmaxbuf > 16777215)
00464           return -1;
```

```
00465         break;
00466
00467       case RESPONSE_CHARSET:
00468         if (strcmp (DEFAULT_CHARSET, value) != 0)
00469           return -1;
00470         out->utf8 = 1;
00471         break;
00472
00473       case RESPONSE_CIPHER:
00474         if (out->cipher)
00475           return -1;
00476         if (strcmp (value, "3des") == 0)
00477           out->cipher = DIGEST_MD5_CIPHER_3DES;
00478         else if (strcmp (value, "des") == 0)
00479           out->cipher = DIGEST_MD5_CIPHER_DES;
00480         else if (strcmp (value, "rc4-40") == 0)
00481           out->cipher = DIGEST_MD5_CIPHER_RC4_40;
00482         else if (strcmp (value, "rc4") == 0)
00483           out->cipher = DIGEST_MD5_CIPHER_RC4;
00484         else if (strcmp (value, "rc4-56") == 0)
00485           out->cipher = DIGEST_MD5_CIPHER_RC4_56;
00486         else if (strcmp (value, "aes-cbc") == 0)
00487           out->cipher = DIGEST_MD5_CIPHER_AES_CBC;
00488         else
00489           return -1;
00490         break;
00491
00492       case RESPONSE_AUTHZID:
00493         /* This directive may appear at most once; if multiple
00494            instances are present, the server MUST abort the
00495            authentication exchange.  «FIXME NOT IN DRAFT» */
00496         if (out->authzid)
00497           return -1;
00498         /*  The authzid MUST NOT be an empty string. */
00499         if (*value == '\0')
00500           return -1;
00501         out->authzid = strdup (value);
00502         if (!out->authzid)
00503           return -1;
00504         break;
00505
00506       default:
00507         /* The client MUST ignore any unrecognized directives. */
00508         break;
00509       }
00510
00511   /* Validate that we have the mandatory fields. */
00512   if (digest_md5_validate_response (out) != 0)
00513     return -1;
00514
00515   return 0;
00516 }
00517
00518 enum
00519 {
00520   /* the order must match the following struct */
00521   RESPONSEAUTH_RSPAUTH = 0
00522 };
00523
00524 static const char *const digest_responseauth_opts[] = {
00525   /* the order must match the previous enum */
00526   "rspauth",
00527   NULL
00528 };
00529
00530 static int
00531 parse_finish (char *finish, digest_md5_finish *out)
00532 {
00533   char *value;
00534
00535   memset (out, 0, sizeof (*out));
00536
00537   /* The size of a response-auth MUST be less than 2048 bytes. */
00538   if (strlen (finish) >= 2048)
00539     return -1;
00540
00541   while (*finish != '\0')
00542     switch (digest_md5_getsubopt (&finish, digest_responseauth_opts, &value))
00543       {
00544       case RESPONSEAUTH_RSPAUTH:
00545         if (*out->rspauth)
00546           return -1;
00547         /* A string of 32 hex digits */
00548         if (strlen (value) != DIGEST_MD5_RESPONSE_LENGTH)
00549           return -1;
00550         strcpy (out->rspauth, value);
00551         break;
```

```
00552
00553       default:
00554         /* The client MUST ignore any unrecognized directives. */
00555         break;
00556       }
00557
00558   /* Validate that we have the mandatory fields. */
00559   if (digest_md5_validate_finish (out) != 0)
00560     return -1;
00561
00562   return 0;
00563 }
00564
00565 int
00566 digest_md5_parse_challenge (const char *challenge, size_t len,
00567                             digest_md5_challenge *out)
00568 {
00569   char *subopts = len ? strndup (challenge, len) : strdup (challenge);
00570   int rc;
00571
00572   if (!subopts)
00573     return -1;
00574
00575   rc = parse_challenge (subopts, out);
00576
00577   free (subopts);
00578
00579   return rc;
00580 }
00581
00582 int
00583 digest_md5_parse_response (const char *response, size_t len,
00584                            digest_md5_response *out)
00585 {
00586   char *subopts = len ? strndup (response, len) : strdup (response);
00587   int rc;
00588
00589   if (!subopts)
00590     return -1;
00591
00592   rc = parse_response (subopts, out);
00593
00594   free (subopts);
00595
00596   return rc;
00597 }
00598
00599 int
00600 digest_md5_parse_finish (const char *finish, size_t len,
00601                          digest_md5_finish *out)
00602 {
00603   char *subopts = len ? strndup (finish, len) : strdup (finish);
00604   int rc;
00605
00606   if (!subopts)
00607     return -1;
00608
00609   rc = parse_finish (subopts, out);
00610
00611   free (subopts);
00612
00613   return rc;
00614 }
```

## 5.135 scram/parser.c File Reference

```
#include <config.h>
#include "parser.h"
#include <stdlib.h>
#include <string.h>
#include "validate.h"
#include "c-ctype.h"
```

### Functions

- int scram_parse_client_first (const char *str, size_t len, struct scram_client_first *cf)

- int scram_parse_server_first (const char *str, size_t len, struct scram_server_first *sf)
- int scram_parse_client_final (const char *str, size_t len, struct scram_client_final *cl)
- int scram_parse_server_final (const char *str, size_t len, struct scram_server_final *sl)

## 5.135.1 Function Documentation

### 5.135.1.1 scram_parse_client_final()

```
int scram_parse_client_final (
            const char * str,
            size_t len,
            struct scram_client_final * cl )
```

Definition at line 328 of file scram/parser.c.

### 5.135.1.2 scram_parse_client_first()

```
int scram_parse_client_first (
            const char * str,
            size_t len,
            struct scram_client_first * cf )
```

Definition at line 75 of file scram/parser.c.

### 5.135.1.3 scram_parse_server_final()

```
int scram_parse_server_final (
            const char * str,
            size_t len,
            struct scram_server_final * sl )
```

Definition at line 458 of file scram/parser.c.

### 5.135.1.4 scram_parse_server_first()

```
int scram_parse_server_first (
            const char * str,
            size_t len,
            struct scram_server_first * sf )
```

Definition at line 217 of file scram/parser.c.

## 5.136 scram/parser.c

Go to the documentation of this file.

```
00001 /* parser.c --- SCRAM parser.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get prototypes. */
00025 #include "parser.h"
00026
00027 /* Get malloc, free. */
00028 #include <stdlib.h>
00029
00030 /* Get memcpy, strlen. */
00031 #include <string.h>
00032
00033 /* Get validator. */
00034 #include "validate.h"
00035
00036 /* Get c_isalpha. */
00037 #include "c-ctype.h"
00038
00039 static char *
00040 unescape (const char *str, size_t len)
00041 {
00042   char *out = malloc (len + 1);
00043   char *p = out;
00044
00045   if (!out)
00046     return NULL;
00047
00048   while (len > 0 && *str)
00049     {
00050       if (len >= 3 && str[0] == '=' && str[1] == '2' && str[2] == 'C')
00051         {
00052           *p++ = ',';
00053           str += 3;
00054           len -= 3;
00055         }
00056       else if (len >= 3 && str[0] == '=' && str[1] == '3' && str[2] == 'D')
00057         {
00058           *p++ = '=';
00059           str += 3;
00060           len -= 3;
00061         }
00062       else
00063         {
00064           *p++ = *str;
00065           str++;
00066           len--;
00067         }
00068     }
00069   *p = '\0';
00070
00071   return out;
00072 }
00073
00074 int
00075 scram_parse_client_first (const char *str, size_t len,
00076                           struct scram_client_first *cf)
00077 {
00078   scram_free_client_first (cf);
00079
00080   /* Minimum client first string is 'n,,n=a,r=b'. */
00081   if (strnlen (str, len) < 10)
00082     return -1;
```

```
00083
00084   if (len == 0 || (*str != 'n' && *str != 'y' && *str != 'p'))
00085     return -1;
00086   cf->cbflag = *str;
00087   str++, len--;
00088
00089   if (cf->cbflag == 'p')
00090     {
00091       const char *p;
00092
00093       if (len == 0 || *str != '=')
00094         return -1;
00095       str++, len--;
00096
00097       p = memchr (str, ',', len);
00098       if (!p)
00099         return -1;
00100       cf->cbname = malloc (p - str + 1);
00101       if (!cf->cbname)
00102         return -1;
00103       memcpy (cf->cbname, str, p - str);
00104       cf->cbname[p - str] = '\0';
00105       len -= (p - str);
00106       str += (p - str);
00107     }
00108
00109   if (len == 0 || *str != ',')
00110     return -1;
00111   str++, len--;
00112
00113   if (len == 0)
00114     return -1;
00115   if (*str == 'a')
00116     {
00117       const char *p;
00118       size_t l;
00119
00120       str++, len--;
00121       if (len == 0 || *str != '=')
00122         return -1;
00123       str++, len--;
00124
00125       p = memchr (str, ',', len);
00126       if (!p)
00127         return -1;
00128
00129       l = p - str;
00130       if (len < l)
00131         return -1;
00132
00133       cf->authzid = unescape (str, l);
00134       if (!cf->authzid)
00135         return -1;
00136
00137       str = p;
00138       len -= l;
00139     }
00140
00141   if (len == 0 || *str != ',')
00142     return -1;
00143   str++, len--;
00144
00145   if (len == 0 || *str != 'n')
00146     return -1;
00147   str++, len--;
00148
00149   if (len == 0 || *str != '=')
00150     return -1;
00151   str++, len--;
00152
00153   {
00154     const char *p;
00155     size_t l;
00156
00157     p = memchr (str, ',', len);
00158     if (!p)
00159       return -1;
00160
00161     l = p - str;
00162     if (len < l)
00163       return -1;
00164
00165     cf->username = unescape (str, l);
00166     if (!cf->username)
00167       return -1;
00168
00169     str = p;
```

```
00170     len -= l;
00171   }
00172
00173   if (len == 0 || *str != ',')
00174     return -1;
00175   str++, len--;
00176
00177   if (len == 0 || *str != 'r')
00178     return -1;
00179   str++, len--;
00180
00181   if (len == 0 || *str != '=')
00182     return -1;
00183   str++, len--;
00184
00185   {
00186     const char *p;
00187     size_t l;
00188
00189     p = memchr (str, ',', len);
00190     if (!p)
00191       p = str + len;
00192
00193     l = p - str;
00194     if (len < l)
00195       return -1;
00196
00197     cf->client_nonce = malloc (l + 1);
00198     if (!cf->client_nonce)
00199       return -1;
00200
00201     memcpy (cf->client_nonce, str, l);
00202     cf->client_nonce[l] = '\0';
00203
00204     str = p;
00205     len -= l;
00206   }
00207
00208   /* FIXME check that any extension fields follow valid syntax. */
00209
00210   if (!scram_valid_client_first (cf))
00211     return -1;
00212
00213   return 0;
00214 }
00215
00216 int
00217 scram_parse_server_first (const char *str, size_t len,
00218                           struct scram_server_first *sf)
00219 {
00220   scram_free_server_first (sf);
00221
00222   /* Minimum server first string is 'r=ab,s=biws,i=1'. */
00223   if (strnlen (str, len) < 15)
00224     return -1;
00225
00226   if (len == 0 || *str != 'r')
00227     return -1;
00228   str++, len--;
00229
00230   if (len == 0 || *str != '=')
00231     return -1;
00232   str++, len--;
00233
00234   {
00235     const char *p;
00236     size_t l;
00237
00238     p = memchr (str, ',', len);
00239     if (!p)
00240       return -1;
00241
00242     l = p - str;
00243     if (len < l)
00244       return -1;
00245
00246     sf->nonce = malloc (l + 1);
00247     if (!sf->nonce)
00248       return -1;
00249
00250     memcpy (sf->nonce, str, l);
00251     sf->nonce[l] = '\0';
00252
00253     str = p;
00254     len -= l;
00255   }
00256
```

```
00257   if (len == 0 || *str != ',')
00258     return -1;
00259   str++, len--;
00260
00261   if (len == 0 || *str != 's')
00262     return -1;
00263   str++, len--;
00264
00265   if (len == 0 || *str != '=')
00266     return -1;
00267   str++, len--;
00268
00269   {
00270     const char *p;
00271     size_t l;
00272
00273     p = memchr (str, ',', len);
00274     if (!p)
00275       return -1;
00276
00277     l = p - str;
00278     if (len < l)
00279       return -1;
00280
00281     sf->salt = malloc (l + 1);
00282     if (!sf->salt)
00283       return -1;
00284
00285     memcpy (sf->salt, str, l);
00286     sf->salt[l] = '\0';
00287
00288     str = p;
00289     len -= l;
00290   }
00291
00292   if (len == 0 || *str != ',')
00293     return -1;
00294   str++, len--;
00295
00296   if (len == 0 || *str != 'i')
00297     return -1;
00298   str++, len--;
00299
00300   if (len == 0 || *str != '=')
00301     return -1;
00302   str++, len--;
00303
00304   sf->iter = 0;
00305   for (; len > 0 && *str >= '0' && *str <= '9'; str++, len--)
00306     {
00307       size_t last_iter = sf->iter;
00308
00309       sf->iter = sf->iter * 10 + (*str - '0');
00310
00311       /* Protect against wrap arounds. */
00312       if (sf->iter < last_iter)
00313         return -1;
00314     }
00315
00316   if (len > 0 && *str != ',')
00317     return -1;
00318
00319   /* FIXME check that any extension fields follow valid syntax. */
00320
00321   if (!scram_valid_server_first (sf))
00322     return -1;
00323
00324   return 0;
00325 }
00326
00327 int
00328 scram_parse_client_final (const char *str, size_t len,
00329                           struct scram_client_final *cl)
00330 {
00331   scram_free_client_final (cl);
00332
00333   /* Minimum client final string is 'c=biws,r=ab,p=ab==' . */
00334   if (strnlen (str, len) < 18)
00335     return -1;
00336
00337   if (len == 0 || *str != 'c')
00338     return -1;
00339   str++, len--;
00340
00341   if (len == 0 || *str != '=')
00342     return -1;
00343   str++, len--;
```

```
00344
00345   {
00346     const char *p;
00347     size_t l;
00348
00349     p = memchr (str, ',', len);
00350     if (!p)
00351       return -1;
00352
00353     l = p - str;
00354     if (len < l)
00355       return -1;
00356
00357     cl->cbind = malloc (l + 1);
00358     if (!cl->cbind)
00359       return -1;
00360
00361     memcpy (cl->cbind, str, l);
00362     cl->cbind[l] = '\0';
00363
00364     str = p;
00365     len -= l;
00366   }
00367
00368   if (len == 0 || *str != ',')
00369     return -1;
00370   str++, len--;
00371
00372   if (len == 0 || *str != 'r')
00373     return -1;
00374   str++, len--;
00375
00376   if (len == 0 || *str != '=')
00377     return -1;
00378   str++, len--;
00379
00380   {
00381     const char *p;
00382     size_t l;
00383
00384     p = memchr (str, ',', len);
00385     if (!p)
00386       return -1;
00387
00388     l = p - str;
00389     if (len < l)
00390       return -1;
00391
00392     cl->nonce = malloc (l + 1);
00393     if (!cl->nonce)
00394       return -1;
00395
00396     memcpy (cl->nonce, str, l);
00397     cl->nonce[l] = '\0';
00398
00399     str = p;
00400     len -= l;
00401   }
00402
00403   if (len == 0 || *str != ',')
00404     return -1;
00405   str++, len--;
00406
00407   /* Ignore extensions. */
00408   while (len > 0 && c_isalpha (*str) && *str != 'p')
00409     {
00410       const char *p;
00411       size_t l;
00412
00413       str++, len--;
00414
00415       if (len == 0 || *str != '=')
00416         return -1;
00417       str++, len--;
00418
00419       p = memchr (str, ',', len);
00420       if (!p)
00421         return -1;
00422       p++;
00423
00424       l = p - str;
00425       if (len < l)
00426         return -1;
00427
00428       str = p;
00429       len -= l;
00430     }
```

```
00431
00432   if (len == 0 || *str != 'p')
00433     return -1;
00434   str++, len--;
00435
00436   if (len == 0 || *str != '=')
00437     return -1;
00438   str++, len--;
00439
00440   /* Sanity check proof. */
00441   if (memchr (str, '\0', len))
00442     return -1;
00443
00444   cl->proof = malloc (len + 1);
00445   if (!cl->proof)
00446     return -1;
00447
00448   memcpy (cl->proof, str, len);
00449   cl->proof[len] = '\0';
00450
00451   if (!scram_valid_client_final (cl))
00452     return -1;
00453
00454   return 0;
00455 }
00456
00457 int
00458 scram_parse_server_final (const char *str, size_t len,
00459                           struct scram_server_final *sl)
00460 {
00461   scram_free_server_final (sl);
00462
00463   /* Minimum client final string is 'v=ab=='. */
00464   if (strnlen (str, len) < 6)
00465     return -1;
00466
00467   if (len == 0 || *str != 'v')
00468     return -1;
00469   str++, len--;
00470
00471   if (len == 0 || *str != '=')
00472     return -1;
00473   str++, len--;
00474
00475   /* Sanity check proof. */
00476   if (memchr (str, '\0', len))
00477     return -1;
00478
00479   sl->verifier = malloc (len + 1);
00480   if (!sl->verifier)
00481     return -1;
00482
00483   memcpy (sl->verifier, str, len);
00484   sl->verifier[len] = '\0';
00485
00486   if (!scram_valid_server_final (sl))
00487     return -1;
00488
00489   return 0;
00490 }
```

## 5.137 digest-md5/parser.h File Reference

```
#include "tokens.h"
```

### Functions

- int digest_md5_getsubopt (char **optionp, const char *const *tokens, char **valuep)
- int digest_md5_parse_challenge (const char *challenge, size_t len, digest_md5_challenge *out)
- int digest_md5_parse_response (const char *response, size_t len, digest_md5_response *out)
- int digest_md5_parse_finish (const char *finish, size_t len, digest_md5_finish *out)

### 5.137.1 Function Documentation

#### 5.137.1.1 digest_md5_getsubopt()

```
int digest_md5_getsubopt (
             char ** optionp,
             const char *const * tokens,
             char ** valuep )  [extern]
```

Definition at line 43 of file getsubopt.c.

#### 5.137.1.2 digest_md5_parse_challenge()

```
int digest_md5_parse_challenge (
             const char * challenge,
             size_t len,
             digest_md5_challenge * out )  [extern]
```

Definition at line 566 of file digest-md5/parser.c.

#### 5.137.1.3 digest_md5_parse_finish()

```
int digest_md5_parse_finish (
             const char * finish,
             size_t len,
             digest_md5_finish * out )  [extern]
```

Definition at line 600 of file digest-md5/parser.c.

#### 5.137.1.4 digest_md5_parse_response()

```
int digest_md5_parse_response (
             const char * response,
             size_t len,
             digest_md5_response * out )  [extern]
```

Definition at line 583 of file digest-md5/parser.c.

## 5.138 digest-md5/parser.h

Go to the documentation of this file.

```
00001 /* parser.h --- DIGEST-MD5 parser.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef DIGEST_MD5_PARSER_H
00023 # define DIGEST_MD5_PARSER_H
00024
00025 /* Get token types. */
00026 # include "tokens.h"
00027
00028 extern int digest_md5_getsubopt (char **optionp,
00029                                  const char *const *tokens, char **valuep);
00030
00031 extern int digest_md5_parse_challenge (const char *challenge, size_t len,
00032                                        digest_md5_challenge * out);
00033
00034 extern int digest_md5_parse_response (const char *response, size_t len,
00035                                       digest_md5_response * out);
00036
00037 extern int digest_md5_parse_finish (const char *finish, size_t len,
00038                                     digest_md5_finish * out);
00039
00040 #endif /* DIGEST_MD5_PARSER_H */
```

## 5.139 scram/parser.h File Reference

```
#include "tokens.h"
```

### Functions

- int scram_parse_client_first (const char *str, size_t len, struct scram_client_first *cf)
- int scram_parse_server_first (const char *str, size_t len, struct scram_server_first *cf)
- int scram_parse_client_final (const char *str, size_t len, struct scram_client_final *cl)
- int scram_parse_server_final (const char *str, size_t len, struct scram_server_final *sl)

### 5.139.1 Function Documentation

#### 5.139.1.1 scram_parse_client_final()

```
int scram_parse_client_final (
            const char * str,
            size_t len,
            struct scram_client_final * cl )  [extern]
```

Definition at line 328 of file scram/parser.c.

#### 5.139.1.2 scram_parse_client_first()

```
int scram_parse_client_first (
            const char * str,
            size_t len,
            struct scram_client_first * cf )  [extern]
```

Definition at line 75 of file scram/parser.c.

#### 5.139.1.3 scram_parse_server_final()

```
int scram_parse_server_final (
            const char * str,
            size_t len,
            struct scram_server_final * sl )  [extern]
```

Definition at line 458 of file scram/parser.c.

#### 5.139.1.4 scram_parse_server_first()

```
int scram_parse_server_first (
            const char * str,
            size_t len,
            struct scram_server_first * cf )  [extern]
```

Definition at line 217 of file scram/parser.c.

## 5.140 scram/parser.h

Go to the documentation of this file.

```
00001 /* parser.h --- SCRAM parser.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef SCRAM_PARSER_H
00023 # define SCRAM_PARSER_H
00024
00025 /* Get token types. */
00026 # include "tokens.h"
00027
00028 extern int scram_parse_client_first (const char *str, size_t len,
00029                                      struct scram_client_first *cf);
00030
00031 extern int scram_parse_server_first (const char *str, size_t len,
00032                                      struct scram_server_first *cf);
00033
00034 extern int scram_parse_client_final (const char *str, size_t len,
00035                                      struct scram_client_final *cl);
00036
00037 extern int scram_parse_server_final (const char *str, size_t len,
00038                                      struct scram_server_final *sl);
00039
00040 #endif /* SCRAM_PARSER_H */
```

# 5.141 digest-md5/printer.c File Reference

```
#include <config.h>
#include "printer.h"
#include <stdlib.h>
#include <stdio.h>
#include "validate.h"
```

### Functions

- char * digest_md5_print_challenge (digest_md5_challenge *c)
- char * digest_md5_print_response (digest_md5_response *r)
- char * digest_md5_print_finish (digest_md5_finish *finish)

## 5.141.1 Function Documentation

### 5.141.1.1 digest_md5_print_challenge()

```
char * digest_md5_print_challenge (
            digest_md5_challenge * c )
```

Definition at line 71 of file digest-md5/printer.c.

### 5.141.1.2 digest_md5_print_finish()

```
char * digest_md5_print_finish (
            digest_md5_finish * finish )
```

Definition at line 384 of file digest-md5/printer.c.

### 5.141.1.3 digest_md5_print_response()

```
char * digest_md5_print_response (
            digest_md5_response * r )
```

Definition at line 240 of file digest-md5/printer.c.

## 5.142 digest-md5/printer.c

Go to the documentation of this file.

```
00001 /* printer.h --- Convert DIGEST-MD5 token structures into strings.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get prototypes. */
00025 #include "printer.h"
00026
00027 /* Get free. */
00028 #include <stdlib.h>
00029
00030 /* Get asprintf. */
00031 #include <stdio.h>
00032
00033 /* Get token validator. */
00034 #include "validate.h"
00035
00036 /* Append a key/value pair to a comma'd string list.  Additionally enclose
00037    the value in quotes if requested. */
00038 static int
00039 comma_append (char **dst, const char *key, const char *value, int quotes)
00040 {
00041   char *tmp;
00042   int result;
00043
00044   if (*dst)
00045     if (value)
00046       if (quotes)
00047         result = asprintf (&tmp, "%s, %s=\"%s\"", *dst, key, value);
00048       else
00049         result = asprintf (&tmp, "%s, %s=%s", *dst, key, value);
00050     else
00051       result = asprintf (&tmp, "%s, %s", *dst, key);
00052   else if (value)
00053     if (quotes)
00054       result = asprintf (&tmp, "%s=\"%s\"", key, value);
00055     else
00056       result = asprintf (&tmp, "%s=%s", key, value);
00057   else
00058     result = asprintf (&tmp, "%s", key);
00059
00060   if (result < 0)
00061     return result;
00062
00063   free (*dst);
00064
00065   *dst = tmp;
00066
00067   return result;
00068 }
00069
00070 char *
00071 digest_md5_print_challenge (digest_md5_challenge *c)
00072 {
00073   char *out = NULL;
00074   size_t i;
00075
00076   /* Below we assume the mandatory fields are present, verify that
00077      first to avoid crashes. */
00078   if (digest_md5_validate_challenge (c) != 0)
00079     return NULL;
00080
00081   for (i = 0; i < c->nrealms; i++)
00082     {
```

```
00083       if (comma_append (&out, "realm", c->realms[i], 1) < 0)
00084         {
00085           free (out);
00086           return NULL;
00087         }
00088     }
00089
00090   if (c->nonce)
00091     if (comma_append (&out, "nonce", c->nonce, 1) < 0)
00092       {
00093         free (out);
00094         return NULL;
00095       }
00096
00097   if (c->qops)
00098     {
00099       char *tmp = NULL;
00100
00101       if (c->qops & DIGEST_MD5_QOP_AUTH)
00102         if (comma_append (&tmp, "auth", NULL, 0) < 0)
00103           {
00104             free (tmp);
00105             free (out);
00106             return NULL;
00107           }
00108
00109       if (c->qops & DIGEST_MD5_QOP_AUTH_INT)
00110         if (comma_append (&tmp, "auth-int", NULL, 0) < 0)
00111           {
00112             free (tmp);
00113             free (out);
00114             return NULL;
00115           }
00116
00117       if (c->qops & DIGEST_MD5_QOP_AUTH_CONF)
00118         if (comma_append (&tmp, "auth-conf", NULL, 0) < 0)
00119           {
00120             free (tmp);
00121             free (out);
00122             return NULL;
00123           }
00124
00125       if (comma_append (&out, "qop", tmp, 1) < 0)
00126         {
00127           free (tmp);
00128           free (out);
00129           return NULL;
00130         }
00131
00132       free (tmp);
00133     }
00134
00135   if (c->stale)
00136     if (comma_append (&out, "stale", "true", 0) < 0)
00137       {
00138         free (out);
00139         return NULL;
00140       }
00141
00142   if (c->servermaxbuf)
00143     {
00144       char *tmp;
00145
00146       if (asprintf (&tmp, "%lu", c->servermaxbuf) < 0)
00147         {
00148           free (out);
00149           return NULL;
00150         }
00151
00152       if (comma_append (&out, "maxbuf", tmp, 0) < 0)
00153         {
00154           free (out);
00155           return NULL;
00156         }
00157
00158       free (tmp);
00159     }
00160
00161   if (c->utf8)
00162     if (comma_append (&out, "charset", "utf-8", 0) < 0)
00163       {
00164         free (out);
00165         return NULL;
00166       }
00167
00168   if (comma_append (&out, "algorithm", "md5-sess", 0) < 0)
00169     {
```

```
00170       free (out);
00171       return NULL;
00172     }
00173
00174   if (c->ciphers)
00175     {
00176       char *tmp = NULL;
00177
00178       if (c->ciphers & DIGEST_MD5_CIPHER_3DES)
00179         if (comma_append (&tmp, "3des", NULL, 0) < 0)
00180           {
00181             free (tmp);
00182             free (out);
00183             return NULL;
00184           }
00185
00186       if (c->ciphers & DIGEST_MD5_CIPHER_DES)
00187         if (comma_append (&tmp, "des", NULL, 0) < 0)
00188           {
00189             free (tmp);
00190             free (out);
00191             return NULL;
00192           }
00193
00194       if (c->ciphers & DIGEST_MD5_CIPHER_RC4_40)
00195         if (comma_append (&tmp, "rc4-40", NULL, 0) < 0)
00196           {
00197             free (tmp);
00198             free (out);
00199             return NULL;
00200           }
00201
00202       if (c->ciphers & DIGEST_MD5_CIPHER_RC4)
00203         if (comma_append (&tmp, "rc4", NULL, 0) < 0)
00204           {
00205             free (tmp);
00206             free (out);
00207             return NULL;
00208           }
00209
00210       if (c->ciphers & DIGEST_MD5_CIPHER_RC4_56)
00211         if (comma_append (&tmp, "rc4-56", NULL, 0) < 0)
00212           {
00213             free (tmp);
00214             free (out);
00215             return NULL;
00216           }
00217
00218       if (c->ciphers & DIGEST_MD5_CIPHER_AES_CBC)
00219         if (comma_append (&tmp, "aes-cbc", NULL, 0) < 0)
00220           {
00221             free (tmp);
00222             free (out);
00223             return NULL;
00224           }
00225
00226       if (comma_append (&out, "cipher", tmp, 1) < 0)
00227         {
00228           free (tmp);
00229           free (out);
00230           return NULL;
00231         }
00232
00233       free (tmp);
00234     }
00235
00236   return out;
00237 }
00238
00239 char *
00240 digest_md5_print_response (digest_md5_response *r)
00241 {
00242   char *out = NULL;
00243   const char *qop = NULL;
00244   const char *cipher = NULL;
00245
00246   /* Below we assume the mandatory fields are present, verify that
00247      first to avoid crashes. */
00248   if (digest_md5_validate_response (r) != 0)
00249     return NULL;
00250
00251   if (r->qop & DIGEST_MD5_QOP_AUTH_CONF)
00252     qop = "qop=auth-conf";
00253   else if (r->qop & DIGEST_MD5_QOP_AUTH_INT)
00254     qop = "qop=auth-int";
00255   else if (r->qop & DIGEST_MD5_QOP_AUTH)
00256     qop = "qop=auth";
```

```
00257
00258   if (r->cipher & DIGEST_MD5_CIPHER_3DES)
00259     cipher = "cipher=3des";
00260   else if (r->cipher & DIGEST_MD5_CIPHER_DES)
00261     cipher = "cipher=des";
00262   else if (r->cipher & DIGEST_MD5_CIPHER_RC4_40)
00263     cipher = "cipher=rc4-40";
00264   else if (r->cipher & DIGEST_MD5_CIPHER_RC4)
00265     cipher = "cipher=rc4";
00266   else if (r->cipher & DIGEST_MD5_CIPHER_RC4_56)
00267     cipher = "cipher=rc4-56";
00268   else if (r->cipher & DIGEST_MD5_CIPHER_AES_CBC)
00269     cipher = "cipher=aes-cbc";
00270
00271   if (r->username)
00272     if (comma_append (&out, "username", r->username, 1) < 0)
00273       {
00274         free (out);
00275         return NULL;
00276       }
00277
00278   if (r->realm)
00279     if (comma_append (&out, "realm", r->realm, 1) < 0)
00280       {
00281         free (out);
00282         return NULL;
00283       }
00284
00285   if (r->nonce)
00286     if (comma_append (&out, "nonce", r->nonce, 1) < 0)
00287       {
00288         free (out);
00289         return NULL;
00290       }
00291
00292   if (r->cnonce)
00293     if (comma_append (&out, "cnonce", r->cnonce, 1) < 0)
00294       {
00295         free (out);
00296         return NULL;
00297       }
00298
00299   if (r->nc)
00300     {
00301       char *tmp;
00302
00303       if (asprintf (&tmp, "%08lx", r->nc) < 0)
00304         {
00305           free (out);
00306           return NULL;
00307         }
00308
00309       if (comma_append (&out, "nc", tmp, 0) < 0)
00310         {
00311           free (tmp);
00312           free (out);
00313           return NULL;
00314         }
00315
00316       free (tmp);
00317     }
00318
00319   if (qop)
00320     if (comma_append (&out, qop, NULL, 0) < 0)
00321       {
00322         free (out);
00323         return NULL;
00324       }
00325
00326   if (r->digesturi)
00327     if (comma_append (&out, "digest-uri", r->digesturi, 1) < 0)
00328       {
00329         free (out);
00330         return NULL;
00331       }
00332
00333   if (comma_append (&out, "response", r->response, 0) < 0)
00334     {
00335       free (out);
00336       return NULL;
00337     }
00338
00339   if (r->clientmaxbuf)
00340     {
00341       char *tmp;
00342
00343       if (asprintf (&tmp, "%lu", r->clientmaxbuf) < 0)
```

```
00344         {
00345           free (out);
00346           return NULL;
00347         }
00348
00349       if (comma_append (&out, "maxbuf", tmp, 0) < 0)
00350         {
00351           free (tmp);
00352           free (out);
00353           return NULL;
00354         }
00355
00356       free (tmp);
00357     }
00358
00359   if (r->utf8)
00360     if (comma_append (&out, "charset", "utf-8", 0) < 0)
00361       {
00362         free (out);
00363         return NULL;
00364       }
00365
00366   if (cipher)
00367     if (comma_append (&out, cipher, NULL, 0) < 0)
00368       {
00369         free (out);
00370         return NULL;
00371       }
00372
00373   if (r->authzid)
00374     if (comma_append (&out, "authzid", r->authzid, 1) < 0)
00375       {
00376         free (out);
00377         return NULL;
00378       }
00379
00380   return out;
00381 }
00382
00383 char *
00384 digest_md5_print_finish (digest_md5_finish *finish)
00385 {
00386   char *out;
00387
00388   /* Below we assume the mandatory fields are present, verify that
00389      first to avoid crashes. */
00390   if (digest_md5_validate_finish (finish) != 0)
00391     return NULL;
00392
00393   if (asprintf (&out, "rspauth=%s", finish->rspauth) < 0)
00394     return NULL;
00395
00396   return out;
00397 }
```

## 5.143 scram/printer.c File Reference

```
#include <config.h>
#include "printer.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "validate.h"
```

### Functions

- int scram_print_client_first (struct scram_client_first *cf, char **out)
- int scram_print_server_first (struct scram_server_first *sf, char **out)
- int scram_print_client_final (struct scram_client_final *cl, char **out)
- int scram_print_server_final (struct scram_server_final *sl, char **out)

### 5.143.1 Function Documentation

#### 5.143.1.1 scram_print_client_final()

```
int scram_print_client_final (
            struct scram_client_final * cl,
            char ** out )
```

Definition at line 144 of file scram/printer.c.

#### 5.143.1.2 scram_print_client_first()

```
int scram_print_client_first (
            struct scram_client_first * cf,
            char ** out )
```

Definition at line 76 of file scram/printer.c.

#### 5.143.1.3 scram_print_server_final()

```
int scram_print_server_final (
            struct scram_server_final * sl,
            char ** out )
```

Definition at line 164 of file scram/printer.c.

#### 5.143.1.4 scram_print_server_first()

```
int scram_print_server_first (
            struct scram_server_first * sf,
            char ** out )
```

Definition at line 123 of file scram/printer.c.

## 5.144 scram/printer.c

Go to the documentation of this file.

```
00001 /* printer.h --- Convert SCRAM token structures into strings.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
```

```
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get prototypes. */
00025 #include "printer.h"
00026
00027 /* Get free. */
00028 #include <stdlib.h>
00029
00030 /* Get asprintf. */
00031 #include <stdio.h>
00032
00033 /* Get strdup. */
00034 #include <string.h>
00035
00036 /* Get token validator. */
00037 #include "validate.h"
00038
00039 static char *
00040 scram_escape (const char *str)
00041 {
00042   char *out = malloc (strlen (str) * 3 + 1);
00043   char *p = out;
00044
00045   if (!out)
00046     return NULL;
00047
00048   while (*str)
00049     {
00050       if (*str == ',')
00051         {
00052           memcpy (p, "=2C", 3);
00053           p += 3;
00054         }
00055       else if (*str == '=')
00056         {
00057           memcpy (p, "=3D", 3);
00058           p += 3;
00059         }
00060       else
00061         {
00062           *p = *str;
00063           p++;
00064         }
00065       str++;
00066     }
00067   *p = '\0';
00068
00069   return out;
00070 }
00071
00072 /* Print SCRAM client-first token into newly allocated output string
00073    OUT.  Returns 0 on success, -1 on invalid token, and -2 on memory
00074    allocation errors. */
00075 int
00076 scram_print_client_first (struct scram_client_first *cf, char **out)
00077 {
00078   char *username = NULL;
00079   char *authzid = NULL;
00080   int n;
00081
00082   /* Below we assume fields are sensible, so first verify that to
00083      avoid crashes. */
00084   if (!scram_valid_client_first (cf))
00085     return -1;
00086
00087   /* Escape username and authzid. */
00088
00089   username = scram_escape (cf->username);
00090   if (!username)
00091     return -2;
00092
00093   if (cf->authzid)
00094     {
00095       authzid = scram_escape (cf->authzid);
00096       if (!authzid)
00097         {
00098           free (username);
00099           return -2;
00100         }
00101     }
00102
00103   n = asprintf (out, "%c%s%s,%s%s,n=%s,r=%s",
00104                 cf->cbflag,
00105                 cf->cbflag == 'p' ? "=" : "",
00106                 cf->cbflag == 'p' ? cf->cbname : "",
```

```
00107                 authzid ? "a=" : "",
00108                 authzid ? authzid : "", username, cf->client_nonce);
00109
00110   free (username);
00111   free (authzid);
00112
00113   if (n <= 0 || *out == NULL)
00114     return -1;
00115
00116   return 0;
00117 }
00118
00119 /* Print SCRAM server-first token into newly allocated output string
00120    OUT.  Returns 0 on success, -1 on invalid token, and -2 on memory
00121    allocation errors. */
00122 int
00123 scram_print_server_first (struct scram_server_first *sf, char **out)
00124 {
00125   int n;
00126
00127   /* Below we assume fields are sensible, so first verify that to
00128      avoid crashes. */
00129   if (!scram_valid_server_first (sf))
00130     return -1;
00131
00132   n = asprintf (out, "r=%s,s=%s,i=%lu",
00133                 sf->nonce, sf->salt, (unsigned long) sf->iter);
00134   if (n <= 0 || *out == NULL)
00135     return -1;
00136
00137   return 0;
00138 }
00139
00140 /* Print SCRAM client-final token into newly allocated output string
00141    OUT.  Returns 0 on success, -1 on invalid token, and -2 on memory
00142    allocation errors. */
00143 int
00144 scram_print_client_final (struct scram_client_final *cl, char **out)
00145 {
00146   int n;
00147
00148   /* Below we assume fields are sensible, so first verify that to
00149      avoid crashes. */
00150   if (!scram_valid_client_final (cl))
00151     return -1;
00152
00153   n = asprintf (out, "c=%s,r=%s,p=%s", cl->cbind, cl->nonce, cl->proof);
00154   if (n <= 0 || *out == NULL)
00155     return -1;
00156
00157   return 0;
00158 }
00159
00160 /* Print SCRAM server-final token into newly allocated output string
00161    OUT.  Returns 0 on success, -1 on invalid token, and -2 on memory
00162    allocation errors. */
00163 int
00164 scram_print_server_final (struct scram_server_final *sl, char **out)
00165 {
00166   int n;
00167
00168   /* Below we assume fields are sensible, so first verify that to
00169      avoid crashes. */
00170   if (!scram_valid_server_final (sl))
00171     return -1;
00172
00173   n = asprintf (out, "v=%s", sl->verifier);
00174   if (n <= 0 || *out == NULL)
00175     return -1;
00176
00177   return 0;
00178 }
```

## 5.145 digest-md5/printer.h File Reference

```
#include "tokens.h"
```

### Functions

- char * digest_md5_print_challenge (digest_md5_challenge *challenge)
- char * digest_md5_print_response (digest_md5_response *response)
- char * digest_md5_print_finish (digest_md5_finish *out)

## 5.145.1 Function Documentation

### 5.145.1.1 digest_md5_print_challenge()

```
char * digest_md5_print_challenge (
            digest_md5_challenge * challenge )  [extern]
```

Definition at line 71 of file digest-md5/printer.c.

### 5.145.1.2 digest_md5_print_finish()

```
char * digest_md5_print_finish (
            digest_md5_finish * out )  [extern]
```

Definition at line 384 of file digest-md5/printer.c.

### 5.145.1.3 digest_md5_print_response()

```
char * digest_md5_print_response (
            digest_md5_response * response )  [extern]
```

Definition at line 240 of file digest-md5/printer.c.

# 5.146 digest-md5/printer.h

Go to the documentation of this file.

```
00001 /* printer.h --- Convert DIGEST-MD5 token structures into strings.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef DIGEST_MD5_PRINTER_H
00023 # define DIGEST_MD5_PRINTER_H
00024
00025 /* Get token types. */
00026 # include "tokens.h"
00027
00028 extern char *digest_md5_print_challenge (digest_md5_challenge * challenge);
00029
00030 extern char *digest_md5_print_response (digest_md5_response * response);
00031
00032 extern char *digest_md5_print_finish (digest_md5_finish * out);
00033
00034 #endif /* DIGEST_MD5_PRINTER_H */
```

# 5.147 scram/printer.h File Reference

```
#include "tokens.h"
```

### Functions

- int scram_print_client_first (struct scram_client_first *cf, char **out)
- int scram_print_server_first (struct scram_server_first *cf, char **out)
- int scram_print_client_final (struct scram_client_final *cl, char **out)
- int scram_print_server_final (struct scram_server_final *sl, char **out)

## 5.147.1 Function Documentation

### 5.147.1.1 scram_print_client_final()

```
int scram_print_client_final (
            struct scram_client_final * cl,
            char ** out )  [extern]
```

Definition at line 144 of file scram/printer.c.

### 5.147.1.2 scram_print_client_first()

```
int scram_print_client_first (
            struct scram_client_first * cf,
            char ** out )  [extern]
```

Definition at line 76 of file scram/printer.c.

### 5.147.1.3 scram_print_server_final()

```
int scram_print_server_final (
            struct scram_server_final * sl,
            char ** out )  [extern]
```

Definition at line 164 of file scram/printer.c.

### 5.147.1.4 scram_print_server_first()

```
int scram_print_server_first (
            struct scram_server_first * cf,
            char ** out )  [extern]
```

Definition at line 123 of file scram/printer.c.

## 5.148 scram/printer.h

Go to the documentation of this file.

```
00001 /* printer.h --- Convert SCRAM token structures into strings.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef SCRAM_PRINTER_H
00023 # define SCRAM_PRINTER_H
00024
00025 /* Get token types. */
00026 # include "tokens.h"
00027
00028 extern int
00029 scram_print_client_first (struct scram_client_first *cf, char **out);
00030
00031 extern int
00032 scram_print_server_first (struct scram_server_first *cf, char **out);
00033
00034 extern int
00035 scram_print_client_final (struct scram_client_final *cl, char **out);
00036
00037 extern int
00038 scram_print_server_final (struct scram_server_final *sl, char **out);
00039
00040 #endif /* SCRAM_PRINTER_H */
```

## 5.149 scram.h File Reference

```
#include <gsasl.h>
```

## 5.150 scram.h

Go to the documentation of this file.

```
00001 /* scram.h --- Prototypes for SCRAM mechanism
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
```

```
00021
00022 #ifndef SCRAM_H
00023 # define SCRAM_H
00024
00025 # include <gsasl.h>
00026
00027 # ifdef USE_SCRAM_SHA1
00028
00029 #  define GSASL_SCRAM_SHA1_NAME "SCRAM-SHA-1"
00030 #  define GSASL_SCRAM_SHA1_PLUS_NAME "SCRAM-SHA-1-PLUS"
00031
00032 extern Gsasl_mechanism _gsasl_scram_sha1_mechanism;
00033 extern Gsasl_mechanism _gsasl_scram_sha1_plus_mechanism;
00034
00035 int _gsasl_scram_sha1_client_start (Gsasl_session * sctx, void **mech_data);
00036
00037 int
00038 _gsasl_scram_sha1_plus_client_start (Gsasl_session * sctx, void **mech_data);
00039
00040 int
00041 _gsasl_scram_client_step (Gsasl_session * sctx,
00042                           void *mech_data,
00043                           const char *input, size_t input_len,
00044                           char **output, size_t *output_len);
00045
00046 void _gsasl_scram_client_finish (Gsasl_session * sctx, void *mech_data);
00047
00048
00049 int _gsasl_scram_sha1_server_start (Gsasl_session * sctx, void **mech_data);
00050
00051 int
00052 _gsasl_scram_sha1_plus_server_start (Gsasl_session * sctx, void **mech_data);
00053
00054 int _gsasl_scram_server_step (Gsasl_session * sctx,
00055                               void *mech_data,
00056                               const char *input,
00057                               size_t input_len,
00058                               char **output, size_t *output_len);
00059
00060 void _gsasl_scram_server_finish (Gsasl_session * sctx, void *mech_data);
00061
00062 # endif
00063
00064 # ifdef USE_SCRAM_SHA256
00065
00066 #  define GSASL_SCRAM_SHA256_NAME "SCRAM-SHA-256"
00067 #  define GSASL_SCRAM_SHA256_PLUS_NAME "SCRAM-SHA-256-PLUS"
00068
00069 extern Gsasl_mechanism _gsasl_scram_sha256_mechanism;
00070 extern Gsasl_mechanism _gsasl_scram_sha256_plus_mechanism;
00071
00072 int _gsasl_scram_sha256_client_start (Gsasl_session * sctx, void **mech_data);
00073
00074 int
00075 _gsasl_scram_sha256_plus_client_start (Gsasl_session * sctx,
00076                                        void **mech_data);
00077
00078 int
00079 _gsasl_scram_client_step (Gsasl_session * sctx,
00080                           void *mech_data,
00081                           const char *input, size_t input_len,
00082                           char **output, size_t *output_len);
00083
00084 void _gsasl_scram_client_finish (Gsasl_session * sctx, void *mech_data);
00085
00086
00087 int _gsasl_scram_sha256_server_start (Gsasl_session * sctx, void **mech_data);
00088
00089 int
00090 _gsasl_scram_sha256_plus_server_start (Gsasl_session * sctx,
00091                                        void **mech_data);
00092
00093 int _gsasl_scram_server_step (Gsasl_session * sctx,
00094                               void *mech_data,
00095                               const char *input,
00096                               size_t input_len,
00097                               char **output, size_t *output_len);
00098
00099 void _gsasl_scram_server_finish (Gsasl_session * sctx, void *mech_data);
00100
00101 # endif
00102
00103 #endif /* SCRAM_H */
```

## 5.151 tokens.c File Reference

```
#include <config.h>
#include "tokens.h"
#include <stdlib.h>
#include <string.h>
```

### Functions

- void scram_free_client_first (struct scram_client_first ∗cf)
- void scram_free_server_first (struct scram_server_first ∗sf)
- void scram_free_client_final (struct scram_client_final ∗cl)
- void scram_free_server_final (struct scram_server_final ∗sl)

### 5.151.1 Function Documentation

#### 5.151.1.1 scram_free_client_final()

```
void scram_free_client_final (
            struct scram_client_final * cl )
```

Definition at line 54 of file tokens.c.

#### 5.151.1.2 scram_free_client_first()

```
void scram_free_client_first (
            struct scram_client_first * cf )
```

Definition at line 34 of file tokens.c.

#### 5.151.1.3 scram_free_server_final()

```
void scram_free_server_final (
            struct scram_server_final * sl )
```

Definition at line 64 of file tokens.c.

#### 5.151.1.4 scram_free_server_first()

```
void scram_free_server_first (
            struct scram_server_first * sf )
```

Definition at line 45 of file tokens.c.

## 5.152 tokens.c

Go to the documentation of this file.

```
00001 /* tokens.c --- Free allocated data in SCRAM tokens.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get prototypes. */
00025 #include "tokens.h"
00026
00027 /* Get free. */
00028 #include <stdlib.h>
00029
00030 /* Get memset. */
00031 #include <string.h>
00032
00033 void
00034 scram_free_client_first (struct scram_client_first *cf)
00035 {
00036   free (cf->cbname);
00037   free (cf->authzid);
00038   free (cf->username);
00039   free (cf->client_nonce);
00040
00041   memset (cf, 0, sizeof (*cf));
00042 }
00043
00044 void
00045 scram_free_server_first (struct scram_server_first *sf)
00046 {
00047   free (sf->nonce);
00048   free (sf->salt);
00049
00050   memset (sf, 0, sizeof (*sf));
00051 }
00052
00053 void
00054 scram_free_client_final (struct scram_client_final *cl)
00055 {
00056   free (cl->cbind);
00057   free (cl->nonce);
00058   free (cl->proof);
00059
00060   memset (cl, 0, sizeof (*cl));
00061 }
00062
00063 void
00064 scram_free_server_final (struct scram_server_final *sl)
00065 {
00066   free (sl->verifier);
00067
00068   memset (sl, 0, sizeof (*sl));
00069 }
```

## 5.153 digest-md5/tokens.h File Reference

```
#include <stddef.h>
```

**Data Structures**

- struct digest_md5_challenge
- struct digest_md5_response
- struct digest_md5_finish

**Macros**

- #define DIGEST_MD5_LENGTH 16
- #define DIGEST_MD5_RESPONSE_LENGTH 32

**Typedefs**

- typedef enum digest_md5_qop digest_md5_qop
- typedef enum digest_md5_cipher digest_md5_cipher
- typedef struct digest_md5_challenge digest_md5_challenge
- typedef struct digest_md5_response digest_md5_response
- typedef struct digest_md5_finish digest_md5_finish

**Enumerations**

- enum digest_md5_qop { DIGEST_MD5_QOP_AUTH = 1 , DIGEST_MD5_QOP_AUTH_INT = 2 , DIGEST_MD5_QOP_AUTH_CONF = 4 }
- enum digest_md5_cipher {
  DIGEST_MD5_CIPHER_DES = 1 , DIGEST_MD5_CIPHER_3DES = 2 , DIGEST_MD5_CIPHER_RC4 = 4 ,
  DIGEST_MD5_CIPHER_RC4_40 = 8 ,
  DIGEST_MD5_CIPHER_RC4_56 = 16 , DIGEST_MD5_CIPHER_AES_CBC = 32 }

## 5.153.1 Macro Definition Documentation

### 5.153.1.1 DIGEST_MD5_LENGTH

```
#define DIGEST_MD5_LENGTH 16
```

Definition at line 29 of file digest-md5/tokens.h.

### 5.153.1.2 DIGEST_MD5_RESPONSE_LENGTH

```
#define DIGEST_MD5_RESPONSE_LENGTH 32
```

Definition at line 94 of file digest-md5/tokens.h.

## 5.153.2 Typedef Documentation

### 5.153.2.1 digest_md5_challenge

```
typedef struct digest_md5_challenge digest_md5_challenge
```

Definition at line 92 of file digest-md5/tokens.h.

#### 5.153.2.2 digest_md5_cipher

```
typedef enum digest_md5_cipher digest_md5_cipher
```

Definition at line 50 of file digest-md5/tokens.h.

#### 5.153.2.3 digest_md5_finish

```
typedef struct digest_md5_finish digest_md5_finish
```

Definition at line 149 of file digest-md5/tokens.h.

#### 5.153.2.4 digest_md5_qop

```
typedef enum digest_md5_qop digest_md5_qop
```

Definition at line 38 of file digest-md5/tokens.h.

#### 5.153.2.5 digest_md5_response

```
typedef struct digest_md5_response digest_md5_response
```

Definition at line 140 of file digest-md5/tokens.h.

### 5.153.3 Enumeration Type Documentation

#### 5.153.3.1 digest_md5_cipher

```
enum digest_md5_cipher
```

**Enumerator**

| | |
|---|---|
| DIGEST_MD5_CIPHER_DES | |
| DIGEST_MD5_CIPHER_3DES | |
| DIGEST_MD5_CIPHER_RC4 | |
| DIGEST_MD5_CIPHER_RC4_40 | |
| DIGEST_MD5_CIPHER_RC4_56 | |
| DIGEST_MD5_CIPHER_AES_CBC | |

Definition at line 41 of file digest-md5/tokens.h.

#### 5.153.3.2 digest_md5_qop

```
enum digest_md5_qop
```

**Enumerator**

| | |
|---|---|
| DIGEST_MD5_QOP_AUTH | |
| DIGEST_MD5_QOP_AUTH_INT | |
| DIGEST_MD5_QOP_AUTH_CONF | |

Definition at line 32 of file digest-md5/tokens.h.

## 5.154 digest-md5/tokens.h

Go to the documentation of this file.

```
00001 /* tokens.h --- Types for DIGEST-MD5 tokens.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef DIGEST_MD5_TOKENS_H
00023 # define DIGEST_MD5_TOKENS_H
00024
00025 /* Get size_t. */
00026 # include <stddef.h>
00027
00028 /* Length of MD5 output. */
00029 # define DIGEST_MD5_LENGTH 16
00030
00031 /* Quality of Protection types. */
00032 enum digest_md5_qop
00033 {
00034   DIGEST_MD5_QOP_AUTH = 1,
00035   DIGEST_MD5_QOP_AUTH_INT = 2,
00036   DIGEST_MD5_QOP_AUTH_CONF = 4
00037 };
00038 typedef enum digest_md5_qop digest_md5_qop;
00039
00040 /* Cipher types. */
00041 enum digest_md5_cipher
00042 {
00043   DIGEST_MD5_CIPHER_DES = 1,
00044   DIGEST_MD5_CIPHER_3DES = 2,
00045   DIGEST_MD5_CIPHER_RC4 = 4,
00046   DIGEST_MD5_CIPHER_RC4_40 = 8,
00047   DIGEST_MD5_CIPHER_RC4_56 = 16,
00048   DIGEST_MD5_CIPHER_AES_CBC = 32
00049 };
00050 typedef enum digest_md5_cipher digest_md5_cipher;
00051
00052 /*
00053  * digest-challenge  =
00054  *       1#( realm | nonce | qop-options | stale | server_maxbuf | charset
00055  *             algorithm | cipher-opts | auth-param )
00056  *
00057  * realm             = "realm" "=" <"> realm-value <">
00058  * realm-value       = qdstr-val
00059  * nonce             = "nonce" "=" <"> nonce-value <">
00060  * nonce-value       = *qdtext
00061  * qop-options       = "qop" "=" <"> qop-list <">
00062  * qop-list          = 1#qop-value
00063  * qop-value         = "auth" | "auth-int" | "auth-conf" | qop-token
00064  *                    ;; qop-token is reserved for identifying future
00065  *                    ;; extensions to DIGEST-MD5
```

```
00066  * qop-token         = token
00067  * stale             = "stale" "=" "true"
00068  * server_maxbuf     = "maxbuf" "=" maxbuf-value
00069  * maxbuf-value      = 1*DIGIT
00070  * charset           = "charset" "=" "utf-8"
00071  * algorithm         = "algorithm" "=" "md5-sess"
00072  * cipher-opts       = "cipher" "=" <"> 1#cipher-value <">
00073  * cipher-value      = "3des" | "des" | "rc4-40" | "rc4" |
00074  *                     "rc4-56" | "aes-cbc" | cipher-token
00075  *                     ;; "des" and "3des" ciphers are obsolete.
00076  *                     ;; cipher-token is reserved for new ciphersuites
00077  * cipher-token      = token
00078  * auth-param        = token "=" ( token | quoted-string )
00079  *
00080  */
00081 struct digest_md5_challenge
00082 {
00083   size_t nrealms;
00084   char **realms;
00085   char *nonce;
00086   int qops;
00087   int stale;
00088   unsigned long servermaxbuf;
00089   int utf8;
00090   int ciphers;
00091 };
00092 typedef struct digest_md5_challenge digest_md5_challenge;
00093
00094 # define DIGEST_MD5_RESPONSE_LENGTH 32
00095
00096 /*
00097  * digest-response  = 1#( username | realm | nonce | cnonce |
00098  *                        nonce-count | qop | digest-uri | response |
00099  *                        client_maxbuf | charset | cipher | authzid |
00100  *                        auth-param )
00101  *
00102  *     username         = "username" "=" <"> username-value <">
00103  *     username-value   = qdstr-val
00104  *     cnonce           = "cnonce" "=" <"> cnonce-value <">
00105  *     cnonce-value     = *qdtext
00106  *     nonce-count      = "nc" "=" nc-value
00107  *     nc-value         = 8LHEX
00108  *     client_maxbuf    = "maxbuf" "=" maxbuf-value
00109  *     qop              = "qop" "=" qop-value
00110  *     digest-uri       = "digest-uri" "=" <"> digest-uri-value <">
00111  *     digest-uri-value  = serv-type "/" host [ "/" serv-name ]
00112  *     serv-type        = 1*ALPHA
00113  *     serv-name        = host
00114  *     response         = "response" "=" response-value
00115  *     response-value   = 32LHEX
00116  *     LHEX             = "0" | "1" | "2" | "3" |
00117  *                        "4" | "5" | "6" | "7" |
00118  *                        "8" | "9" | "a" | "b" |
00119  *                        "c" | "d" | "e" | "f"
00120  *     cipher           = "cipher" "=" cipher-value
00121  *     authzid          = "authzid" "=" <"> authzid-value <">
00122  *     authzid-value    = qdstr-val
00123  *
00124  */
00125 struct digest_md5_response
00126 {
00127   char *username;
00128   char *realm;
00129   char *nonce;
00130   char *cnonce;
00131   unsigned long nc;
00132   digest_md5_qop qop;
00133   char *digesturi;
00134   unsigned long clientmaxbuf;
00135   int utf8;
00136   digest_md5_cipher cipher;
00137   char *authzid;
00138   char response[DIGEST_MD5_RESPONSE_LENGTH + 1];
00139 };
00140 typedef struct digest_md5_response digest_md5_response;
00141
00142 /*
00143  * response-auth = "rspauth" "=" response-value
00144  */
00145 struct digest_md5_finish
00146 {
00147   char rspauth[DIGEST_MD5_RESPONSE_LENGTH + 1];
00148 };
00149 typedef struct digest_md5_finish digest_md5_finish;
00150
00151 #endif /* DIGEST_MD5_TOKENS_H */
```

## 5.155 scram/tokens.h File Reference

```
#include <stddef.h>
```

### Data Structures

- struct scram_client_first
- struct scram_server_first
- struct scram_client_final
- struct scram_server_final

### Functions

- void scram_free_client_first (struct scram_client_first *cf)
- void scram_free_server_first (struct scram_server_first *sf)
- void scram_free_client_final (struct scram_client_final *cl)
- void scram_free_server_final (struct scram_server_final *sl)

### 5.155.1 Function Documentation

#### 5.155.1.1 scram_free_client_final()

```
void scram_free_client_final (
            struct scram_client_final * cl )  [extern]
```

Definition at line 54 of file tokens.c.

#### 5.155.1.2 scram_free_client_first()

```
void scram_free_client_first (
            struct scram_client_first * cf )  [extern]
```

Definition at line 34 of file tokens.c.

#### 5.155.1.3 scram_free_server_final()

```
void scram_free_server_final (
            struct scram_server_final * sl )  [extern]
```

Definition at line 64 of file tokens.c.

#### 5.155.1.4 scram_free_server_first()

```
void scram_free_server_first (
            struct scram_server_first * sf )  [extern]
```

Definition at line 45 of file tokens.c.

## 5.156 scram/tokens.h

Go to the documentation of this file.

```
00001 /* tokens.h --- Types for SCRAM tokens.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef SCRAM_TOKENS_H
00023 # define SCRAM_TOKENS_H
00024
00025 /* Get size_t. */
00026 # include <stddef.h>
00027
00028 struct scram_client_first
00029 {
00030   char cbflag;
00031   char *cbname;
00032   char *authzid;
00033   char *username;
00034   char *client_nonce;
00035 };
00036
00037 struct scram_server_first
00038 {
00039   char *nonce;
00040   char *salt;
00041   size_t iter;
00042 };
00043
00044 struct scram_client_final
00045 {
00046   char *cbind;
00047   char *nonce;
00048   char *proof;
00049 };
00050
00051 struct scram_server_final
00052 {
00053   char *verifier;
00054 };
00055
00056 extern void scram_free_client_first (struct scram_client_first *cf);
00057
00058 extern void scram_free_server_first (struct scram_server_first *sf);
00059
00060 extern void scram_free_client_final (struct scram_client_final *cl);
00061
00062 extern void scram_free_server_final (struct scram_server_final *sl);
00063
00064 #endif /* SCRAM_TOKENS_H */
```

## 5.157 tools.c File Reference

```
#include <config.h>
#include "tools.h"
#include "mechtools.h"
```

### Functions

- int set_saltedpassword (Gsasl_session *sctx, Gsasl_hash hash, const char *hashbuf)

## 5.157.1 Function Documentation

### 5.157.1.1 set_saltedpassword()

```
int set_saltedpassword (
            Gsasl_session * sctx,
            Gsasl_hash hash,
            const char * hashbuf )
```

Definition at line 30 of file tools.c.

# 5.158 tools.c

Go to the documentation of this file.

```
00001 /* tools.c --- Shared client/server SCRAM code
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 #include "tools.h"
00025 #include "mechtools.h"
00026
00027 /* Hex encode HASHBUF which is HASH digest output and set salted
00028    password property to the hex encoded value. */
00029 int
00030 set_saltedpassword (Gsasl_session *sctx, Gsasl_hash hash, const char *hashbuf)
00031 {
00032   char hexstr[GSASL_HASH_MAX_SIZE * 2 + 1];
00033
00034   _gsasl_hex_encode (hashbuf, gsasl_hash_length (hash), hexstr);
00035   return gsasl_property_set (sctx, GSASL_SCRAM_SALTED_PASSWORD, hexstr);
00036 }
```

# 5.159 tools.h File Reference

```
#include <gsasl.h>
```

### Functions

- int set_saltedpassword (Gsasl_session *sctx, Gsasl_hash hash, const char *hashbuf)

### 5.159.1 Function Documentation

#### 5.159.1.1 set_saltedpassword()

```
int set_saltedpassword (
            Gsasl_session * sctx,
            Gsasl_hash hash,
            const char * hashbuf )  [extern]
```

Definition at line 30 of file tools.c.

## 5.160 tools.h

Go to the documentation of this file.

```
00001 /* tools.h --- Shared client/server SCRAM code
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef SCRAM_TOOLS_H
00023 # define SCRAM_TOOLS_H
00024
00025 # include <gsasl.h>
00026
00027 extern int
00028 set_saltedpassword (Gsasl_session * sctx,
00029                     Gsasl_hash hash, const char *hashbuf);
00030
00031 #endif /* SCRAM_TOOLS_H */
```

## 5.161 digest-md5/validate.c File Reference

```
#include <config.h>
#include "validate.h"
#include <string.h>
```

### Functions

- int digest_md5_validate_challenge (digest_md5_challenge *c)
- int digest_md5_validate_response (digest_md5_response *r)
- int digest_md5_validate_finish (digest_md5_finish *f)
- int digest_md5_validate (digest_md5_challenge *c, digest_md5_response *r)

### 5.161.1 Function Documentation

#### 5.161.1.1 digest_md5_validate()

```
int digest_md5_validate (
            digest_md5_challenge * c,
            digest_md5_response * r )
```

Definition at line 113 of file digest-md5/validate.c.

#### 5.161.1.2 digest_md5_validate_challenge()

```
int digest_md5_validate_challenge (
            digest_md5_challenge * c )
```

Definition at line 31 of file digest-md5/validate.c.

#### 5.161.1.3 digest_md5_validate_finish()

```
int digest_md5_validate_finish (
            digest_md5_finish * f )
```

Definition at line 100 of file digest-md5/validate.c.

#### 5.161.1.4 digest_md5_validate_response()

```
int digest_md5_validate_response (
            digest_md5_response * r )
```

Definition at line 50 of file digest-md5/validate.c.

## 5.162 digest-md5/validate.c

Go to the documentation of this file.

```
00001 /* validate.c --- Validate consistency of DIGEST-MD5 tokens.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
```

```
00024 /* Get prototypes. */
00025 #include "validate.h"
00026
00027 /* Get strcmp, strlen. */
00028 #include <string.h>
00029
00030 int
00031 digest_md5_validate_challenge (digest_md5_challenge *c)
00032 {
00033   /* This directive is required and MUST appear exactly once; if
00034      not present, or if multiple instances are present, the
00035      client should abort the authentication exchange. */
00036   if (!c->nonce)
00037     return -1;
00038
00039   /* This directive must be present exactly once if "auth-conf" is
00040      offered in the "qop-options" directive */
00041   if (c->ciphers && !(c->qops & DIGEST_MD5_QOP_AUTH_CONF))
00042     return -1;
00043   if (!c->ciphers && (c->qops & DIGEST_MD5_QOP_AUTH_CONF))
00044     return -1;
00045
00046   return 0;
00047 }
00048
00049 int
00050 digest_md5_validate_response (digest_md5_response *r)
00051 {
00052   /* This directive is required and MUST be present exactly
00053      once; otherwise, authentication fails. */
00054   if (!r->username)
00055     return -1;
00056
00057   /* This directive is required and MUST be present exactly
00058      once; otherwise, authentication fails. */
00059   if (!r->nonce)
00060     return -1;
00061
00062   /* This directive is required and MUST be present exactly once;
00063      otherwise, authentication fails. */
00064   if (!r->cnonce)
00065     return -1;
00066
00067   /* This directive is required and MUST be present exactly once;
00068      otherwise, or if the value is 0, authentication fails. */
00069   if (!r->nc)
00070     return -1;
00071
00072   /* This directive is required and MUST be present exactly
00073      once; if multiple instances are present, the client MUST
00074      abort the authentication exchange. */
00075   if (!r->digesturi)
00076     return -1;
00077
00078   /* This directive is required and MUST be present exactly
00079      once; otherwise, authentication fails. */
00080   if (!*r->response)
00081     return -1;
00082
00083   if (strlen (r->response) != DIGEST_MD5_RESPONSE_LENGTH)
00084     return -1;
00085
00086   /* This directive MUST appear exactly once if "auth-conf" is
00087      negotiated; if required and not present, authentication fails.
00088      If the client recognizes no cipher and the server only advertised
00089      "auth-conf" in the qop option, the client MUST abort the
00090      authentication exchange.  */
00091   if (r->qop == DIGEST_MD5_QOP_AUTH_CONF && !r->cipher)
00092     return -1;
00093   if (r->qop != DIGEST_MD5_QOP_AUTH_CONF && r->cipher)
00094     return -1;
00095
00096   return 0;
00097 }
00098
00099 int
00100 digest_md5_validate_finish (digest_md5_finish *f)
00101 {
00102   if (!*f->rspauth)
00103     return -1;
00104
00105   /* A string of 32 hex digits */
00106   if (strlen (f->rspauth) != DIGEST_MD5_RESPONSE_LENGTH)
00107     return -1;
00108
00109   return 0;
00110 }
```

```
00111
00112 int
00113 digest_md5_validate (digest_md5_challenge *c, digest_md5_response *r)
00114 {
00115   if (!c->nonce || !r->nonce)
00116     return -1;
00117
00118   if (strcmp (c->nonce, r->nonce) != 0)
00119     return -1;
00120
00121   if (r->nc != 1)
00122     return -1;
00123
00124   if (!c->utf8 && r->utf8)
00125     return -1;
00126
00127   if (!((c->qops ? c->qops : DIGEST_MD5_QOP_AUTH) &
00128         (r->qop ? r->qop : DIGEST_MD5_QOP_AUTH)))
00129     return -1;
00130
00131   if ((r->qop & DIGEST_MD5_QOP_AUTH_CONF) && !(c->ciphers & r->cipher))
00132     return -1;
00133
00134   /* FIXME: Check more? */
00135
00136   return 0;
00137 }
```

## 5.163 scram/validate.c File Reference

```
#include <config.h>
#include "validate.h"
#include <string.h>
```

### Functions

- bool scram_valid_client_first (struct scram_client_first *cf)
- bool scram_valid_server_first (struct scram_server_first *sf)
- bool scram_valid_client_final (struct scram_client_final *cl)
- bool scram_valid_server_final (struct scram_server_final *sl)

### 5.163.1 Function Documentation

#### 5.163.1.1 scram_valid_client_final()

```
bool scram_valid_client_final (
            struct scram_client_final * cl )
```

Definition at line 103 of file scram/validate.c.

#### 5.163.1.2 scram_valid_client_first()

```
bool scram_valid_client_first (
            struct scram_client_first * cf )
```

Definition at line 31 of file scram/validate.c.

#### 5.163.1.3 scram_valid_server_final()

```
bool scram_valid_server_final (
             struct scram_server_final * sl )
```

Definition at line 133 of file scram/validate.c.

#### 5.163.1.4 scram_valid_server_first()

```
bool scram_valid_server_first (
             struct scram_server_first * sf )
```

Definition at line 78 of file scram/validate.c.

## 5.164 scram/validate.c

Go to the documentation of this file.

```
00001 /* validate.c --- Validate consistency of SCRAM tokens.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get prototypes. */
00025 #include "validate.h"
00026
00027 /* Get strcmp, strlen. */
00028 #include <string.h>
00029
00030 bool
00031 scram_valid_client_first (struct scram_client_first *cf)
00032 {
00033   /* Check that cbflag is one of permitted values. */
00034   switch (cf->cbflag)
00035     {
00036     case 'p':
00037     case 'n':
00038     case 'y':
00039       break;
00040
00041     default:
00042       return false;
00043     }
00044
00045   /* Check that cbname is only set when cbflag is p. */
00046   if (cf->cbflag == 'p' && cf->cbname == NULL)
00047     return false;
00048   else if (cf->cbflag != 'p' && cf->cbname != NULL)
00049     return false;
00050
00051   if (cf->cbname)
00052     {
00053       const char *p = cf->cbname;
00054
00055       while (*p && strchr ("ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

```
00056                           "abcdefghijklmnopqrstuvwxyz" "0123456789.-", *p))
00057         p++;
00058       if (*p)
00059         return false;
00060     }
00061
00062   /* We require a non-zero username string. */
00063   if (cf->username == NULL || *cf->username == '\0')
00064     return false;
00065
00066   /* We require a non-zero client nonce. */
00067   if (cf->client_nonce == NULL || *cf->client_nonce == '\0')
00068     return false;
00069
00070   /* Nonce cannot contain ','. */
00071   if (strchr (cf->client_nonce, ','))
00072     return false;
00073
00074   return true;
00075 }
00076
00077 bool
00078 scram_valid_server_first (struct scram_server_first *sf)
00079 {
00080   /* We require a non-zero nonce. */
00081   if (sf->nonce == NULL || *sf->nonce == '\0')
00082     return false;
00083
00084   /* Nonce cannot contain ','. */
00085   if (strchr (sf->nonce, ','))
00086     return false;
00087
00088   /* We require a non-zero salt. */
00089   if (sf->salt == NULL || *sf->salt == '\0')
00090     return false;
00091
00092   /* FIXME check that salt is valid base64. */
00093   if (strchr (sf->salt, ','))
00094     return false;
00095
00096   if (sf->iter == 0)
00097     return false;
00098
00099   return true;
00100 }
00101
00102 bool
00103 scram_valid_client_final (struct scram_client_final *cl)
00104 {
00105   /* We require a non-zero cbind. */
00106   if (cl->cbind == NULL || *cl->cbind == '\0')
00107     return false;
00108
00109   /* FIXME check that cbind is valid base64. */
00110   if (strchr (cl->cbind, ','))
00111     return false;
00112
00113   /* We require a non-zero nonce. */
00114   if (cl->nonce == NULL || *cl->nonce == '\0')
00115     return false;
00116
00117   /* Nonce cannot contain ','. */
00118   if (strchr (cl->nonce, ','))
00119     return false;
00120
00121   /* We require a non-zero proof. */
00122   if (cl->proof == NULL || *cl->proof == '\0')
00123     return false;
00124
00125   /* FIXME check that proof is valid base64. */
00126   if (strchr (cl->proof, ','))
00127     return false;
00128
00129   return true;
00130 }
00131
00132 bool
00133 scram_valid_server_final (struct scram_server_final *sl)
00134 {
00135   /* We require a non-zero verifier. */
00136   if (sl->verifier == NULL || *sl->verifier == '\0')
00137     return false;
00138
00139   /* FIXME check that verifier is valid base64. */
00140   if (strchr (sl->verifier, ','))
00141     return false;
00142
```

```
00143   return true;
00144 }
```

# 5.165 digest-md5/validate.h File Reference

```
#include "tokens.h"
```

### Functions

- int digest_md5_validate_challenge (digest_md5_challenge *c)
- int digest_md5_validate_response (digest_md5_response *r)
- int digest_md5_validate_finish (digest_md5_finish *f)
- int digest_md5_validate (digest_md5_challenge *c, digest_md5_response *r)

## 5.165.1 Function Documentation

### 5.165.1.1 digest_md5_validate()

```
int digest_md5_validate (
            digest_md5_challenge * c,
            digest_md5_response * r )  [extern]
```

Definition at line 113 of file digest-md5/validate.c.

### 5.165.1.2 digest_md5_validate_challenge()

```
int digest_md5_validate_challenge (
            digest_md5_challenge * c )  [extern]
```

Definition at line 31 of file digest-md5/validate.c.

### 5.165.1.3 digest_md5_validate_finish()

```
int digest_md5_validate_finish (
            digest_md5_finish * f )  [extern]
```

Definition at line 100 of file digest-md5/validate.c.

### 5.165.1.4 digest_md5_validate_response()

```
int digest_md5_validate_response (
            digest_md5_response * r )  [extern]
```

Definition at line 50 of file digest-md5/validate.c.

## 5.166 digest-md5/validate.h

Go to the documentation of this file.

```
00001 /* validate.h --- Validate consistency of DIGEST-MD5 tokens.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef DIGEST_MD5_VALIDATE_H
00023 # define DIGEST_MD5_VALIDATE_H
00024
00025 /* Get token types. */
00026 # include "tokens.h"
00027
00028 extern int digest_md5_validate_challenge (digest_md5_challenge * c);
00029
00030 extern int digest_md5_validate_response (digest_md5_response * r);
00031
00032 extern int digest_md5_validate_finish (digest_md5_finish * f);
00033
00034 extern int digest_md5_validate (digest_md5_challenge * c,
00035                                 digest_md5_response * r);
00036
00037 #endif /* DIGEST_MD5_VALIDATE_H */
```

## 5.167 scram/validate.h File Reference

```
#include "tokens.h"
#include <stdbool.h>
```

### Functions

- bool scram_valid_client_first (struct scram_client_first *cf)
- bool scram_valid_server_first (struct scram_server_first *sf)
- bool scram_valid_client_final (struct scram_client_final *cl)
- bool scram_valid_server_final (struct scram_server_final *sl)

### 5.167.1 Function Documentation

#### 5.167.1.1 scram_valid_client_final()

```
bool scram_valid_client_final (
            struct scram_client_final * cl )  [extern]
```

Definition at line 103 of file scram/validate.c.

#### 5.167.1.2 scram_valid_client_first()

```
bool scram_valid_client_first (
            struct scram_client_first * cf )  [extern]
```

Definition at line 31 of file scram/validate.c.

#### 5.167.1.3 scram_valid_server_final()

```
bool scram_valid_server_final (
            struct scram_server_final * sl )  [extern]
```

Definition at line 133 of file scram/validate.c.

#### 5.167.1.4 scram_valid_server_first()

```
bool scram_valid_server_first (
            struct scram_server_first * sf )  [extern]
```

Definition at line 78 of file scram/validate.c.

## 5.168 scram/validate.h

Go to the documentation of this file.

```
00001 /* validate.h --- Validate consistency of SCRAM tokens.
00002  * Copyright (C) 2009-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef SCRAM_VALIDATE_H
00023 # define SCRAM_VALIDATE_H
00024
00025 /* Get token types. */
00026 # include "tokens.h"
00027
00028 /* Get bool. */
00029 # include <stdbool.h>
00030
00031 extern bool scram_valid_client_first (struct scram_client_first *cf);
00032
00033 extern bool scram_valid_server_first (struct scram_server_first *sf);
00034
00035 extern bool scram_valid_client_final (struct scram_client_final *cl);
00036
00037 extern bool scram_valid_server_final (struct scram_server_final *sl);
00038
00039 #endif /* SCRAM_VALIDATE_H */
```

## 5.169 securid.h File Reference

```
#include <gsasl.h>
```

### Macros

- #define GSASL_SECURID_NAME "SECURID"

### Functions

- int _gsasl_securid_client_start (Gsasl_session *sctx, void **mech_data)
- int _gsasl_securid_client_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)
- void _gsasl_securid_client_finish (Gsasl_session *sctx, void *mech_data)
- int _gsasl_securid_server_step (Gsasl_session *sctx, void *mech_data, const char *input, size_t input_len, char **output, size_t *output_len)

### Variables

- Gsasl_mechanism _gsasl_securid_mechanism

### 5.169.1 Macro Definition Documentation

#### 5.169.1.1 GSASL_SECURID_NAME

```
#define GSASL_SECURID_NAME "SECURID"
```

Definition at line 27 of file securid.h.

### 5.169.2 Function Documentation

#### 5.169.2.1 _gsasl_securid_client_finish()

```
void _gsasl_securid_client_finish (
            Gsasl_session * sctx,
            void * mech_data )  [extern]
```

#### 5.169.2.2 _gsasl_securid_client_start()

```
int _gsasl_securid_client_start (
            Gsasl_session * sctx,
            void ** mech_data )  [extern]
```

#### 5.169.2.3 _gsasl_securid_client_step()

```
int _gsasl_securid_client_step (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )  [extern]
```

Definition at line 53 of file securid/client.c.

#### 5.169.2.4 _gsasl_securid_server_step()

```
int _gsasl_securid_server_step (
              Gsasl_session * sctx,
              void * mech_data,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )  [extern]
```

### 5.169.3 Variable Documentation

#### 5.169.3.1 _gsasl_securid_mechanism

```
Gsasl_mechanism _gsasl_securid_mechanism  [extern]
```

Definition at line 27 of file securid/mechinfo.c.

## 5.170 securid.h

Go to the documentation of this file.

```
00001 /* securid.h --- Prototypes for SASL mechanism SECURID as defined in RFC 2808.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef SECURID_H
00023 # define SECURID_H
00024
00025 # include <gsasl.h>
00026
00027 # define GSASL_SECURID_NAME "SECURID"
```

```
00028
00029 extern Gsasl_mechanism _gsasl_securid_mechanism;
00030
00031 extern int _gsasl_securid_client_start (Gsasl_session * sctx,
00032                                         void **mech_data);
00033 extern int _gsasl_securid_client_step (Gsasl_session * sctx,
00034                                        void *mech_data,
00035                                        const char *input, size_t input_len,
00036                                        char **output, size_t *output_len);
00037 extern void _gsasl_securid_client_finish (Gsasl_session * sctx,
00038                                           void *mech_data);
00039
00040 extern int _gsasl_securid_server_step (Gsasl_session * sctx,
00041                                        void *mech_data,
00042                                        const char *input, size_t input_len,
00043                                        char **output, size_t *output_len);
00044
00045 #endif /* SECURID_H */
```

## 5.171 base64.c File Reference

```
#include <config.h>
#include "internal.h"
#include "base64.h"
#include "mechtools.h"
```

### Functions

- int gsasl_base64_to (const char *in, size_t inlen, char **out, size_t *outlen)
- int gsasl_base64_from (const char *in, size_t inlen, char **out, size_t *outlen)
- int gsasl_hex_to (const char *in, size_t inlen, char **out, size_t *outlen)
- int gsasl_hex_from (const char *in, char **out, size_t *outlen)

### 5.171.1 Function Documentation

#### 5.171.1.1 gsasl_base64_from()

```
int gsasl_base64_from (
            const char * in,
            size_t inlen,
            char ** out,
            size_t * outlen )
```

gsasl_base64_from:

**Parameters**

| | |
|---|---|
| *in* | input byte array |
| *inlen* | size of input byte array |
| *out* | pointer to newly allocated output byte array |
| *outlen* | pointer to size of newly allocated output byte array |

Decode Base64 data. The @out buffer must be deallocated by the caller.

Return value: Returns GSASL_OK on success, GSASL_BASE64_ERROR if input was invalid, and GSASL_↩ MALLOC_ERROR on memory allocation errors.

Since: 0.2.2

Definition at line 74 of file base64.c.

#### 5.171.1.2 gsasl_base64_to()

```
int gsasl_base64_to (
            const char * in,
            size_t inlen,
            char ** out,
            size_t * outlen )
```

gsasl_base64_to:

**Parameters**

| | |
|---|---|
| *in* | input byte array. |
| *inlen* | size of input byte array. |
| *out* | pointer to newly allocated base64-encoded string. |
| *outlen* | pointer to size of newly allocated base64-encoded string. |

Encode data as base64. The @out string is zero terminated, and @outlen holds the length excluding the terminating zero. The @out buffer must be deallocated by the caller.

Return value: Returns GSASL_OK on success, or GSASL_MALLOC_ERROR if input was too large or memory allocation fail.

Since: 0.2.2

Definition at line 44 of file base64.c.

#### 5.171.1.3 gsasl_hex_from()

```
int gsasl_hex_from (
            const char * in,
            char ** out,
            size_t * outlen )
```

gsasl_hex_from:

**Parameters**

| | |
|---|---|
| *in* | input byte array |
| *out* | pointer to newly allocated output byte array |
| *outlen* | pointer to size of newly allocated output byte array |

Decode hex data. The @out buffer must be deallocated by the caller.

Return value: Returns GSASL_OK on success, GSASL_BASE64_ERROR if input was invalid, and GSASL_↩ MALLOC_ERROR on memory allocation errors.

Since: 1.10

Definition at line 143 of file base64.c.

#### 5.171.1.4 gsasl_hex_to()

```
int gsasl_hex_to (
             const char * in,
             size_t inlen,
             char ** out,
             size_t * outlen )
```

gsasl_hex_to:

**Parameters**

| | |
|---|---|
| *in* | input byte array. |
| *inlen* | size of input byte array. |
| *out* | pointer to newly allocated hex-encoded string. |
| *outlen* | pointer to size of newly allocated hex-encoded string. |

Hex encode data. The @out string is zero terminated, and @outlen holds the length excluding the terminating zero. The @out buffer must be deallocated by the caller.

Return value: Returns GSASL_OK on success, or GSASL_MALLOC_ERROR if input was too large or memory allocation fail.

Since: 1.10

Definition at line 110 of file base64.c.

## 5.172 base64.c

Go to the documentation of this file.

```
00001 /* base64.c --- Base64 encoding/decoding functions.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
```

```
00023 #include "internal.h"
00024
00025 #include "base64.h"
00026
00043 int
00044 gsasl_base64_to (const char *in, size_t inlen, char **out, size_t *outlen)
00045 {
00046   idx_t len = base64_encode_alloc (in, inlen, out);
00047
00048   if (outlen)
00049     *outlen = len;
00050
00051   if (*out == NULL)
00052     return GSASL_MALLOC_ERROR;
00053
00054   return GSASL_OK;
00055 }
00056
00073 int
00074 gsasl_base64_from (const char *in, size_t inlen, char **out, size_t *outlen)
00075 {
00076   idx_t ol;
00077   int ok = base64_decode_alloc (in, inlen, out, &ol);
00078
00079   if (!ok)
00080     return GSASL_BASE64_ERROR;
00081
00082   if (outlen)
00083     *outlen = ol;
00084
00085   if (*out == NULL)
00086     return GSASL_MALLOC_ERROR;
00087
00088   return GSASL_OK;
00089 }
00090
00091 #include "mechtools.h"
00092
00109 int
00110 gsasl_hex_to (const char *in, size_t inlen, char **out, size_t *outlen)
00111 {
00112   size_t len = 2 * inlen;
00113
00114   if (outlen)
00115     *outlen = len;
00116
00117   *out = malloc (len + 1);
00118   if (*out == NULL)
00119     return GSASL_MALLOC_ERROR;
00120
00121   _gsasl_hex_encode (in, inlen, *out);
00122   (*out)[len] = '\0';
00123
00124   return GSASL_OK;
00125 }
00126
00142 int
00143 gsasl_hex_from (const char *in, char **out, size_t *outlen)
00144 {
00145   size_t inlen = strlen (in);
00146   size_t l = inlen / 2;
00147
00148   if (inlen % 2 != 0)
00149     return GSASL_BASE64_ERROR;
00150
00151   if (!_gsasl_hex_p (in))
00152     return GSASL_BASE64_ERROR;
00153
00154   *out = malloc (l);
00155   if (!*out)
00156     return GSASL_MALLOC_ERROR;
00157
00158   _gsasl_hex_decode (in, *out);
00159
00160   if (outlen)
00161     *outlen = l;
00162
00163   return GSASL_OK;
00164 }
```

## 5.173 callback.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- void gsasl_callback_set (Gsasl *ctx, Gsasl_callback_function cb)
- int gsasl_callback (Gsasl *ctx, Gsasl_session *sctx, Gsasl_property prop)
- void gsasl_callback_hook_set (Gsasl *ctx, void *hook)
- void * gsasl_callback_hook_get (Gsasl *ctx)
- void gsasl_session_hook_set (Gsasl_session *sctx, void *hook)
- void * gsasl_session_hook_get (Gsasl_session *sctx)

### 5.173.1 Function Documentation

#### 5.173.1.1 gsasl_callback()

```
int gsasl_callback (
            Gsasl * ctx,
            Gsasl_session * sctx,
            Gsasl_property prop )
```

gsasl_callback:

**Parameters**

| | |
|---|---|
| *ctx* | handle received from gsasl_init(), may be NULL to derive it from @sctx. |
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type. |

Invoke the application callback. The @prop value indicate what the callback is expected to do. For example, for GSASL_ANONYMOUS_TOKEN, the function is expected to invoke gsasl_property_set(@SCTX, GSASL_↩ ANONYMOUS_TOKEN, "token") where "token" is the anonymous token the application wishes the SASL mechanism to use. See the manual for the meaning of all parameters.

Return value: Returns whatever the application callback returns, or GSASL_NO_CALLBACK if no application was known.

Since: 0.2.0

Definition at line 70 of file callback.c.

#### 5.173.1.2 gsasl_callback_hook_get()

```
void * gsasl_callback_hook_get (
            Gsasl * ctx )
```

gsasl_callback_hook_get:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |

Retrieve application specific data from libgsasl handle.

The application data is set using gsasl_callback_hook_set(). This is normally used by the application to maintain a global state between the main program and callbacks.

Return value: Returns the application specific data, or NULL.

Since: 0.2.0

Definition at line 119 of file callback.c.

#### 5.173.1.3 gsasl_callback_hook_set()

```
void gsasl_callback_hook_set (
            Gsasl * ctx,
            void * hook )
```

gsasl_callback_hook_set:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *hook* | opaque pointer to application specific data. |

Store application specific data in the libgsasl handle.

The application data can be later (for instance, inside a callback) be retrieved by calling gsasl_callback_hook_get(). This is normally used by the application to maintain a global state between the main program and callbacks.

Since: 0.2.0

Definition at line 99 of file callback.c.

#### 5.173.1.4 gsasl_callback_set()

```
void gsasl_callback_set (
            Gsasl * ctx,
            Gsasl_callback_function cb )
```

gsasl_callback_set:

**Parameters**

| | |
|---|---|
| *ctx* | handle received from gsasl_init(). |
| *cb* | pointer to function implemented by application. |

Store the pointer to the application provided callback in the library handle. The callback will be used, via gsasl_callback(), by mechanisms to discover various parameters (such as username and passwords). The callback function will be called with a Gsasl_property value indicating the requested behaviour. For example, for GSASL_↩ ANONYMOUS_TOKEN, the function is expected to invoke gsasl_property_set(@CTX, GSASL_ANONYMOUS_↩ TOKEN, "token") where "token" is the anonymous token the application wishes the SASL mechanism to use. See the manual for the meaning of all parameters.

Since: 0.2.0

Definition at line 44 of file callback.c.

#### 5.173.1.5 gsasl_session_hook_get()

```
void * gsasl_session_hook_get (
            Gsasl_session * sctx )
```

gsasl_session_hook_get:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |

Retrieve application specific data from libgsasl session handle.

The application data is set using gsasl_callback_hook_set(). This is normally used by the application to maintain a per-session state between the main program and callbacks.

Return value: Returns the application specific data, or NULL.

Since: 0.2.14

Definition at line 159 of file callback.c.

#### 5.173.1.6 gsasl_session_hook_set()

```
void gsasl_session_hook_set (
            Gsasl_session * sctx,
            void * hook )
```

gsasl_session_hook_set:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |
| *hook* | opaque pointer to application specific data. |

Store application specific data in the libgsasl session handle.

The application data can be later (for instance, inside a callback) be retrieved by calling gsasl_session_hook_get(). This is normally used by the application to maintain a per-session state between the main program and callbacks.

Since: 0.2.14

Definition at line 139 of file callback.c.

## 5.174 callback.c

Go to the documentation of this file.

```
00001 /* callback.c --- Callback handling.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00043 void
00044 gsasl_callback_set (Gsasl *ctx, Gsasl_callback_function cb)
00045 {
00046   ctx->cb = cb;
00047 }
00048
00069 int
00070 gsasl_callback (Gsasl *ctx, Gsasl_session *sctx, Gsasl_property prop)
00071 {
00072   if (ctx == NULL && sctx == NULL)
00073     return GSASL_NO_CALLBACK;
00074
00075   if (ctx == NULL)
00076     ctx = sctx->ctx;
00077
00078   if (ctx->cb)
00079     return ctx->cb (ctx, sctx, prop);
00080
00081   return GSASL_NO_CALLBACK;
00082 }
00083
00098 void
00099 gsasl_callback_hook_set (Gsasl *ctx, void *hook)
00100 {
00101   ctx->application_hook = hook;
00102 }
00103
00118 void *
00119 gsasl_callback_hook_get (Gsasl *ctx)
00120 {
00121   return ctx->application_hook;
00122 }
00123
00138 void
00139 gsasl_session_hook_set (Gsasl_session *sctx, void *hook)
00140 {
00141   sctx->application_hook = hook;
00142 }
00143
00158 void *
00159 gsasl_session_hook_get (Gsasl_session *sctx)
00160 {
00161   return sctx->application_hook;
00162 }
```

## 5.175 crypto.c File Reference

```
#include <config.h>
#include "internal.h"
```

```
#include "mechtools.h"
#include "gc.h"
```

**Macros**

- #define CLIENT_KEY "Client Key"
- #define SERVER_KEY "Server Key"

**Functions**

- int gsasl_nonce (char *data, size_t datalen)
- int gsasl_random (char *data, size_t datalen)
- size_t gsasl_hash_length (Gsasl_hash hash)
- int gsasl_scram_secrets_from_salted_password (Gsasl_hash hash, const char *salted_password, char *client_key, char *server_key, char *stored_key)
- int gsasl_scram_secrets_from_password (Gsasl_hash hash, const char *password, unsigned int iteration_count, const char *salt, size_t saltlen, char *salted_password, char *client_key, char *server_key, char *stored_key)

## 5.175.1 Macro Definition Documentation

### 5.175.1.1 CLIENT_KEY

```
#define CLIENT_KEY "Client Key"
```

### 5.175.1.2 SERVER_KEY

```
#define SERVER_KEY "Server Key"
```

## 5.175.2 Function Documentation

### 5.175.2.1 gsasl_hash_length()

```
size_t gsasl_hash_length (
            Gsasl_hash hash )
```

gsasl_hash_length:

**Parameters**

| | |
|---|---|
| *hash* | a Gsasl_hash element, e.g., GSASL_HASH_SHA256. |

Return the digest output size for hash function @hash. For example, gsasl_hash_length(GSASL_HASH_SHA256) returns GSASL_HASH_SHA256_SIZE which is 32.

Returns: size of supplied Gsasl_hash element.

Since: 1.10

Definition at line 72 of file crypto.c.

#### 5.175.2.2 gsasl_nonce()

```
int gsasl_nonce (
             char * data,
             size_t datalen )
```

gsasl_nonce:

**Parameters**

| | |
|---|---|
| *data* | output array to be filled with unpredictable random data. |
| *datalen* | size of output array. |

Store unpredictable data of given size in the provided buffer.

Return value: Returns GSASL_OK iff successful.

Definition at line 38 of file crypto.c.

#### 5.175.2.3 gsasl_random()

```
int gsasl_random (
             char * data,
             size_t datalen )
```

gsasl_random:

**Parameters**

| | |
|---|---|
| *data* | output array to be filled with strong random data. |
| *datalen* | size of output array. |

Store cryptographically strong random data of given size in the provided buffer.

Return value: Returns GSASL_OK iff successful.

Definition at line 54 of file crypto.c.

#### 5.175.2.4 gsasl_scram_secrets_from_password()

```
int gsasl_scram_secrets_from_password (
             Gsasl_hash hash,
             const char * password,
             unsigned int iteration_count,
             const char * salt,
```

```
            size_t saltlen,
            char * salted_password,
            char * client_key,
            char * server_key,
            char * stored_key )
```

gsasl_scram_secrets_from_password:

**Parameters**

| | |
|---|---|
| *hash* | a Gsasl_hash element, e.g., GSASL_HASH_SHA256. |
| *password* | input parameter with password. |
| *iteration_count* | number of PBKDF2 rounds to apply. |
| *salt* | input character array of @saltlen length with salt for PBKDF2. |
| *saltlen* | length of @salt. |
| *salted_password* | pre-allocated output array with derived salted password. |
| *client_key* | pre-allocated output array with derived client key. |
| *server_key* | pre-allocated output array with derived server key. |
| *stored_key* | pre-allocated output array with derived stored key. |

Helper function to generate SCRAM secrets from a password. The @salted_password, @client_key, @server_key, and @stored_key buffers must have room to hold digest for given @hash, use GSASL_HASH_MAX_SIZE which is sufficient for all hashes.

Return value: Returns GSASL_OK if successful, or error code.

Since: 1.10

Definition at line 155 of file crypto.c.

#### 5.175.2.5 gsasl_scram_secrets_from_salted_password()

```
int gsasl_scram_secrets_from_salted_password (
            Gsasl_hash hash,
            const char * salted_password,
            char * client_key,
            char * server_key,
            char * stored_key )
```

gsasl_scram_secrets_from_salted_password:

**Parameters**

| | |
|---|---|
| *hash* | a Gsasl_hash element, e.g., GSASL_HASH_SHA256. |
| *salted_password* | input array with salted password. |
| *client_key* | pre-allocated output array with derived client key. |
| *server_key* | pre-allocated output array with derived server key. |
| *stored_key* | pre-allocated output array with derived stored key. |

Helper function to derive SCRAM ClientKey/ServerKey/StoredKey. The @client_key, @server_key, and @stored↩ _key buffers must have room to hold digest for given @hash, use GSASL_HASH_MAX_SIZE which is sufficient for

all hashes.

Return value: Returns GSASL_OK if successful, or error code.

Since: 1.10

Definition at line 103 of file crypto.c.

## 5.176 crypto.c

Go to the documentation of this file.

```
00001 /* crypto.c --- Simple crypto wrappers for applications.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024 #include "mechtools.h"
00025
00026 #include "gc.h"
00027
00037 int
00038 gsasl_nonce (char *data, size_t datalen)
00039 {
00040   return gc_nonce (data, datalen);
00041 }
00042
00053 int
00054 gsasl_random (char *data, size_t datalen)
00055 {
00056   return gc_random (data, datalen);
00057 }
00058
00071 size_t
00072 gsasl_hash_length (Gsasl_hash hash)
00073 {
00074   switch (hash)
00075     {
00076     case GSASL_HASH_SHA1:
00077       return GSASL_HASH_SHA1_SIZE;
00078     case GSASL_HASH_SHA256:
00079       return GSASL_HASH_SHA256_SIZE;
00080     }
00081
00082   return 0;
00083 }
00084
00102 int
00103 gsasl_scram_secrets_from_salted_password (Gsasl_hash hash,
00104                                           const char *salted_password,
00105                                           char *client_key,
00106                                           char *server_key, char *stored_key)
00107 {
00108   int res;
00109   size_t hashlen = gsasl_hash_length (hash);
00110
00111   /* ClientKey */
00112 #define CLIENT_KEY "Client Key"
00113   res = _gsasl_hmac (hash, salted_password, hashlen,
00114                      CLIENT_KEY, strlen (CLIENT_KEY), client_key);
00115   if (res != GSASL_OK)
```

```
00116     return res;
00117
00118   /* StoredKey */
00119   res = _gsasl_hash (hash, client_key, hashlen, stored_key);
00120   if (res != GSASL_OK)
00121     return res;
00122
00123   /* ServerKey */
00124 #define SERVER_KEY "Server Key"
00125   res = _gsasl_hmac (hash, salted_password, hashlen,
00126                      SERVER_KEY, strlen (SERVER_KEY), server_key);
00127   if (res != GSASL_OK)
00128     return res;
00129
00130   return GSASL_OK;
00131 }
00132
00154 int
00155 gsasl_scram_secrets_from_password (Gsasl_hash hash,
00156                                    const char *password,
00157                                    unsigned int iteration_count,
00158                                    const char *salt,
00159                                    size_t saltlen,
00160                                    char *salted_password,
00161                                    char *client_key,
00162                                    char *server_key, char *stored_key)
00163 {
00164   int res;
00165   char *preppass;
00166
00167   res = gsasl_saslprep (password, GSASL_ALLOW_UNASSIGNED, &preppass, NULL);
00168   if (res != GSASL_OK)
00169     return res;
00170
00171   res = _gsasl_pbkdf2 (hash, preppass, strlen (preppass),
00172                        salt, saltlen, iteration_count, salted_password, 0);
00173   free (preppass);
00174   if (res != GSASL_OK)
00175     return res;
00176
00177   return gsasl_scram_secrets_from_salted_password (hash, salted_password,
00178                                                    client_key, server_key,
00179                                                    stored_key);
00180 }
```

## 5.177 done.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- void gsasl_done (Gsasl *ctx)

### 5.177.1 Function Documentation

#### 5.177.1.1 gsasl_done()

```
void gsasl_done (
            Gsasl * ctx )
```

gsasl_done:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |

This function destroys a libgsasl handle. The handle must not be used with other libgsasl functions after this call.

Definition at line 33 of file done.c.

## 5.178 done.c

Go to the documentation of this file.

```
00001 /* done.c --- Exit point for libgsasl.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00032 void
00033 gsasl_done (Gsasl *ctx)
00034 {
00035   size_t i;
00036
00037   if (ctx == NULL)
00038     return;
00039
00040 #ifdef USE_CLIENT
00041   for (i = 0; i < ctx->n_client_mechs; i++)
00042     if (ctx->client_mechs[i].client.done)
00043       ctx->client_mechs[i].client.done (ctx);
00044
00045   free (ctx->client_mechs);
00046 #endif
00047
00048 #ifdef USE_SERVER
00049   for (i = 0; i < ctx->n_server_mechs; i++)
00050     if (ctx->server_mechs[i].server.done)
00051       ctx->server_mechs[i].server.done (ctx);
00052
00053   free (ctx->server_mechs);
00054 #endif
00055
00056   free (ctx);
00057
00058   return;
00059 }
```

## 5.179 doxygen.c File Reference

## 5.180 doxygen.c

Go to the documentation of this file.

```
00001
```

## 5.181 error.c File Reference

```
#include <config.h>
#include "internal.h"
#include "gettext.h"
```

### Macros

- #define _(String) dgettext (PACKAGE, String)
- #define gettext_noop(String) String
- #define N_(String) gettext_noop (String)
- #define ERR(name, desc) { name, #name, desc }

### Functions

- const char * gsasl_strerror (int err)
- const char * gsasl_strerror_name (int err)

### 5.181.1 Macro Definition Documentation

#### 5.181.1.1 _

```
#define _(
              String ) dgettext (PACKAGE, String)
```

Definition at line 27 of file error.c.

#### 5.181.1.2 ERR

```
#define ERR(
              name,
              desc ) { name, #name, desc }
```

Definition at line 31 of file error.c.

#### 5.181.1.3 gettext_noop

```
#define gettext_noop(
              String ) String
```

Definition at line 28 of file error.c.

#### 5.181.1.4 N_

```
#define N_(
              String ) gettext_noop (String)
```

Definition at line 29 of file error.c.

### 5.181.2 Function Documentation

#### 5.181.2.1 gsasl_strerror()

```
const char * gsasl_strerror (
            int err )
```

gsasl_strerror:

**Parameters**

| | |
|---|---|
| *err* | libgsasl error code |

Convert return code to human readable string explanation of the reason for the particular error code.

This string can be used to output a diagnostic message to the user.

This function is one of few in the library that can be used without a successful call to gsasl_init().

Return value: Returns a pointer to a statically allocated string containing an explanation of the error code @err.

Definition at line 184 of file error.c.

#### 5.181.2.2 gsasl_strerror_name()

```
const char * gsasl_strerror_name (
            int err )
```

gsasl_strerror_name:

**Parameters**

| | |
|---|---|
| *err* | libgsasl error code |

Convert return code to human readable string representing the error code symbol itself. For example, gsasl_↩ strerror_name(GSASL_OK) returns the string "GSASL_OK".

This string can be used to output a diagnostic message to the user.

This function is one of few in the library that can be used without a successful call to gsasl_init().

Return value: Returns a pointer to a statically allocated string containing a string version of the error code @err, or NULL if the error code is not known.

Since: 0.2.29

Definition at line 222 of file error.c.

### 5.181.3 Variable Documentation

#### 5.181.3.1 description

```
const char* description
```

Definition at line 38 of file error.c.

#### 5.181.3.2 name

```
const char* name
```

Definition at line 37 of file error.c.

#### 5.181.3.3 rc

```
int rc
```

Definition at line 36 of file error.c.

## 5.182 error.c

Go to the documentation of this file.

```
00001 /* error.c --- Error handling functionality.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00025 /* I18n of error codes. */
00026 #include "gettext.h"
00027 #define _(String) dgettext (PACKAGE, String)
00028 #define gettext_noop(String) String
00029 #define N_(String) gettext_noop (String)
00030
00031 #define ERR(name, desc) { name, #name, desc }
00032
00033 /* *INDENT-OFF* */
00034 static struct
00035 {
00036   int rc;
00037   const char *name;
00038   const char *description;
00039 } errors[] = {
00040   ERR (GSASL_OK, N_("Libgsasl success")),
00041   ERR (GSASL_NEEDS_MORE, N_("SASL mechanism needs more data")),
00042   ERR (GSASL_UNKNOWN_MECHANISM, N_("Unknown SASL mechanism")),
00043   ERR (GSASL_MECHANISM_CALLED_TOO_MANY_TIMES,
00044        N_("SASL mechanism called too many times")),
00045   { 4, NULL, NULL },
00046   { 5, NULL, NULL },
00047   { 6, NULL, NULL },
00048   ERR (GSASL_MALLOC_ERROR, N_("Memory allocation error in SASL library")),
00049   ERR (GSASL_BASE64_ERROR, N_("Base 64 coding error in SASL library")),
00050   ERR (GSASL_CRYPTO_ERROR, N_("Low-level crypto error in SASL library")),
00051   { 10, NULL, NULL },
00052   { 11, NULL, NULL },
00053   { 12, NULL, NULL },
00054   { 13, NULL, NULL },
00055   { 14, NULL, NULL },
00056   { 15, NULL, NULL },
00057   { 16, NULL, NULL },
00058   { 17, NULL, NULL },
00059   { 18, NULL, NULL },
00060   { 19, NULL, NULL },
00061   { 20, NULL, NULL },
00062   { 21, NULL, NULL },
00063   { 22, NULL, NULL },
00064   { 23, NULL, NULL },
00065   { 24, NULL, NULL },
00066   { 25, NULL, NULL },
00067   { 26, NULL, NULL },
00068   { 27, NULL, NULL },
00069   { 28, NULL, NULL },
```

```
00070   ERR (GSASL_SASLPREP_ERROR,
00071        N_("Could not prepare internationalized (non-ASCII) string.")),
00072   ERR (GSASL_MECHANISM_PARSE_ERROR,
00073        N_("SASL mechanism could not parse input")),
00074   ERR (GSASL_AUTHENTICATION_ERROR, N_("Error authenticating user")),
00075   { 32, NULL, NULL },
00076   ERR (GSASL_INTEGRITY_ERROR, N_("Integrity error in application payload")),
00077   { 34, NULL, NULL },
00078   ERR (GSASL_NO_CLIENT_CODE,
00079        N_("Client-side functionality not available in library "
00080           "(application error)")),
00081   ERR (GSASL_NO_SERVER_CODE,
00082        N_("Server-side functionality not available in library "
00083           "(application error)")),
00084   ERR (GSASL_GSSAPI_RELEASE_BUFFER_ERROR,
00085        N_("GSSAPI library could not deallocate memory in "
00086           "gss_release_buffer() in SASL library.  This is a serious "
00087           "internal error.")),
00088   ERR (GSASL_GSSAPI_IMPORT_NAME_ERROR,
00089        N_("GSSAPI library could not understand a peer name in "
00090           "gss_import_name() in SASL library.  This is most likely due "
00091           "to incorrect service and/or hostnames.")),
00092   ERR (GSASL_GSSAPI_INIT_SEC_CONTEXT_ERROR,
00093        N_("GSSAPI error in client while negotiating security context in "
00094           "gss_init_sec_context() in SASL library.  This is most likely "
00095           "due insufficient credentials or malicious interactions.")),
00096   ERR (GSASL_GSSAPI_ACCEPT_SEC_CONTEXT_ERROR,
00097        N_("GSSAPI error in server while negotiating security context in "
00098           "gss_accept_sec_context() in SASL library.  This is most likely due "
00099           "insufficient credentials or malicious interactions.")),
00100   ERR (GSASL_GSSAPI_UNWRAP_ERROR,
00101        N_("GSSAPI error while decrypting or decoding data in gss_unwrap() in "
00102           "SASL library.  This is most likely due to data corruption.")),
00103   ERR (GSASL_GSSAPI_WRAP_ERROR,
00104        N_("GSSAPI error while encrypting or encoding data in gss_wrap() in "
00105           "SASL library.")),
00106   ERR (GSASL_GSSAPI_ACQUIRE_CRED_ERROR,
00107        N_("GSSAPI error acquiring credentials in gss_acquire_cred() in "
00108           "SASL library.  This is most likely due to not having the proper "
00109           "Kerberos key available in /etc/krb5.keytab on the server.")),
00110   ERR (GSASL_GSSAPI_DISPLAY_NAME_ERROR,
00111        N_("GSSAPI error creating a display name denoting the client in "
00112           "gss_display_name() in SASL library.  This is probably because "
00113           "the client supplied bad data.")),
00114   ERR (GSASL_GSSAPI_UNSUPPORTED_PROTECTION_ERROR,
00115        N_("Other entity requested integrity or confidentiality protection "
00116           "in GSSAPI mechanism but this is currently not implemented.")),
00117   { 46, NULL, NULL },
00118   { 47, NULL, NULL },
00119   ERR (GSASL_SECURID_SERVER_NEED_ADDITIONAL_PASSCODE,
00120        N_("SecurID needs additional passcode.")),
00121   ERR (GSASL_SECURID_SERVER_NEED_NEW_PIN,
00122        N_("SecurID needs new pin.")),
00123   { 50, NULL, NULL },
00124   ERR (GSASL_NO_CALLBACK,
00125        N_("No callback specified by caller (application error).")),
00126   ERR (GSASL_NO_ANONYMOUS_TOKEN,
00127        N_("Authentication failed because the anonymous token was "
00128           "not provided.")),
00129   ERR (GSASL_NO_AUTHID,
00130        N_("Authentication failed because the authentication identity was "
00131           "not provided.")),
00132   ERR (GSASL_NO_AUTHZID,
00133        N_("Authentication failed because the authorization identity was "
00134           "not provided.")),
00135   ERR (GSASL_NO_PASSWORD,
00136        N_("Authentication failed because the password was not provided.")),
00137   ERR (GSASL_NO_PASSCODE,
00138        N_("Authentication failed because the passcode was not provided.")),
00139   ERR (GSASL_NO_PIN,
00140        N_("Authentication failed because the pin code was not provided.")),
00141   ERR (GSASL_NO_SERVICE,
00142        N_("Authentication failed because the service name was not provided.")),
00143   ERR (GSASL_NO_HOSTNAME,
00144        N_("Authentication failed because the host name was not provided.")),
00145   ERR (GSASL_GSSAPI_ENCAPSULATE_TOKEN_ERROR,
00146        N_("GSSAPI error encapsulating token.")),
00147   ERR (GSASL_GSSAPI_DECAPSULATE_TOKEN_ERROR,
00148        N_("GSSAPI error decapsulating token.")),
00149   ERR (GSASL_GSSAPI_INQUIRE_MECH_FOR_SASLNAME_ERROR,
00150        N_("GSSAPI error getting OID for SASL mechanism name.")),
00151   ERR (GSASL_GSSAPI_TEST_OID_SET_MEMBER_ERROR,
00152        N_("GSSAPI error testing for OID in OID set.")),
00153   ERR (GSASL_GSSAPI_RELEASE_OID_SET_ERROR,
00154        N_("GSSAPI error releasing OID set.")),
00155   ERR (GSASL_NO_CB_TLS_UNIQUE,
00156        N_("Authentication failed because a tls-unique CB was not provided.")),
```

```
00157   ERR (GSASL_NO_SAML20_IDP_IDENTIFIER,
00158        N_("Callback failed to provide SAML20 IdP identifier.")),
00159   ERR (GSASL_NO_SAML20_REDIRECT_URL,
00160        N_("Callback failed to provide SAML20 redirect URL.")),
00161   ERR (GSASL_NO_OPENID20_REDIRECT_URL,
00162        N_("Callback failed to provide OPENID20 redirect URL.")),
00163   ERR (GSASL_NO_CB_TLS_EXPORTER,
00164        N_("Authentication failed because a tls-exporter channel binding was not provided."))
00165 };
00166 /* *INDENT-ON* */
00167
00183 const char *
00184 gsasl_strerror (int err)
00185 {
00186   static const char *unknown = N_("Libgsasl unknown error");
00187   const char *p;
00188
00189   bindtextdomain (PACKAGE, LOCALEDIR);
00190
00191   if (err < 0 || err >= (int) (sizeof (errors) / sizeof (errors[0])))
00192     return _(unknown);
00193
00194   p = errors[err].description;
00195   if (!p)
00196     p = unknown;
00197
00198   return _(p);
00199 }
00200
00201
00221 const char *
00222 gsasl_strerror_name (int err)
00223 {
00224   if (err < 0 || err >= (int) (sizeof (errors) / sizeof (errors[0])))
00225     return NULL;
00226
00227   return errors[err].name;
00228 }
```

## 5.183 digest-md5/free.c File Reference

```
#include <config.h>
#include "free.h"
#include <stdlib.h>
#include <string.h>
```

### Functions

- void digest_md5_free_challenge (digest_md5_challenge *c)
- void digest_md5_free_response (digest_md5_response *r)
- void digest_md5_free_finish (digest_md5_finish *f)

### 5.183.1 Function Documentation

#### 5.183.1.1 digest_md5_free_challenge()

```
void digest_md5_free_challenge (
            digest_md5_challenge * c )
```

Definition at line 34 of file digest-md5/free.c.

#### 5.183.1.2 digest_md5_free_finish()

```
void digest_md5_free_finish (
            digest_md5_finish * f )
```

Definition at line 60 of file digest-md5/free.c.

#### 5.183.1.3 digest_md5_free_response()

```
void digest_md5_free_response (
            digest_md5_response * r )
```

Definition at line 47 of file digest-md5/free.c.

## 5.184 digest-md5/free.c

Go to the documentation of this file.

```
00001 /* free.h --- Free allocated data in DIGEST-MD5 token structures.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get prototypes. */
00025 #include "free.h"
00026
00027 /* Get free. */
00028 #include <stdlib.h>
00029
00030 /* Get memset. */
00031 #include <string.h>
00032
00033 void
00034 digest_md5_free_challenge (digest_md5_challenge *c)
00035 {
00036   size_t i;
00037
00038   for (i = 0; i < c->nrealms; i++)
00039     free (c->realms[i]);
00040   free (c->realms);
00041   free (c->nonce);
00042
00043   memset (c, 0, sizeof (*c));
00044 }
00045
00046 void
00047 digest_md5_free_response (digest_md5_response *r)
00048 {
00049   free (r->username);
00050   free (r->realm);
00051   free (r->nonce);
00052   free (r->cnonce);
00053   free (r->digesturi);
00054   free (r->authzid);
00055
```

```
00056   memset (r, 0, sizeof (*r));
00057 }
00058
00059 void
00060 digest_md5_free_finish (digest_md5_finish *f)
00061 {
00062   memset (f, 0, sizeof (*f));
00063 }
```

## 5.185 src/free.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- void gsasl_free (void *ptr)

### 5.185.1 Function Documentation

#### 5.185.1.1 gsasl_free()

```
void gsasl_free (
            void * ptr )
```

gsasl_free:

**Parameters**

| | |
|---|---|
| *ptr* | memory pointer |

Invoke free(@ptr) to de-allocate memory pointer. Typically used on strings allocated by other libgsasl functions.

This is useful on Windows where libgsasl is linked to one CRT and the application is linked to another CRT. Then malloc/free will not use the same heap. This happens if you build libgsasl using mingw32 and the application with Visual Studio.

Since: 0.2.19

Definition at line 40 of file src/free.c.

## 5.186 src/free.c

Go to the documentation of this file.

```
00001 /* free.c --- Wrapper around the `free' function, primarily for Windows
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
```

```
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00039 void
00040 gsasl_free (void *ptr)
00041 {
00042   free (ptr);
00043 }
```

# 5.187 gsasl-mech.h File Reference

### Data Structures

- struct Gsasl_mechanism_functions
- struct Gsasl_mechanism

### Typedefs

- typedef int(* Gsasl_init_function) (Gsasl *ctx)
- typedef void(* Gsasl_done_function) (Gsasl *ctx)
- typedef int(* Gsasl_start_function) (Gsasl_session *sctx, void **mech_data)
- typedef int(* Gsasl_step_function) (Gsasl_session *sctx, void *mech_data, const char *input, size_t input↩ _len, char **output, size_t *output_len)
- typedef void(* Gsasl_finish_function) (Gsasl_session *sctx, void *mech_data)
- typedef int(* Gsasl_code_function) (Gsasl_session *sctx, void *mech_data, const char *input, size_t input↩ _len, char **output, size_t *output_len)
- typedef struct Gsasl_mechanism_functions Gsasl_mechanism_functions
- typedef struct Gsasl_mechanism Gsasl_mechanism

### Functions

- _GSASL_API int gsasl_register (Gsasl *ctx, const Gsasl_mechanism *mech)

## 5.187.1 Typedef Documentation

### 5.187.1.1 Gsasl_code_function

```
typedef int(* Gsasl_code_function) (Gsasl_session *sctx, void *mech_data, const char *input,
size_t input_len, char **output, size_t *output_len)
```

Gsasl_code_function:

**Parameters**

| | |
|---|---|
| *sctx* | a Gsasl_session session handle. |
| *mech_data* | pointer to void* with mechanism-specific data. |
| *input* | input byte array. |
| *input_len* | size of input byte array. |
| *output* | newly allocated output byte array. |
| *output_len* | pointer to output variable with size of output byte array. |

The implementation of this function should perform data encoding or decoding for the mechanism, after authentication has completed. This might mean that data is integrity or privacy protected.

The @output buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl↩ _free(@output).

Return value: Returns GSASL_OK if encoding was successful, otherwise an error code.

Definition at line 133 of file gsasl-mech.h.

#### 5.187.1.2 Gsasl_done_function

```
typedef void(* Gsasl_done_function) (Gsasl *ctx)
```

Gsasl_done_function:

**Parameters**

| | |
|---|---|
| *ctx* | a Gsasl libgsasl handle. |

The implementation of this function pointer deallocate all resources associated with the mechanism.

Definition at line 57 of file gsasl-mech.h.

#### 5.187.1.3 Gsasl_finish_function

```
typedef void(* Gsasl_finish_function) (Gsasl_session *sctx, void *mech_data)
```

Gsasl_finish_function:

**Parameters**

| | |
|---|---|
| *sctx* | a Gsasl_session session handle. |
| *mech_data* | pointer to void* with mechanism-specific data. |

The implementation of this function should release all resources associated with the particular authentication process.

Definition at line 111 of file gsasl-mech.h.

#### 5.187.1.4 Gsasl_init_function

```
typedef int(* Gsasl_init_function) (Gsasl *ctx)
```

SECTION:gsasl-mech

**Parameters**

| | |
|---|---|
| *title* | gsasl-mech.h |
| *short_description* | register new application-defined mechanism |

The builtin mechanisms should suffice for most applications. Applications can register a new mechanism in the library using application-supplied functions. The mechanism will operate as the builtin mechanisms, and the supplied functions will be invoked when necessary. The application uses the normal logic, e.g., calls gsasl_client_start() followed by a sequence of calls to gsasl_step() and finally gsasl_finish(). Gsasl_init_function:

**Parameters**

| | |
|---|---|
| *ctx* | a Gsasl libgsasl handle. |

The implementation of this function pointer should fail if the mechanism for some reason is not available for further use.

Return value: Returns GSASL_OK iff successful.

Definition at line 48 of file gsasl-mech.h.

#### 5.187.1.5 Gsasl_mechanism

```
typedef struct Gsasl_mechanism Gsasl_mechanism
```

Definition at line 177 of file gsasl-mech.h.

#### 5.187.1.6 Gsasl_mechanism_functions

```
typedef struct Gsasl_mechanism_functions Gsasl_mechanism_functions
```

Definition at line 160 of file gsasl-mech.h.

#### 5.187.1.7 Gsasl_start_function

```
typedef int(* Gsasl_start_function) (Gsasl_session *sctx, void **mech_data)
```

Gsasl_start_function:

**Parameters**

| | |
|---|---|
| *sctx* | a Gsasl_session session handle. |
| *mech_data* | pointer to void* with mechanism-specific data. |

The implementation of this function should start a new authentication process.

Return value: Returns GSASL_OK iff successful.

Definition at line 69 of file gsasl-mech.h.

#### 5.187.1.8 Gsasl_step_function

```
typedef int(* Gsasl_step_function) (Gsasl_session *sctx, void *mech_data, const char *input,
size_t input_len, char **output, size_t *output_len)
```

Gsasl_step_function:

**Parameters**

| | |
|---|---|
| *sctx* | a Gsasl_session session handle. |
| *mech_data* | pointer to void* with mechanism-specific data. |
| *input* | input byte array. |
| *input_len* | size of input byte array. |
| *output* | newly allocated output byte array. |
| *output_len* | pointer to output variable with size of output byte array. |

The implementation of this function should perform one step of the authentication process.

This reads data from the other end (from @input and @input_len), processes it (potentially invoking callbacks to the application), and writes data to server (into newly allocated variable @output and @output_len that indicate the length of @output).

The contents of the @output buffer is unspecified if this functions returns anything other than GSASL_OK or GSASL_NEEDS_MORE. If this function return GSASL_OK or GSASL_NEEDS_MORE, however, the @output buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl_free(@output).

Return value: Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

Definition at line 99 of file gsasl-mech.h.

### 5.187.2 Function Documentation

#### 5.187.2.1 gsasl_register()

```
_GSASL_API int gsasl_register (
            Gsasl * ctx,
            const Gsasl_mechanism * mech )  [extern]
```

gsasl_register:

**Parameters**

| | |
|---|---|
| *ctx* | pointer to libgsasl handle. |
| *mech* | plugin structure with information about plugin. |

This function initialize given mechanism, and if successful, add it to the list of plugins that is used by the library.

Return value: GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Since: 0.2.0

Definition at line 38 of file register.c.

## 5.188 gsasl-mech.h

Go to the documentation of this file.

```
00001 /* gsasl-mech.h --- Header file for mechanism handling in GNU SASL Library.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef GSASL_MECH_H
00023 # define GSASL_MECH_H
00024
00048 typedef int (*Gsasl_init_function) (Gsasl * ctx);
00049
00057 typedef void (*Gsasl_done_function) (Gsasl * ctx);
00058
00069 typedef int (*Gsasl_start_function) (Gsasl_session * sctx, void **mech_data);
00070
00099 typedef int (*Gsasl_step_function) (Gsasl_session * sctx, void *mech_data,
00100                                     const char *input, size_t input_len,
00101                                     char **output, size_t *output_len);
00102
00111 typedef void (*Gsasl_finish_function) (Gsasl_session * sctx, void *mech_data);
00112
00133 typedef int (*Gsasl_code_function) (Gsasl_session * sctx, void *mech_data,
00134                                     const char *input, size_t input_len,
00135                                     char **output, size_t *output_len);
00136
00150 struct Gsasl_mechanism_functions
00151 {
00152   Gsasl_init_function init;
00153   Gsasl_done_function done;
00154   Gsasl_start_function start;
00155   Gsasl_step_function step;
00156   Gsasl_finish_function finish;
00157   Gsasl_code_function encode;
00158   Gsasl_code_function decode;
00159 };
00160 typedef struct Gsasl_mechanism_functions Gsasl_mechanism_functions;
00161
00170 struct Gsasl_mechanism
00171 {
00172   const char *name;
00173
00174   struct Gsasl_mechanism_functions client;
00175   struct Gsasl_mechanism_functions server;
00176 };
00177 typedef struct Gsasl_mechanism Gsasl_mechanism;
00178
00179 /* Register new mechanism: register.c. */
00180 extern _GSASL_API int gsasl_register (Gsasl * ctx,
00181                                       const Gsasl_mechanism * mech);
00182
00183 #endif /* GSASL_MECH_H */
```

## 5.189 gsasl-version.h File Reference

### Macros

- #define GSASL_VERSION "2.2.3"
- #define GSASL_VERSION_MAJOR 2
- #define GSASL_VERSION_MINOR 2
- #define GSASL_VERSION_PATCH 3
- #define GSASL_VERSION_NUMBER 0x020203

### 5.189.1 Macro Definition Documentation

#### 5.189.1.1 GSASL_VERSION

```
#define GSASL_VERSION "2.2.3"
```

SECTION:gsasl-version

**Parameters**

| | |
|---|---|
| *title* | gsasl-version.h |
| *short_description* | version symbols |

The gsasl-version.h file contains version symbols. It should not be included directly, only via gsasl.h. GSASL_↩ VERSION

Pre-processor symbol with a string that describe the header file version number. Used together with gsasl_check_version() to verify header file and run-time library consistency.

Definition at line 41 of file gsasl-version.h.

#### 5.189.1.2 GSASL_VERSION_MAJOR

```
#define GSASL_VERSION_MAJOR 2
```

GSASL_VERSION_MAJOR

Pre-processor symbol with a decimal value that describe the major level of the header file version number. For example, when the header version is 1.2.3 this symbol will be 1.

Since: 1.1

Definition at line 52 of file gsasl-version.h.

#### 5.189.1.3 GSASL_VERSION_MINOR

```
#define GSASL_VERSION_MINOR 2
```

GSASL_VERSION_MINOR

Pre-processor symbol with a decimal value that describe the minor level of the header file version number. For example, when the header version is 1.2.3 this symbol will be 2.

Since: 1.1

Definition at line 63 of file gsasl-version.h.

#### 5.189.1.4 GSASL_VERSION_NUMBER

```
#define GSASL_VERSION_NUMBER 0x020203
```

GSASL_VERSION_NUMBER

Pre-processor symbol with a hexadecimal value describing the header file version number. For example, when the header version is 1.2.3 this symbol will have the value 0x010203.

Since: 1.1

Definition at line 85 of file gsasl-version.h.

#### 5.189.1.5 GSASL_VERSION_PATCH

```
#define GSASL_VERSION_PATCH 3
```

GSASL_VERSION_PATCH

Pre-processor symbol with a decimal value that describe the patch level of the header file version number. For example, when the header version is 1.2.3 this symbol will be 3.

Since: 1.1

Definition at line 74 of file gsasl-version.h.

## 5.190 gsasl-version.h

Go to the documentation of this file.

```
00001 /* gsasl-version.h --- Header file for GNU SASL Library version symbols.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef GSASL_VERSION_H
00023 # define GSASL_VERSION_H
00024
00041 # define GSASL_VERSION "2.2.3"
00042
00052 # define GSASL_VERSION_MAJOR 2
00053
00063 # define GSASL_VERSION_MINOR 2
00064
00074 # define GSASL_VERSION_PATCH 3
00075
00085 # define GSASL_VERSION_NUMBER 0x020203
00086
00087 #endif /* GSASL_VERSION_H */
```

## 5.191 gsasl.h File Reference

```
#include <stdio.h>
#include <stddef.h>
#include <sys/types.h>
#include <gsasl-version.h>
#include <gsasl-mech.h>
```

### Macros

- #define _GSASL_API

### Typedefs

- typedef struct Gsasl Gsasl
- typedef struct Gsasl_session Gsasl_session
- typedef int(* Gsasl_callback_function) (Gsasl *ctx, Gsasl_session *sctx, Gsasl_property prop)

### Enumerations

- enum Gsasl_rc {
  GSASL_OK = 0 , GSASL_NEEDS_MORE = 1 , GSASL_UNKNOWN_MECHANISM = 2 , GSASL_MECHANISM_CALLED_TOO
  = 3 ,
  GSASL_MALLOC_ERROR = 7 , GSASL_BASE64_ERROR = 8 , GSASL_CRYPTO_ERROR = 9 ,
  GSASL_SASLPREP_ERROR = 29 ,
  GSASL_MECHANISM_PARSE_ERROR = 30 , GSASL_AUTHENTICATION_ERROR = 31 , GSASL_INTEGRITY_ERROR
  = 33 , GSASL_NO_CLIENT_CODE = 35 ,
  GSASL_NO_SERVER_CODE = 36 , GSASL_NO_CALLBACK = 51 , GSASL_NO_ANONYMOUS_TOKEN
  = 52 , GSASL_NO_AUTHID = 53 ,
  GSASL_NO_AUTHZID = 54 , GSASL_NO_PASSWORD = 55 , GSASL_NO_PASSCODE = 56 ,
  GSASL_NO_PIN = 57 ,
  GSASL_NO_SERVICE = 58 , GSASL_NO_HOSTNAME = 59 , GSASL_NO_CB_TLS_UNIQUE = 65 ,
  GSASL_NO_SAML20_IDP_IDENTIFIER = 66 ,
  GSASL_NO_SAML20_REDIRECT_URL = 67 , GSASL_NO_OPENID20_REDIRECT_URL = 68 ,
  GSASL_NO_CB_TLS_EXPORTER = 69 , GSASL_GSSAPI_RELEASE_BUFFER_ERROR = 37 ,
  GSASL_GSSAPI_IMPORT_NAME_ERROR = 38 , GSASL_GSSAPI_INIT_SEC_CONTEXT_ERROR = 39 ,
  GSASL_GSSAPI_ACCEPT_SEC_CONTEXT_ERROR = 40 , GSASL_GSSAPI_UNWRAP_ERROR = 41 ,
  GSASL_GSSAPI_WRAP_ERROR = 42 , GSASL_GSSAPI_ACQUIRE_CRED_ERROR = 43 , GSASL_GSSAPI_DISPLAY_NA
  = 44 , GSASL_GSSAPI_UNSUPPORTED_PROTECTION_ERROR = 45 ,
  GSASL_SECURID_SERVER_NEED_ADDITIONAL_PASSCODE = 48 , GSASL_SECURID_SERVER_NEED_NEW_PIN
  = 49 , GSASL_GSSAPI_ENCAPSULATE_TOKEN_ERROR = 60 , GSASL_GSSAPI_DECAPSULATE_TOKEN_ERROR
  = 61 ,
  GSASL_GSSAPI_INQUIRE_MECH_FOR_SASLNAME_ERROR = 62 , GSASL_GSSAPI_TEST_OID_SET_MEMBER_ERRO
  = 63 , GSASL_GSSAPI_RELEASE_OID_SET_ERROR = 64 }
- enum Gsasl_property {
  GSASL_AUTHID = 1 , GSASL_AUTHZID = 2 , GSASL_PASSWORD = 3 , GSASL_ANONYMOUS_TOKEN
  = 4 ,
  GSASL_SERVICE = 5 , GSASL_HOSTNAME = 6 , GSASL_GSSAPI_DISPLAY_NAME = 7 ,
  GSASL_PASSCODE = 8 ,
  GSASL_SUGGESTED_PIN = 9 , GSASL_PIN = 10 , GSASL_REALM = 11 , GSASL_DIGEST_MD5_HASHED_PASSWORD
  = 12 ,
  GSASL_QOPS = 13 , GSASL_QOP = 14 , GSASL_SCRAM_ITER = 15 , GSASL_SCRAM_SALT = 16 ,
  GSASL_SCRAM_SALTED_PASSWORD = 17 , GSASL_SCRAM_SERVERKEY = 23 , GSASL_SCRAM_STOREDKEY

= 24 , GSASL_CB_TLS_UNIQUE = 18 ,
GSASL_SAML20_IDP_IDENTIFIER = 19 , GSASL_SAML20_REDIRECT_URL = 20 , GSASL_OPENID20_REDIRECT_URL = 21 , GSASL_OPENID20_OUTCOME_DATA = 22 ,
GSASL_CB_TLS_EXPORTER = 25 , GSASL_SAML20_AUTHENTICATE_IN_BROWSER = 250 , GSASL_OPENID20_AUTHENTICATE_IN_BROWSER = 251 , GSASL_VALIDATE_SIMPLE = 500 ,
GSASL_VALIDATE_EXTERNAL = 501 , GSASL_VALIDATE_ANONYMOUS = 502 , GSASL_VALIDATE_GSSAPI = 503 , GSASL_VALIDATE_SECURID = 504 ,
GSASL_VALIDATE_SAML20 = 505 , GSASL_VALIDATE_OPENID20 = 506 }

- enum Gsasl_mechname_limits { GSASL_MIN_MECHANISM_SIZE = 1 , GSASL_MAX_MECHANISM_SIZE = 20 }
- enum Gsasl_qop { GSASL_QOP_AUTH = 1 , GSASL_QOP_AUTH_INT = 2 , GSASL_QOP_AUTH_CONF = 4 }
- enum Gsasl_saslprep_flags { GSASL_ALLOW_UNASSIGNED = 1 }
- enum Gsasl_hash { GSASL_HASH_SHA1 = 2 , GSASL_HASH_SHA256 = 3 }
- enum Gsasl_hash_length { GSASL_HASH_SHA1_SIZE = 20 , GSASL_HASH_SHA256_SIZE = 32 , GSASL_HASH_MAX_SIZE = GSASL_HASH_SHA256_SIZE }

## Functions

- _GSASL_API int gsasl_init (Gsasl **ctx)
- _GSASL_API void gsasl_done (Gsasl *ctx)
- _GSASL_API const char * gsasl_check_version (const char *req_version)
- _GSASL_API void gsasl_callback_set (Gsasl *ctx, Gsasl_callback_function cb)
- _GSASL_API int gsasl_callback (Gsasl *ctx, Gsasl_session *sctx, Gsasl_property prop)
- _GSASL_API void gsasl_callback_hook_set (Gsasl *ctx, void *hook)
- _GSASL_API void * gsasl_callback_hook_get (Gsasl *ctx)
- _GSASL_API void gsasl_session_hook_set (Gsasl_session *sctx, void *hook)
- _GSASL_API void * gsasl_session_hook_get (Gsasl_session *sctx)
- _GSASL_API int gsasl_property_set (Gsasl_session *sctx, Gsasl_property prop, const char *data)
- _GSASL_API int gsasl_property_set_raw (Gsasl_session *sctx, Gsasl_property prop, const char *data, size_t len)
- _GSASL_API void gsasl_property_free (Gsasl_session *sctx, Gsasl_property prop)
- _GSASL_API const char * gsasl_property_get (Gsasl_session *sctx, Gsasl_property prop)
- _GSASL_API const char * gsasl_property_fast (Gsasl_session *sctx, Gsasl_property prop)
- _GSASL_API int gsasl_client_mechlist (Gsasl *ctx, char **out)
- _GSASL_API int gsasl_client_support_p (Gsasl *ctx, const char *name)
- _GSASL_API const char * gsasl_client_suggest_mechanism (Gsasl *ctx, const char *mechlist)
- _GSASL_API int gsasl_server_mechlist (Gsasl *ctx, char **out)
- _GSASL_API int gsasl_server_support_p (Gsasl *ctx, const char *name)
- _GSASL_API int gsasl_mechanism_name_p (const char *mech)
- _GSASL_API int gsasl_client_start (Gsasl *ctx, const char *mech, Gsasl_session **sctx)
- _GSASL_API int gsasl_server_start (Gsasl *ctx, const char *mech, Gsasl_session **sctx)
- _GSASL_API int gsasl_step (Gsasl_session *sctx, const char *input, size_t input_len, char **output, size_t *output_len)
- _GSASL_API int gsasl_step64 (Gsasl_session *sctx, const char *b64input, char **b64output)
- _GSASL_API void gsasl_finish (Gsasl_session *sctx)
- _GSASL_API int gsasl_encode (Gsasl_session *sctx, const char *input, size_t input_len, char **output, size_t *output_len)
- _GSASL_API int gsasl_decode (Gsasl_session *sctx, const char *input, size_t input_len, char **output, size_t *output_len)
- _GSASL_API const char * gsasl_mechanism_name (Gsasl_session *sctx)
- _GSASL_API const char * gsasl_strerror (int err)
- _GSASL_API const char * gsasl_strerror_name (int err)
- _GSASL_API int gsasl_saslprep (const char *in, Gsasl_saslprep_flags flags, char **out, int *stringpreprc)
- _GSASL_API int gsasl_nonce (char *data, size_t datalen)

- _GSASL_API int gsasl_random (char *data, size_t datalen)
- _GSASL_API size_t gsasl_hash_length (Gsasl_hash hash)
- _GSASL_API int gsasl_scram_secrets_from_salted_password (Gsasl_hash hash, const char *salted_↩ password, char *client_key, char *server_key, char *stored_key)
- _GSASL_API int gsasl_scram_secrets_from_password (Gsasl_hash hash, const char *password, unsigned int iteration_count, const char *salt, size_t saltlen, char *salted_password, char *client_key, char *server_↩ key, char *stored_key)
- _GSASL_API int gsasl_simple_getpass (const char *filename, const char *username, char **key)
- _GSASL_API int gsasl_base64_to (const char *in, size_t inlen, char **out, size_t *outlen)
- _GSASL_API int gsasl_base64_from (const char *in, size_t inlen, char **out, size_t *outlen)
- _GSASL_API int gsasl_hex_to (const char *in, size_t inlen, char **out, size_t *outlen)
- _GSASL_API int gsasl_hex_from (const char *in, char **out, size_t *outlen)
- _GSASL_API void gsasl_free (void *ptr)

## 5.191.1 Macro Definition Documentation

### 5.191.1.1 _GSASL_API

```
#define _GSASL_API
```

SECTION:gsasl

**Parameters**

| | |
|---|---|
| *title* | gsasl.h |
| *short_description* | main library interfaces |

The main library interfaces are declared in gsasl.h.

Definition at line 48 of file gsasl.h.

## 5.191.2 Typedef Documentation

### 5.191.2.1 Gsasl

```
typedef struct Gsasl Gsasl
```

Gsasl:

Handle to global library context.

Definition at line 62 of file gsasl.h.

### 5.191.2.2 Gsasl_callback_function

```
typedef int(* Gsasl_callback_function) (Gsasl *ctx, Gsasl_session *sctx, Gsasl_property prop)
```

Gsasl_callback_function:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *sctx* | session handle, may be NULL. |
| *prop* | enumerated value of Gsasl_property type. |

Prototype of function that the application should implement. Use gsasl_callback_set() to inform the library about your callback function.

It is called by the SASL library when it need some information from the application. Depending on the value of @prop, it should either set some property (e.g., username or password) using gsasl_property_set(), or it should extract some properties (e.g., authentication and authorization identities) using gsasl_property_fast() and use them to make a policy decision, perhaps returning GSASL_AUTHENTICATION_ERROR or GSASL_OK depending on whether the policy permitted the operation.

Return value: Any valid return code, the interpretation of which depend on the @prop value.

Since: 0.2.0

Definition at line 285 of file gsasl.h.

#### 5.191.2.3 Gsasl_session

```
typedef struct Gsasl_session Gsasl_session
```

Gsasl_session:

Handle to SASL session context.

Definition at line 69 of file gsasl.h.

### 5.191.3 Enumeration Type Documentation

#### 5.191.3.1 Gsasl_hash

```
enum Gsasl_hash
```

Gsasl_hash:

**Parameters**

| | |
|---|---|
| *GSASL_HASH_SHA1* | Hash function SHA-1. |
| *GSASL_HASH_SHA256* | Hash function SHA-256. |

Hash functions. You may use gsasl_hash_length() to get the output size of a hash function.

Currently only used as parameter to gsasl_scram_secrets_from_salted_password() and gsasl_scram_secrets_from_password() to specify for which SCRAM mechanism to prepare secrets for.

Since: 1.10

**Enumerator**

| | |
|---|---|
| GSASL_HASH_SHA1 | |
| GSASL_HASH_SHA256 | |

Definition at line 426 of file gsasl.h.

#### 5.191.3.2 Gsasl_hash_length

`enum Gsasl_hash_length`

Gsasl_hash_length:

**Parameters**

| | |
|---|---|
| *GSASL_HASH_SHA1_SIZE* | Output size of hash function SHA-1. |
| *GSASL_HASH_SHA256_SIZE* | Output size of hash function SHA-256. |
| *GSASL_HASH_MAX_SIZE* | Maximum output size of any Gsasl_hash_length. |

Identifiers specifying the output size of hash functions.

These can be used when statically allocating the buffers needed for, e.g., gsasl_scram_secrets_from_password().

Since: 1.10

**Enumerator**

| | |
|---|---|
| GSASL_HASH_SHA1_SIZE | |
| GSASL_HASH_SHA256_SIZE | |
| GSASL_HASH_MAX_SIZE | |

Definition at line 446 of file gsasl.h.

#### 5.191.3.3 Gsasl_mechname_limits

`enum Gsasl_mechname_limits`

Gsasl_mechname_limits:

**Parameters**

| | |
|---|---|
| *GSASL_MIN_MECHANISM_SIZE* | Minimum size of mechanism name strings. |
| *GSASL_MAX_MECHANISM_SIZE* | Maximum size of mechanism name strings. |

SASL mechanisms are named by strings, from 1 to 20 characters in length, consisting of upper-case letters, digits, hyphens, and/or underscores. See also gsasl_mechanism_name_p().

**Enumerator**

| | |
|---|---|
| GSASL_MIN_MECHANISM_SIZE | |
| GSASL_MAX_MECHANISM_SIZE | |

Definition at line 297 of file gsasl.h.

#### 5.191.3.4 Gsasl_property

```
enum Gsasl_property
```

Gsasl_property:

**Parameters**

| | |
|---|---|
| *GSASL_AUTHID* | Authentication identity (username). |
| *GSASL_AUTHZID* | Authorization identity. |
| *GSASL_PASSWORD* | Password. |
| *GSASL_ANONYMOUS_TOKEN* | Anonymous identifier. |
| *GSASL_SERVICE* | Service name |
| *GSASL_HOSTNAME* | Host name. |
| *GSASL_GSSAPI_DISPLAY_NAME* | GSS-API credential principal name. |
| *GSASL_PASSCODE* | SecurID passcode. |
| *GSASL_SUGGESTED_PIN* | SecurID suggested PIN. |
| *GSASL_PIN* | SecurID PIN. |
| *GSASL_REALM* | User realm. |
| *GSASL_DIGEST_MD5_HASHED_PASSWORD* | Pre-computed hashed DIGEST-MD5 password, to avoid storing passwords in the clear. |
| *GSASL_QOPS* | Set of quality-of-protection values. |
| *GSASL_QOP* | Quality-of-protection value. |
| *GSASL_SCRAM_ITER* | Number of iterations in password-to-key hashing. |
| *GSASL_SCRAM_SALT* | Salt for password-to-key hashing. |
| *GSASL_SCRAM_SALTED_PASSWORD* | Hex-encoded hashed/salted password. |
| *GSASL_SCRAM_SERVERKEY* | Hex-encoded SCRAM ServerKey derived from users' password. |
| *GSASL_SCRAM_STOREDKEY* | Hex-encoded SCRAM StoredKey derived from users' password. |
| *GSASL_CB_TLS_UNIQUE* | Base64 encoded tls-unique channel binding. |
| *GSASL_CB_TLS_EXPORTER* | Base64 encoded tls-exporter channel binding. |
| *GSASL_SAML20_IDP_IDENTIFIER* | SAML20 user IdP URL. |
| *GSASL_SAML20_REDIRECT_URL* | SAML 2.0 URL to access in browser. |
| *GSASL_OPENID20_REDIRECT_URL* | OpenID 2.0 URL to access in browser. |
| *GSASL_OPENID20_OUTCOME_DATA* | OpenID 2.0 authentication outcome data. |
| *GSASL_SAML20_AUTHENTICATE_IN_BROWSER* | Request to perform SAML 2.0 authentication in browser. |
| *GSASL_OPENID20_AUTHENTICATE_IN_↩ BROWSER* | Request to perform OpenID 2.0 authentication in browser. |
| *GSASL_VALIDATE_SIMPLE* | Request for simple validation. |
| *GSASL_VALIDATE_EXTERNAL* | Request for validation of EXTERNAL. |
| *GSASL_VALIDATE_ANONYMOUS* | Request for validation of ANONYMOUS. |

**Parameters**

| | |
|---|---|
| *GSASL_VALIDATE_GSSAPI* | Request for validation of GSSAPI/GS2. |
| *GSASL_VALIDATE_SECURID* | Request for validation of SecurID. |
| *GSASL_VALIDATE_SAML20* | Request for validation of SAML20. |
| *GSASL_VALIDATE_OPENID20* | Request for validation of OpenID 2.0 login. |

Callback/property types.

**Enumerator**

| | |
|---|---|
| GSASL_AUTHID | |
| GSASL_AUTHZID | |
| GSASL_PASSWORD | |
| GSASL_ANONYMOUS_TOKEN | |
| GSASL_SERVICE | |
| GSASL_HOSTNAME | |
| GSASL_GSSAPI_DISPLAY_NAME | |
| GSASL_PASSCODE | |
| GSASL_SUGGESTED_PIN | |
| GSASL_PIN | |
| GSASL_REALM | |
| GSASL_DIGEST_MD5_HASHED_PASSWORD | |
| GSASL_QOPS | |
| GSASL_QOP | |
| GSASL_SCRAM_ITER | |
| GSASL_SCRAM_SALT | |
| GSASL_SCRAM_SALTED_PASSWORD | |
| GSASL_SCRAM_SERVERKEY | |
| GSASL_SCRAM_STOREDKEY | |
| GSASL_CB_TLS_UNIQUE | |
| GSASL_SAML20_IDP_IDENTIFIER | |
| GSASL_SAML20_REDIRECT_URL | |
| GSASL_OPENID20_REDIRECT_URL | |
| GSASL_OPENID20_OUTCOME_DATA | |
| GSASL_CB_TLS_EXPORTER | |
| GSASL_SAML20_AUTHENTICATE_IN_BROWSER | |
| GSASL_OPENID20_AUTHENTICATE_IN_BROWSER | |
| GSASL_VALIDATE_SIMPLE | |
| GSASL_VALIDATE_EXTERNAL | |
| GSASL_VALIDATE_ANONYMOUS | |
| GSASL_VALIDATE_GSSAPI | |
| GSASL_VALIDATE_SECURID | |
| GSASL_VALIDATE_SAML20 | |
| GSASL_VALIDATE_OPENID20 | |

Definition at line 220 of file gsasl.h.

#### 5.191.3.5 Gsasl_qop

```
enum Gsasl_qop
```

Gsasl_qop:

**Parameters**

| | |
|---|---|
| *GSASL_QOP_AUTH* | Authentication only. |
| *GSASL_QOP_AUTH_INT* | Authentication and integrity. |
| *GSASL_QOP_AUTH_CONF* | Authentication, integrity and confidentiality. |

Quality of Protection types (DIGEST-MD5 and GSSAPI). The integrity and confidentiality values is about application data wrapping. We recommend that you use @GSASL_QOP_AUTH with TLS as that combination is generally more secure and have better chance of working than the integrity/confidentiality layers of SASL.

**Enumerator**

| | |
|---|---|
| GSASL_QOP_AUTH | |
| GSASL_QOP_AUTH_INT | |
| GSASL_QOP_AUTH_CONF | |

Definition at line 315 of file gsasl.h.

#### 5.191.3.6 Gsasl_rc

```
enum Gsasl_rc
```

Gsasl_rc:

**Parameters**

| | |
|---|---|
| *GSASL_OK* | Successful return code, guaranteed to be always 0. |
| *GSASL_NEEDS_MORE* | Mechanism expects another round-trip. |
| *GSASL_UNKNOWN_MECHANISM* | Application requested an unknown mechanism. |
| *GSASL_MECHANISM_CALLED_TOO_MANY_↩ TIMES* | Application requested too many round trips from mechanism. |
| *GSASL_MALLOC_ERROR* | Memory allocation failed. |
| *GSASL_BASE64_ERROR* | Base64 encoding/decoding failed. |
| *GSASL_CRYPTO_ERROR* | Cryptographic error. |
| *GSASL_SASLPREP_ERROR* | Failed to prepare internationalized string. |
| *GSASL_MECHANISM_PARSE_ERROR* | Mechanism could not parse input. |
| *GSASL_AUTHENTICATION_ERROR* | Authentication has failed. |
| *GSASL_INTEGRITY_ERROR* | Application data integrity check failed. |
| *GSASL_NO_CLIENT_CODE* | Library was built with client functionality. |
| *GSASL_NO_SERVER_CODE* | Library was built with server functionality. |
| *GSASL_NO_CALLBACK* | Application did not provide a callback. |
| *GSASL_NO_ANONYMOUS_TOKEN* | Could not get required anonymous token. |
| *GSASL_NO_AUTHID* | Could not get required authentication identity (username). |

**Parameters**

| | |
|---|---|
| *GSASL_NO_AUTHZID* | Could not get required authorization identity. |
| *GSASL_NO_PASSWORD* | Could not get required password. |
| *GSASL_NO_PASSCODE* | Could not get required SecurID PIN. |
| *GSASL_NO_PIN* | Could not get required SecurID PIN. |
| *GSASL_NO_SERVICE* | Could not get required service name. |
| *GSASL_NO_HOSTNAME* | Could not get required hostname. |
| *GSASL_NO_CB_TLS_UNIQUE* | Could not get required tls-unique CB. |
| *GSASL_NO_CB_TLS_EXPORTER* | Could not get required tls-exporter CB. |
| *GSASL_NO_SAML20_IDP_IDENTIFIER* | Could not get required SAML IdP. |
| *GSASL_NO_SAML20_REDIRECT_URL* | Could not get required SAML redirect URL. |
| *GSASL_NO_OPENID20_REDIRECT_URL* | Could not get required OpenID redirect URL. |
| *GSASL_GSSAPI_RELEASE_BUFFER_ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_IMPORT_NAME_ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_INIT_SEC_CONTEXT_ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_ACCEPT_SEC_CONTEXT_↩ ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_UNWRAP_ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_WRAP_ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_ACQUIRE_CRED_ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_DISPLAY_NAME_ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_UNSUPPORTED_PROTECTION↩ _ERROR* | An unsupported quality-of-protection layer was requested. |
| *GSASL_GSSAPI_ENCAPSULATE_TOKEN_ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_DECAPSULATE_TOKEN_ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_INQUIRE_MECH_FOR_↩ SASLNAME_ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_TEST_OID_SET_MEMBER_↩ ERROR* | GSS-API library call error. |
| *GSASL_GSSAPI_RELEASE_OID_SET_ERROR* | GSS-API library call error. |
| *GSASL_SECURID_SERVER_NEED_↩ ADDITIONAL_PASSCODE* | SecurID mechanism needs an additional passcode. |
| *GSASL_SECURID_SERVER_NEED_NEW_PIN* | SecurID mechanism needs an new PIN. |

Error codes for library functions.

**Enumerator**

| | |
|---|---|
| GSASL_OK | |
| GSASL_NEEDS_MORE | |
| GSASL_UNKNOWN_MECHANISM | |
| GSASL_MECHANISM_CALLED_TOO_MANY_TIMES | |
| GSASL_MALLOC_ERROR | |
| GSASL_BASE64_ERROR | |
| GSASL_CRYPTO_ERROR | |
| GSASL_SASLPREP_ERROR | |
| GSASL_MECHANISM_PARSE_ERROR | |
| GSASL_AUTHENTICATION_ERROR | |
| GSASL_INTEGRITY_ERROR | |
| GSASL_NO_CLIENT_CODE | |

**Enumerator**

| | |
|---|---|
| GSASL_NO_SERVER_CODE | |
| GSASL_NO_CALLBACK | |
| GSASL_NO_ANONYMOUS_TOKEN | |
| GSASL_NO_AUTHID | |
| GSASL_NO_AUTHZID | |
| GSASL_NO_PASSWORD | |
| GSASL_NO_PASSCODE | |
| GSASL_NO_PIN | |
| GSASL_NO_SERVICE | |
| GSASL_NO_HOSTNAME | |
| GSASL_NO_CB_TLS_UNIQUE | |
| GSASL_NO_SAML20_IDP_IDENTIFIER | |
| GSASL_NO_SAML20_REDIRECT_URL | |
| GSASL_NO_OPENID20_REDIRECT_URL | |
| GSASL_NO_CB_TLS_EXPORTER | |
| GSASL_GSSAPI_RELEASE_BUFFER_ERROR | |
| GSASL_GSSAPI_IMPORT_NAME_ERROR | |
| GSASL_GSSAPI_INIT_SEC_CONTEXT_ERROR | |
| GSASL_GSSAPI_ACCEPT_SEC_CONTEXT_ERROR | |
| GSASL_GSSAPI_UNWRAP_ERROR | |
| GSASL_GSSAPI_WRAP_ERROR | |
| GSASL_GSSAPI_ACQUIRE_CRED_ERROR | |
| GSASL_GSSAPI_DISPLAY_NAME_ERROR | |
| GSASL_GSSAPI_UNSUPPORTED_PROTECTION_ERROR | |
| GSASL_SECURID_SERVER_NEED_ADDITIONAL_PASSCODE | |
| GSASL_SECURID_SERVER_NEED_NEW_PIN | |
| GSASL_GSSAPI_ENCAPSULATE_TOKEN_ERROR | |
| GSASL_GSSAPI_DECAPSULATE_TOKEN_ERROR | |
| GSASL_GSSAPI_INQUIRE_MECH_FOR_SASLNAME_ERROR | |
| GSASL_GSSAPI_TEST_OID_SET_MEMBER_ERROR | |
| GSASL_GSSAPI_RELEASE_OID_SET_ERROR | |

Definition at line 126 of file gsasl.h.

#### 5.191.3.7 Gsasl_saslprep_flags

```
enum Gsasl_saslprep_flags
```

Gsasl_saslprep_flags:

**Parameters**

| | |
|---|---|
| *GSASL_ALLOW_UNASSIGNED* | Allow unassigned code points. |

Flags for the SASLprep function, see gsasl_saslprep(). For background, see the GNU Libidn documentation.

**Enumerator**

| | |
|---|---|
| GSASL_ALLOW_UNASSIGNED | |

Definition at line 329 of file gsasl.h.

## 5.191.4 Function Documentation

### 5.191.4.1 gsasl_base64_from()

```
_GSASL_API int gsasl_base64_from (
             const char * in,
             size_t inlen,
             char ** out,
             size_t * outlen )  [extern]
```

gsasl_base64_from:

**Parameters**

| | |
|---|---|
| *in* | input byte array |
| *inlen* | size of input byte array |
| *out* | pointer to newly allocated output byte array |
| *outlen* | pointer to size of newly allocated output byte array |

Decode Base64 data. The @out buffer must be deallocated by the caller.

Return value: Returns GSASL_OK on success, GSASL_BASE64_ERROR if input was invalid, and GSASL_↩ MALLOC_ERROR on memory allocation errors.

Since: 0.2.2

Definition at line 74 of file base64.c.

### 5.191.4.2 gsasl_base64_to()

```
_GSASL_API int gsasl_base64_to (
             const char * in,
             size_t inlen,
             char ** out,
             size_t * outlen )  [extern]
```

gsasl_base64_to:

**Parameters**

| | |
|---|---|
| *in* | input byte array. |
| *inlen* | size of input byte array. |
| *out* | pointer to newly allocated base64-encoded string. |
| *outlen* | pointer to size of newly allocated base64-encoded string. |

Encode data as base64. The @out string is zero terminated, and @outlen holds the length excluding the terminating zero. The @out buffer must be deallocated by the caller.

Return value: Returns GSASL_OK on success, or GSASL_MALLOC_ERROR if input was too large or memory allocation fail.

Since: 0.2.2

Definition at line 44 of file base64.c.

#### 5.191.4.3 gsasl_callback()

```
_GSASL_API int gsasl_callback (
             Gsasl * ctx,
             Gsasl_session * sctx,
             Gsasl_property prop )  [extern]
```

gsasl_callback:

**Parameters**

| | |
|---|---|
| *ctx* | handle received from gsasl_init(), may be NULL to derive it from @sctx. |
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type. |

Invoke the application callback. The @prop value indicate what the callback is expected to do. For example, for GSASL_ANONYMOUS_TOKEN, the function is expected to invoke gsasl_property_set(@SCTX, GSASL_↩ ANONYMOUS_TOKEN, "token") where "token" is the anonymous token the application wishes the SASL mechanism to use. See the manual for the meaning of all parameters.

Return value: Returns whatever the application callback returns, or GSASL_NO_CALLBACK if no application was known.

Since: 0.2.0

Definition at line 70 of file callback.c.

#### 5.191.4.4 gsasl_callback_hook_get()

```
_GSASL_API void * gsasl_callback_hook_get (
             Gsasl * ctx )  [extern]
```

gsasl_callback_hook_get:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |

Retrieve application specific data from libgsasl handle.

The application data is set using gsasl_callback_hook_set(). This is normally used by the application to maintain a global state between the main program and callbacks.

Return value: Returns the application specific data, or NULL.

Since: 0.2.0

Definition at line 119 of file callback.c.

#### 5.191.4.5 gsasl_callback_hook_set()

```
_GSASL_API void gsasl_callback_hook_set (
            Gsasl * ctx,
            void * hook )  [extern]
```

gsasl_callback_hook_set:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *hook* | opaque pointer to application specific data. |

Store application specific data in the libgsasl handle.

The application data can be later (for instance, inside a callback) be retrieved by calling gsasl_callback_hook_get(). This is normally used by the application to maintain a global state between the main program and callbacks.

Since: 0.2.0

Definition at line 99 of file callback.c.

#### 5.191.4.6 gsasl_callback_set()

```
_GSASL_API void gsasl_callback_set (
            Gsasl * ctx,
            Gsasl_callback_function cb )  [extern]
```

gsasl_callback_set:

**Parameters**

| | |
|---|---|
| *ctx* | handle received from gsasl_init(). |
| *cb* | pointer to function implemented by application. |

Store the pointer to the application provided callback in the library handle. The callback will be used, via gsasl_callback(), by mechanisms to discover various parameters (such as username and passwords). The callback function will be called with a Gsasl_property value indicating the requested behaviour. For example, for GSASL_↩ ANONYMOUS_TOKEN, the function is expected to invoke gsasl_property_set(@CTX, GSASL_ANONYMOUS_↩ TOKEN, "token") where "token" is the anonymous token the application wishes the SASL mechanism to use. See the manual for the meaning of all parameters.

Since: 0.2.0

Definition at line 44 of file callback.c.

#### 5.191.4.7 gasl_check_version()

```
_GSASL_API const char * gsasl_check_version (
             const char * req_version )  [extern]
```

gsasl_check_version:

**Parameters**

| | |
|---|---|
| *req_version* | version string to compare with, or NULL. |

Check GNU SASL Library version.

See GSASL_VERSION for a suitable @req_version string.

This function is one of few in the library that can be used without a successful call to gsasl_init().

Return value: Check that the version of the library is at minimum the one given as a string in @req_version and return the actual version string of the library; return NULL if the condition is not met. If NULL is passed to this function no check is done and only the version string is returned.

Definition at line 45 of file version.c.

#### 5.191.4.8 gasl_client_mechlist()

```
_GSASL_API int gsasl_client_mechlist (
             Gsasl * ctx,
             char ** out )  [extern]
```

gsasl_client_mechlist:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *out* | newly allocated output character array. |

Return a newly allocated string containing SASL names, separated by space, of mechanisms supported by the libgsasl client. @out is allocated by this function, and it is the responsibility of caller to deallocate it.

Return value: Returns GSASL_OK if successful, or error code.

Definition at line 74 of file listmech.c.

#### 5.191.4.9 gasl_client_start()

```
_GSASL_API int gsasl_client_start (
             Gsasl * ctx,
             const char * mech,
             Gsasl_session ** sctx )  [extern]
```

gsasl_client_start:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *mech* | name of SASL mechanism. |
| *sctx* | pointer to client handle. |

This functions initiates a client SASL authentication. This function must be called before any other gsasl_client_*() function is called.

Return value: Returns GSASL_OK if successful, or error code.

Definition at line 119 of file xstart.c.

#### 5.191.4.10 gsasl_client_suggest_mechanism()

```
_GSASL_API const char * gsasl_client_suggest_mechanism (
             Gsasl * ctx,
             const char * mechlist )  [extern]
```

gsasl_client_suggest_mechanism:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *mechlist* | input character array with SASL mechanism names, separated by invalid characters (e.g. SPC). |

Given a list of mechanisms, suggest which to use.

Return value: Returns name of "best" SASL mechanism supported by the libgsasl client which is present in the input string, or NULL if no supported mechanism is found.

Definition at line 87 of file suggest.c.

#### 5.191.4.11 gsasl_client_support_p()

```
_GSASL_API int gsasl_client_support_p (
             Gsasl * ctx,
             const char * name )  [extern]
```

gsasl_client_support_p:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *name* | name of SASL mechanism. |

Decide whether there is client-side support for a specified mechanism.

Return value: Returns 1 if the libgsasl client supports the named mechanism, otherwise 0.

Definition at line 49 of file supportp.c.

#### 5.191.4.12 gsasl_decode()

```
_GSASL_API int gsasl_decode (
             Gsasl_session * sctx,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

gsasl_decode:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |
| *input* | input byte array. |
| *input_len* | size of input byte array. |
| *output* | newly allocated output byte array. |
| *output_len* | pointer to output variable with size of output byte array. |

Decode data according to negotiated SASL mechanism. This might mean that data is integrity or privacy protected.

The @output buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl↩ _free(@output).

Return value: Returns GSASL_OK if encoding was successful, otherwise an error code.

Definition at line 98 of file xcode.c.

#### 5.191.4.13 gsasl_done()

```
_GSASL_API void gsasl_done (
             Gsasl * ctx )  [extern]
```

gsasl_done:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |

This function destroys a libgsasl handle. The handle must not be used with other libgsasl functions after this call.

Definition at line 33 of file done.c.

#### 5.191.4.14 gsasl_encode()

```
_GSASL_API int gsasl_encode (
             Gsasl_session * sctx,
             const char * input,
             size_t input_len,
```

```
            char ** output,
            size_t * output_len )  [extern]
```

gsasl_encode:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |
| *input* | input byte array. |
| *input_len* | size of input byte array. |
| *output* | newly allocated output byte array. |
| *output_len* | pointer to output variable with size of output byte array. |

Encode data according to negotiated SASL mechanism. This might mean that data is integrity or privacy protected.

The @output buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl↩ _free(@output).

Return value: Returns GSASL_OK if encoding was successful, otherwise an error code.

Definition at line 65 of file xcode.c.

#### 5.191.4.15 gsasl_finish()

```
_GSASL_API void gsasl_finish (
             Gsasl_session * sctx )  [extern]
```

gsasl_finish:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |

Destroy a libgsasl client or server handle. The handle must not be used with other libgsasl functions after this call.

Definition at line 33 of file xfinish.c.

#### 5.191.4.16 gsasl_free()

```
_GSASL_API void gsasl_free (
             void * ptr )  [extern]
```

gsasl_free:

**Parameters**

| | |
|---|---|
| *ptr* | memory pointer |

Invoke free(@ptr) to de-allocate memory pointer. Typically used on strings allocated by other libgsasl functions.

This is useful on Windows where libgsasl is linked to one CRT and the application is linked to another CRT. Then malloc/free will not use the same heap. This happens if you build libgsasl using mingw32 and the application with Visual Studio.

Since: 0.2.19

Definition at line 40 of file src/free.c.

#### 5.191.4.17 gsasl_hash_length()

```
_GSASL_API size_t gsasl_hash_length (
            Gsasl_hash hash )  [extern]
```

gsasl_hash_length:

**Parameters**

| | |
|---|---|
| *hash* | a Gsasl_hash element, e.g., GSASL_HASH_SHA256. |

Return the digest output size for hash function @hash. For example, gsasl_hash_length(GSASL_HASH_SHA256) returns GSASL_HASH_SHA256_SIZE which is 32.

Returns: size of supplied Gsasl_hash element.

Since: 1.10

Definition at line 72 of file crypto.c.

#### 5.191.4.18 gsasl_hex_from()

```
_GSASL_API int gsasl_hex_from (
            const char * in,
            char ** out,
            size_t * outlen )  [extern]
```

gsasl_hex_from:

**Parameters**

| | |
|---|---|
| *in* | input byte array |
| *out* | pointer to newly allocated output byte array |
| *outlen* | pointer to size of newly allocated output byte array |

Decode hex data. The @out buffer must be deallocated by the caller.

Return value: Returns GSASL_OK on success, GSASL_BASE64_ERROR if input was invalid, and GSASL_↩ MALLOC_ERROR on memory allocation errors.

Since: 1.10

Definition at line 143 of file base64.c.

#### 5.191.4.19 gsasl_hex_to()

```
_GSASL_API int gsasl_hex_to (
            const char * in,
            size_t inlen,
            char ** out,
            size_t * outlen )  [extern]
```

gsasl_hex_to:

**Parameters**

| | |
|---|---|
| *in* | input byte array. |
| *inlen* | size of input byte array. |
| *out* | pointer to newly allocated hex-encoded string. |
| *outlen* | pointer to size of newly allocated hex-encoded string. |

Hex encode data. The @out string is zero terminated, and @outlen holds the length excluding the terminating zero. The @out buffer must be deallocated by the caller.

Return value: Returns GSASL_OK on success, or GSASL_MALLOC_ERROR if input was too large or memory allocation fail.

Since: 1.10

Definition at line 110 of file base64.c.

#### 5.191.4.20 gsasl_init()

```
_GSASL_API int gsasl_init (
             Gsasl ** ctx )  [extern]
```

gsasl_init:

**Parameters**

| | |
|---|---|
| *ctx* | pointer to libgsasl handle. |

This functions initializes libgsasl. The handle pointed to by ctx is valid for use with other libgsasl functions iff this function is successful. It also register all builtin SASL mechanisms, using gsasl_register().

Return value: GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Definition at line 157 of file init.c.

#### 5.191.4.21 gsasl_mechanism_name()

```
_GSASL_API const char * gsasl_mechanism_name (
             Gsasl_session * sctx )  [extern]
```

gsasl_mechanism_name:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |

This function returns the name of the SASL mechanism used in the session. The pointer must not be deallocated by the caller.

Return value: Returns a zero terminated character array with the name of the SASL mechanism, or NULL if not known.

Since: 0.2.28

Definition at line 38 of file mechname.c.

#### 5.191.4.22 gsasl_mechanism_name_p()

```
_GSASL_API int gsasl_mechanism_name_p (
             const char * mech )  [extern]
```

gsasl_mechanism_name_p:

**Parameters**

| | |
|---|---|
| *mech* | input variable with mechanism name string. |

Check if the mechanism name string @mech follows syntactical rules. It does not check that the name is registered with IANA. It does not check that the mechanism name is actually implemented and supported.

SASL mechanisms are named by strings, from 1 to 20 characters in length, consisting of upper-case letters, digits, hyphens, and/or underscores.

Returns: non-zero when mechanism name string @mech conforms to rules, zero when it does not meet the requirements.

Since: 2.0.0

Definition at line 52 of file suggest.c.

#### 5.191.4.23 gsasl_nonce()

```
_GSASL_API int gsasl_nonce (
             char * data,
             size_t datalen )  [extern]
```

gsasl_nonce:

**Parameters**

| | |
|---|---|
| *data* | output array to be filled with unpredictable random data. |
| *datalen* | size of output array. |

Store unpredictable data of given size in the provided buffer.

Return value: Returns GSASL_OK iff successful.

Definition at line 38 of file crypto.c.

#### 5.191.4.24 gsasl_property_fast()

```
_GSASL_API const char * gsasl_property_fast (
             Gsasl_session * sctx,
             Gsasl_property prop )  [extern]
```

gsasl_property_fast:

**Parameters**

| | |
|---|---|
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type, indicating the type of data in @data. |

Retrieve the data stored in the session handle for given property @prop.

The pointer is to live data, and must not be deallocated or modified in any way.

This function will not invoke the application callback.

Return value: Return property value, if known, or NULL if no value known.

Since: 0.2.0

Definition at line 261 of file property.c.

#### 5.191.4.25 gsasl_property_free()

```
_GSASL_API void gsasl_property_free (
             Gsasl_session * sctx,
             Gsasl_property prop )  [extern]
```

gsasl_property_free:

**Parameters**

| | |
|---|---|
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type to clear |

Deallocate associated data with property @prop in session handle. After this call, gsasl_property_fast(@sctx, @prop) will always return NULL.

Since: 2.0.0

Definition at line 158 of file property.c.

#### 5.191.4.26 gsasl_property_get()

```
_GSASL_API const char * gsasl_property_get (
             Gsasl_session * sctx,
             Gsasl_property prop )  [extern]
```

gsasl_property_get:

**Parameters**

| | |
|---|---|
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type, indicating the type of data in @data. |

Retrieve the data stored in the session handle for given property @prop, possibly invoking the application callback to get the value.

The pointer is to live data, and must not be deallocated or modified in any way.

This function will invoke the application callback, using gsasl_callback(), when a property value is not known.

Return value: Return data for property, or NULL if no value known.

Since: 0.2.0

Definition at line 291 of file property.c.

#### 5.191.4.27 gsasl_property_set()

```
_GSASL_API int gsasl_property_set (
             Gsasl_session * sctx,
             Gsasl_property prop,
             const char * data )  [extern]
```

gsasl_property_set:

**Parameters**

| | |
|---|---|
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type, indicating the type of data in @data. |
| *data* | zero terminated character string to store. |

Make a copy of @data and store it in the session handle for the indicated property @prop.

You can immediately deallocate @data after calling this function, without affecting the data stored in the session handle.

Return value: GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Since: 0.2.0

Definition at line 188 of file property.c.

#### 5.191.4.28 gsasl_property_set_raw()

```
_GSASL_API int gsasl_property_set_raw (
             Gsasl_session * sctx,
             Gsasl_property prop,
             const char * data,
             size_t len )  [extern]
```

gsasl_property_set_raw:

**Parameters**

| | |
|---|---|
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type, indicating the type of data in @data. |
| *data* | character string to store. |
| *len* | length of character string to store. |

Make a copy of @len sized @data and store a zero terminated version of it in the session handle for the indicated property @prop.

You can immediately deallocate @data after calling this function, without affecting the data stored in the session handle.

Except for the length indicator, this function is identical to gsasl_property_set.

Return value: GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Since: 0.2.0

Definition at line 217 of file property.c.

#### 5.191.4.29 gsasl_random()

```
_GSASL_API int gsasl_random (
             char * data,
             size_t datalen )  [extern]
```

gsasl_random:

**Parameters**

| | |
|---|---|
| *data* | output array to be filled with strong random data. |
| *datalen* | size of output array. |

Store cryptographically strong random data of given size in the provided buffer.

Return value: Returns GSASL_OK iff successful.

Definition at line 54 of file crypto.c.

#### 5.191.4.30 gsasl_saslprep()

```
_GSASL_API int gsasl_saslprep (
             const char * in,
             Gsasl_saslprep_flags flags,
             char ** out,
             int * stringpreprc )  [extern]
```

#### 5.191.4.31 gsasl_scram_secrets_from_password()

```
_GSASL_API int gsasl_scram_secrets_from_password (
             Gsasl_hash hash,
             const char * password,
             unsigned int iteration_count,
             const char * salt,
             size_t saltlen,
             char * salted_password,
             char * client_key,
             char * server_key,
             char * stored_key )  [extern]
```

gsasl_scram_secrets_from_password:

**Parameters**

| | |
|---|---|
| *hash* | a Gsasl_hash element, e.g., GSASL_HASH_SHA256. |
| *password* | input parameter with password. |
| *iteration_count* | number of PBKDF2 rounds to apply. |
| *salt* | input character array of @saltlen length with salt for PBKDF2. |
| *saltlen* | length of @salt. |
| *salted_password* | pre-allocated output array with derived salted password. |
| *client_key* | pre-allocated output array with derived client key. |
| *server_key* | pre-allocated output array with derived server key. |
| *stored_key* | pre-allocated output array with derived stored key. |

Helper function to generate SCRAM secrets from a password. The @salted_password, @client_key, @server_key, and @stored_key buffers must have room to hold digest for given @hash, use GSASL_HASH_MAX_SIZE which is sufficient for all hashes.

Return value: Returns GSASL_OK if successful, or error code.

Since: 1.10

Definition at line 155 of file crypto.c.

#### 5.191.4.32 gsasl_scram_secrets_from_salted_password()

```
_GSASL_API int gsasl_scram_secrets_from_salted_password (
             Gsasl_hash hash,
             const char * salted_password,
             char * client_key,
             char * server_key,
             char * stored_key )  [extern]
```

gsasl_scram_secrets_from_salted_password:

**Parameters**

| | |
|---|---|
| *hash* | a Gsasl_hash element, e.g., GSASL_HASH_SHA256. |
| *salted_password* | input array with salted password. |
| *client_key* | pre-allocated output array with derived client key. |
| *server_key* | pre-allocated output array with derived server key. |
| *stored_key* | pre-allocated output array with derived stored key. |

Helper function to derive SCRAM ClientKey/ServerKey/StoredKey. The @client_key, @server_key, and @stored↩ _key buffers must have room to hold digest for given @hash, use GSASL_HASH_MAX_SIZE which is sufficient for all hashes.

Return value: Returns GSASL_OK if successful, or error code.

Since: 1.10

Definition at line 103 of file crypto.c.

#### 5.191.4.33 gsasl_server_mechlist()

```
_GSASL_API int gsasl_server_mechlist (
             Gsasl * ctx,
             char ** out )  [extern]
```

gsasl_server_mechlist:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *out* | newly allocated output character array. |

Return a newly allocated string containing SASL names, separated by space, of mechanisms supported by the libgsasl server. @out is allocated by this function, and it is the responsibility of caller to deallocate it.

Return value: Returns GSASL_OK if successful, or error code.

Definition at line 93 of file listmech.c.

#### 5.191.4.34 gsasl_server_start()

```
_GSASL_API int gsasl_server_start (
             Gsasl * ctx,
             const char * mech,
             Gsasl_session ** sctx )  [extern]
```

gsasl_server_start:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *mech* | name of SASL mechanism. |
| *sctx* | pointer to server handle. |

This functions initiates a server SASL authentication. This function must be called before any other gsasl_server_∗() function is called.

Return value: Returns GSASL_OK if successful, or error code.

Definition at line 137 of file xstart.c.

#### 5.191.4.35 gsasl_server_support_p()

```
_GSASL_API int gsasl_server_support_p (
             Gsasl * ctx,
             const char * name )  [extern]
```

gsasl_server_support_p:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *name* | name of SASL mechanism. |

Decide whether there is server-side support for a specified mechanism.

Return value: Returns 1 if the libgsasl server supports the named mechanism, otherwise 0.

Definition at line 66 of file supportp.c.

#### 5.191.4.36 gsasl_session_hook_get()

```
_GSASL_API void * gsasl_session_hook_get (
             Gsasl_session * sctx )  [extern]
```

gsasl_session_hook_get:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |

Retrieve application specific data from libgsasl session handle.

The application data is set using gsasl_callback_hook_set(). This is normally used by the application to maintain a per-session state between the main program and callbacks.

Return value: Returns the application specific data, or NULL.

Since: 0.2.14

Definition at line 159 of file callback.c.

#### 5.191.4.37 gsasl_session_hook_set()

```
_GSASL_API void gsasl_session_hook_set (
             Gsasl_session * sctx,
             void * hook )  [extern]
```

gsasl_session_hook_set:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |
| *hook* | opaque pointer to application specific data. |

Store application specific data in the libgsasl session handle.

The application data can be later (for instance, inside a callback) be retrieved by calling gsasl_session_hook_get(). This is normally used by the application to maintain a per-session state between the main program and callbacks.

Since: 0.2.14

Definition at line 139 of file callback.c.

#### 5.191.4.38 gsasl_simple_getpass()

```
_GSASL_API int gsasl_simple_getpass (
             const char * filename,
             const char * username,
             char ** key )  [extern]
```

gsasl_simple_getpass:

**Parameters**

| | |
|---|---|
| *filename* | filename of file containing passwords. |
| *username* | username string. |
| *key* | newly allocated output character array. |

Retrieve password for user from specified file. The buffer @key contain the password if this function is successful. The caller is responsible for deallocating it.

The file should be on the UoW "MD5 Based Authentication" format, which means it is in text format with comments denoted by # first on the line, with user entries looking as "usernameTABpassword". This function removes CR and LF at the end of lines before processing. TAB, CR, and LF denote ASCII values 9, 13, and 10, respectively.

Return value: Return GSASL_OK if output buffer contains the password, GSASL_AUTHENTICATION_ERROR if the user could not be found, or other error code.

Definition at line 47 of file md5pwd.c.

#### 5.191.4.39 gsasl_step()

```
_GSASL_API int gsasl_step (
             Gsasl_session * sctx,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )  [extern]
```

gsasl_step:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |
| *input* | input byte array. |
| *input_len* | size of input byte array. |
| *output* | newly allocated output byte array. |
| *output_len* | pointer to output variable with size of output byte array. |

Perform one step of SASL authentication. This reads data from the other end (from @input and @input_len), processes it (potentially invoking callbacks to the application), and writes data to server (into newly allocated variable @output and @output_len that indicate the length of @output).

The contents of the @output buffer is unspecified if this functions returns anything other than GSASL_OK or GSASL_NEEDS_MORE. If this function return GSASL_OK or GSASL_NEEDS_MORE, however, the @output buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl_free(@output).

Return value: Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

Definition at line 51 of file xstep.c.

#### 5.191.4.40 gsasl_step64()

```
_GSASL_API int gsasl_step64 (
             Gsasl_session * sctx,
             const char * b64input,
             char ** b64output )  [extern]
```

gsasl_step64:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl client handle. |
| *b64input* | input base64 encoded byte array. |
| *b64output* | newly allocated output base64 encoded byte array. |

This is a simple wrapper around gsasl_step() that base64 decodes the input and base64 encodes the output.

The contents of the @b64output buffer is unspecified if this functions returns anything other than GSASL↩ _OK or GSASL_NEEDS_MORE. If this function return GSASL_OK or GSASL_NEEDS_MORE, however, the @b64output buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl↩ _free(@b64output).

Return value: Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

Definition at line 86 of file xstep.c.

#### 5.191.4.41 gsasl_strerror()

```
_GSASL_API const char * gsasl_strerror (
             int err )  [extern]
```

gsasl_strerror:

**Parameters**

| | |
|---|---|
| *err* | libgsasl error code |

Convert return code to human readable string explanation of the reason for the particular error code.

This string can be used to output a diagnostic message to the user.

This function is one of few in the library that can be used without a successful call to gsasl_init().

Return value: Returns a pointer to a statically allocated string containing an explanation of the error code @err.

Definition at line 184 of file error.c.

#### 5.191.4.42 gsasl_strerror_name()

```
_GSASL_API const char * gsasl_strerror_name (
            int err )  [extern]
```

gsasl_strerror_name:

**Parameters**

| | |
|---|---|
| *err* | libgsasl error code |

Convert return code to human readable string representing the error code symbol itself. For example, gsasl_↩ strerror_name(GSASL_OK) returns the string "GSASL_OK".

This string can be used to output a diagnostic message to the user.

This function is one of few in the library that can be used without a successful call to gsasl_init().

Return value: Returns a pointer to a statically allocated string containing a string version of the error code @err, or NULL if the error code is not known.

Since: 0.2.29

Definition at line 222 of file error.c.

## 5.192 gsasl.h

Go to the documentation of this file.

```
00001 /* gsasl.h --- Header file for GNU SASL Library.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
```

```
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef GSASL_H
00023 # define GSASL_H
00024
00033 # include <stdio.h>             /* FILE */
00034 # include <stddef.h>            /* size_t */
00035 # include <sys/types.h>         /* ssize_t */
00036
00037 /* Get version symbols. */
00038 # include <gsasl-version.h>
00039
00040 # ifndef _GSASL_API
00041 #  if defined GSASL_BUILDING && defined HAVE_VISIBILITY && HAVE_VISIBILITY
00042 #   define _GSASL_API __attribute__((__visibility__("default")))
00043 #  elif defined GSASL_BUILDING && defined _MSC_VER && ! defined GSASL_STATIC
00044 #   define _GSASL_API __declspec(dllexport)
00045 #  elif defined _MSC_VER && ! defined GSASL_STATIC
00046 #   define _GSASL_API __declspec(dllimport)
00047 #  else
00048 #   define _GSASL_API
00049 #  endif
00050 # endif
00051
00052 # ifdef __cplusplus
00053 extern "C"
00054 {
00055 # endif
00056
00062   typedef struct Gsasl Gsasl;
00063
00069   typedef struct Gsasl_session Gsasl_session;
00070
00126   typedef enum
00127   {
00128     GSASL_OK = 0,
00129     GSASL_NEEDS_MORE = 1,
00130     GSASL_UNKNOWN_MECHANISM = 2,
00131     GSASL_MECHANISM_CALLED_TOO_MANY_TIMES = 3,
00132     GSASL_MALLOC_ERROR = 7,
00133     GSASL_BASE64_ERROR = 8,
00134     GSASL_CRYPTO_ERROR = 9,
00135     GSASL_SASLPREP_ERROR = 29,
00136     GSASL_MECHANISM_PARSE_ERROR = 30,
00137     GSASL_AUTHENTICATION_ERROR = 31,
00138     GSASL_INTEGRITY_ERROR = 33,
00139     GSASL_NO_CLIENT_CODE = 35,
00140     GSASL_NO_SERVER_CODE = 36,
00141     GSASL_NO_CALLBACK = 51,
00142     GSASL_NO_ANONYMOUS_TOKEN = 52,
00143     GSASL_NO_AUTHID = 53,
00144     GSASL_NO_AUTHZID = 54,
00145     GSASL_NO_PASSWORD = 55,
00146     GSASL_NO_PASSCODE = 56,
00147     GSASL_NO_PIN = 57,
00148     GSASL_NO_SERVICE = 58,
00149     GSASL_NO_HOSTNAME = 59,
00150     GSASL_NO_CB_TLS_UNIQUE = 65,
00151     GSASL_NO_SAML20_IDP_IDENTIFIER = 66,
00152     GSASL_NO_SAML20_REDIRECT_URL = 67,
00153     GSASL_NO_OPENID20_REDIRECT_URL = 68,
00154     GSASL_NO_CB_TLS_EXPORTER = 69,
00155     /* Mechanism specific errors. */
00156     GSASL_GSSAPI_RELEASE_BUFFER_ERROR = 37,
00157     GSASL_GSSAPI_IMPORT_NAME_ERROR = 38,
00158     GSASL_GSSAPI_INIT_SEC_CONTEXT_ERROR = 39,
00159     GSASL_GSSAPI_ACCEPT_SEC_CONTEXT_ERROR = 40,
00160     GSASL_GSSAPI_UNWRAP_ERROR = 41,
00161     GSASL_GSSAPI_WRAP_ERROR = 42,
00162     GSASL_GSSAPI_ACQUIRE_CRED_ERROR = 43,
00163     GSASL_GSSAPI_DISPLAY_NAME_ERROR = 44,
00164     GSASL_GSSAPI_UNSUPPORTED_PROTECTION_ERROR = 45,
00165     GSASL_SECURID_SERVER_NEED_ADDITIONAL_PASSCODE = 48,
00166     GSASL_SECURID_SERVER_NEED_NEW_PIN = 49,
00167     GSASL_GSSAPI_ENCAPSULATE_TOKEN_ERROR = 60,
00168     GSASL_GSSAPI_DECAPSULATE_TOKEN_ERROR = 61,
00169     GSASL_GSSAPI_INQUIRE_MECH_FOR_SASLNAME_ERROR = 62,
00170     GSASL_GSSAPI_TEST_OID_SET_MEMBER_ERROR = 63,
00171     GSASL_GSSAPI_RELEASE_OID_SET_ERROR = 64
00172       /* When adding new values, note that integers are not necessarily
00173          assigned monotonously increasingly. */
```

```
00174   } Gsasl_rc;
00175
00220   typedef enum
00221   {
00222     /* Information properties, e.g., username. */
00223     GSASL_AUTHID = 1,
00224     GSASL_AUTHZID = 2,
00225     GSASL_PASSWORD = 3,
00226     GSASL_ANONYMOUS_TOKEN = 4,
00227     GSASL_SERVICE = 5,
00228     GSASL_HOSTNAME = 6,
00229     GSASL_GSSAPI_DISPLAY_NAME = 7,
00230     GSASL_PASSCODE = 8,
00231     GSASL_SUGGESTED_PIN = 9,
00232     GSASL_PIN = 10,
00233     GSASL_REALM = 11,
00234     GSASL_DIGEST_MD5_HASHED_PASSWORD = 12,
00235     GSASL_QOPS = 13,
00236     GSASL_QOP = 14,
00237     GSASL_SCRAM_ITER = 15,
00238     GSASL_SCRAM_SALT = 16,
00239     GSASL_SCRAM_SALTED_PASSWORD = 17,
00240     GSASL_SCRAM_SERVERKEY = 23,
00241     GSASL_SCRAM_STOREDKEY = 24,
00242     GSASL_CB_TLS_UNIQUE = 18,
00243     GSASL_SAML20_IDP_IDENTIFIER = 19,
00244     GSASL_SAML20_REDIRECT_URL = 20,
00245     GSASL_OPENID20_REDIRECT_URL = 21,
00246     GSASL_OPENID20_OUTCOME_DATA = 22,
00247     GSASL_CB_TLS_EXPORTER = 25,
00248     /* Client callbacks. */
00249     GSASL_SAML20_AUTHENTICATE_IN_BROWSER = 250,
00250     GSASL_OPENID20_AUTHENTICATE_IN_BROWSER = 251,
00251     /* Server validation callback properties. */
00252     GSASL_VALIDATE_SIMPLE = 500,
00253     GSASL_VALIDATE_EXTERNAL = 501,
00254     GSASL_VALIDATE_ANONYMOUS = 502,
00255     GSASL_VALIDATE_GSSAPI = 503,
00256     GSASL_VALIDATE_SECURID = 504,
00257     GSASL_VALIDATE_SAML20 = 505,
00258     GSASL_VALIDATE_OPENID20 = 506
00259   } Gsasl_property;
00260
00285   typedef int (*Gsasl_callback_function) (Gsasl * ctx, Gsasl_session * sctx,
00286                                           Gsasl_property prop);
00287
00297   typedef enum
00298   {
00299     GSASL_MIN_MECHANISM_SIZE = 1,
00300     GSASL_MAX_MECHANISM_SIZE = 20
00301   } Gsasl_mechname_limits;
00302
00315   typedef enum
00316   {
00317     GSASL_QOP_AUTH = 1,
00318     GSASL_QOP_AUTH_INT = 2,
00319     GSASL_QOP_AUTH_CONF = 4
00320   } Gsasl_qop;
00321
00329   typedef enum
00330   {
00331     GSASL_ALLOW_UNASSIGNED = 1
00332   } Gsasl_saslprep_flags;
00333
00334   /* Library entry and exit points: version.c, init.c, done.c */
00335   extern _GSASL_API int gsasl_init (Gsasl ** ctx);
00336   extern _GSASL_API void gsasl_done (Gsasl * ctx);
00337   extern _GSASL_API const char *gsasl_check_version (const char *req_version);
00338
00339   /* Callback handling: callback.c */
00340   extern _GSASL_API void gsasl_callback_set (Gsasl * ctx,
00341                                              Gsasl_callback_function cb);
00342   extern _GSASL_API int gsasl_callback (Gsasl * ctx, Gsasl_session * sctx,
00343                                         Gsasl_property prop);
00344
00345   extern _GSASL_API void gsasl_callback_hook_set (Gsasl * ctx, void *hook);
00346   extern _GSASL_API void *gsasl_callback_hook_get (Gsasl * ctx);
00347
00348   extern _GSASL_API void gsasl_session_hook_set (Gsasl_session * sctx,
00349                                                  void *hook);
00350   extern _GSASL_API void *gsasl_session_hook_get (Gsasl_session * sctx);
00351
00352   /* Property handling: property.c */
00353   extern _GSASL_API int gsasl_property_set (Gsasl_session * sctx,
00354                                             Gsasl_property prop,
00355                                             const char *data);
00356   extern _GSASL_API int gsasl_property_set_raw (Gsasl_session * sctx,
```

```
00357                                             Gsasl_property prop,
00358                                             const char *data, size_t len);
00359   extern _GSASL_API void gsasl_property_free (Gsasl_session * sctx,
00360                                               Gsasl_property prop);
00361   extern _GSASL_API const char *gsasl_property_get (Gsasl_session * sctx,
00362                                                     Gsasl_property prop);
00363   extern _GSASL_API const char *gsasl_property_fast (Gsasl_session * sctx,
00364                                                      Gsasl_property prop);
00365
00366   /* Mechanism handling: listmech.c, supportp.c, suggest.c */
00367   extern _GSASL_API int gsasl_client_mechlist (Gsasl * ctx, char **out);
00368   extern _GSASL_API int gsasl_client_support_p (Gsasl * ctx,
00369                                                 const char *name);
00370   extern _GSASL_API const char *gsasl_client_suggest_mechanism (Gsasl * ctx,
00371                                                                 const char
00372                                                                 *mechlist);
00373
00374   extern _GSASL_API int gsasl_server_mechlist (Gsasl * ctx, char **out);
00375   extern _GSASL_API int gsasl_server_support_p (Gsasl * ctx,
00376                                                 const char *name);
00377   extern _GSASL_API int gsasl_mechanism_name_p (const char *mech);
00378
00379   /* Authentication functions: xstart.c, xstep.c, xfinish.c */
00380   extern _GSASL_API int gsasl_client_start (Gsasl * ctx, const char *mech,
00381                                             Gsasl_session ** sctx);
00382   extern _GSASL_API int gsasl_server_start (Gsasl * ctx, const char *mech,
00383                                             Gsasl_session ** sctx);
00384   extern _GSASL_API int gsasl_step (Gsasl_session * sctx,
00385                                     const char *input, size_t input_len,
00386                                     char **output, size_t *output_len);
00387   extern _GSASL_API int gsasl_step64 (Gsasl_session * sctx,
00388                                       const char *b64input, char **b64output);
00389   extern _GSASL_API void gsasl_finish (Gsasl_session * sctx);
00390
00391   /* Session functions: xcode.c, mechname.c */
00392   extern _GSASL_API int gsasl_encode (Gsasl_session * sctx,
00393                                       const char *input, size_t input_len,
00394                                       char **output, size_t *output_len);
00395   extern _GSASL_API int gsasl_decode (Gsasl_session * sctx,
00396                                       const char *input, size_t input_len,
00397                                       char **output, size_t *output_len);
00398   extern _GSASL_API const char *gsasl_mechanism_name (Gsasl_session * sctx);
00399
00400   /* Error handling: error.c */
00401   extern _GSASL_API const char *gsasl_strerror (int err);
00402   extern _GSASL_API const char *gsasl_strerror_name (int err);
00403
00404   /* Internationalized string processing: stringprep.c */
00405   extern _GSASL_API int gsasl_saslprep (const char *in,
00406                                         Gsasl_saslprep_flags flags,
00407                                         char **out, int *stringpreprc);
00408
00409   /* Crypto functions: crypto.c */
00410
00426   typedef enum
00427   {
00428     /* Hash algorithm identifiers. */
00429     GSASL_HASH_SHA1 = 2,
00430     GSASL_HASH_SHA256 = 3,
00431   } Gsasl_hash;
00432
00446   typedef enum
00447   {
00448     /* Output sizes of hashes. */
00449     GSASL_HASH_SHA1_SIZE = 20,
00450     GSASL_HASH_SHA256_SIZE = 32,
00451     GSASL_HASH_MAX_SIZE = GSASL_HASH_SHA256_SIZE
00452   } Gsasl_hash_length;
00453
00454   extern _GSASL_API int gsasl_nonce (char *data, size_t datalen);
00455   extern _GSASL_API int gsasl_random (char *data, size_t datalen);
00456
00457   extern _GSASL_API size_t gsasl_hash_length (Gsasl_hash hash);
00458
00459   extern _GSASL_API int
00460     gsasl_scram_secrets_from_salted_password (Gsasl_hash hash,
00461                                               const char *salted_password,
00462                                               char *client_key,
00463                                               char *server_key,
00464                                               char *stored_key);
00465   extern _GSASL_API int
00466     gsasl_scram_secrets_from_password (Gsasl_hash hash,
00467                                        const char *password,
00468                                        unsigned int iteration_count,
00469                                        const char *salt,
00470                                        size_t saltlen,
00471                                        char *salted_password,
```

```
00472                                     char *client_key,
00473                                     char *server_key, char *stored_key);
00474
00475   /* Utilities: md5pwd.c, base64.c, free.c */
00476   extern _GSASL_API int gsasl_simple_getpass (const char *filename,
00477                                               const char *username,
00478                                               char **key);
00479   extern _GSASL_API int gsasl_base64_to (const char *in, size_t inlen,
00480                                          char **out, size_t *outlen);
00481   extern _GSASL_API int gsasl_base64_from (const char *in, size_t inlen,
00482                                            char **out, size_t *outlen);
00483   extern _GSASL_API int gsasl_hex_to (const char *in, size_t inlen,
00484                                       char **out, size_t *outlen);
00485   extern _GSASL_API int gsasl_hex_from (const char *in, char **out,
00486                                         size_t *outlen);
00487   extern _GSASL_API void gsasl_free (void *ptr);
00488
00489   /* Get the mechanism API. */
00490 # include <gsasl-mech.h>
00491
00492 # ifdef __cplusplus
00493 }
00494 # endif
00495
00496 #endif                          /* GSASL_H */
```

## 5.193 init.c File Reference

```
#include <config.h>
#include "internal.h"
#include <gc.h>
#include "cram-md5/cram-md5.h"
#include "external/external.h"
#include "gssapi/x-gssapi.h"
#include "gs2/gs2.h"
#include "anonymous/anonymous.h"
#include "plain/plain.h"
#include "securid/securid.h"
#include "digest-md5/digest-md5.h"
#include "scram/scram.h"
#include "saml20/saml20.h"
#include "openid20/openid20.h"
#include "login/login.h"
#include "ntlm/x-ntlm.h"
```

### Functions

- int gsasl_init (Gsasl **ctx)

### 5.193.1 Function Documentation

#### 5.193.1.1 gsasl_init()

```
int gsasl_init (
            Gsasl ** ctx )
```

gsasl_init:

**Parameters**

| | |
|---|---|
| *ctx* | pointer to libgsasl handle. |

This functions initializes libgsasl. The handle pointed to by ctx is valid for use with other libgsasl functions iff this function is successful. It also register all builtin SASL mechanisms, using gsasl_register().

Return value: GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Definition at line 157 of file init.c.

## 5.194 init.c

Go to the documentation of this file.

```
00001 /* init.c --- Entry point for libgsasl.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00025 /* Get gc_init. */
00026 #include <gc.h>
00027
00028 /* Get mechanism headers. */
00029 #include "cram-md5/cram-md5.h"
00030 #include "external/external.h"
00031 #include "gssapi/x-gssapi.h"
00032 #include "gs2/gs2.h"
00033 #include "anonymous/anonymous.h"
00034 #include "plain/plain.h"
00035 #include "securid/securid.h"
00036 #include "digest-md5/digest-md5.h"
00037 #include "scram/scram.h"
00038 #include "saml20/saml20.h"
00039 #include "openid20/openid20.h"
00040
00041 #include "login/login.h"
00042 #include "ntlm/x-ntlm.h"
00043
00044 static int
00045 register_builtin_mechs (Gsasl *ctx)
00046 {
00047   int rc = GSASL_OK;
00048
00049 #ifdef USE_ANONYMOUS
00050   rc = gsasl_register (ctx, &_gsasl_anonymous_mechanism);
00051   if (rc != GSASL_OK)
00052     return rc;
00053 #endif /* USE_ANONYMOUS */
00054
00055 #ifdef USE_EXTERNAL
00056   rc = gsasl_register (ctx, &_gsasl_external_mechanism);
00057   if (rc != GSASL_OK)
00058     return rc;
00059 #endif /* USE_EXTERNAL */
00060
00061 #ifdef USE_LOGIN
```

```
00062   rc = gsasl_register (ctx, &_gsasl_login_mechanism);
00063   if (rc != GSASL_OK)
00064     return rc;
00065 #endif /* USE_LOGIN */
00066
00067 #ifdef USE_PLAIN
00068   rc = gsasl_register (ctx, &_gsasl_plain_mechanism);
00069   if (rc != GSASL_OK)
00070     return rc;
00071 #endif /* USE_PLAIN */
00072
00073 #ifdef USE_SECURID
00074   rc = gsasl_register (ctx, &_gsasl_securid_mechanism);
00075   if (rc != GSASL_OK)
00076     return rc;
00077 #endif /* USE_SECURID */
00078
00079 #ifdef USE_NTLM
00080   rc = gsasl_register (ctx, &_gsasl_ntlm_mechanism);
00081   if (rc != GSASL_OK)
00082     return rc;
00083 #endif /* USE_NTLM */
00084
00085 #ifdef USE_DIGEST_MD5
00086   rc = gsasl_register (ctx, &_gsasl_digest_md5_mechanism);
00087   if (rc != GSASL_OK)
00088     return rc;
00089 #endif /* USE_DIGEST_MD5 */
00090
00091 #ifdef USE_CRAM_MD5
00092   rc = gsasl_register (ctx, &_gsasl_cram_md5_mechanism);
00093   if (rc != GSASL_OK)
00094     return rc;
00095 #endif /* USE_CRAM_MD5 */
00096
00097 #ifdef USE_SCRAM_SHA1
00098   rc = gsasl_register (ctx, &_gsasl_scram_sha1_mechanism);
00099   if (rc != GSASL_OK)
00100     return rc;
00101
00102   rc = gsasl_register (ctx, &_gsasl_scram_sha1_plus_mechanism);
00103   if (rc != GSASL_OK)
00104     return rc;
00105 #endif /* USE_SCRAM_SHA1 */
00106
00107 #ifdef USE_SCRAM_SHA256
00108   rc = gsasl_register (ctx, &_gsasl_scram_sha256_mechanism);
00109   if (rc != GSASL_OK)
00110     return rc;
00111
00112   rc = gsasl_register (ctx, &_gsasl_scram_sha256_plus_mechanism);
00113   if (rc != GSASL_OK)
00114     return rc;
00115 #endif /* USE_SCRAM_SHA256 */
00116
00117 #ifdef USE_SAML20
00118   rc = gsasl_register (ctx, &_gsasl_saml20_mechanism);
00119   if (rc != GSASL_OK)
00120     return rc;
00121 #endif /* USE_SAML20 */
00122
00123 #ifdef USE_OPENID20
00124   rc = gsasl_register (ctx, &_gsasl_openid20_mechanism);
00125   if (rc != GSASL_OK)
00126     return rc;
00127 #endif /* USE_OPENID20 */
00128
00129 #ifdef USE_GSSAPI
00130   rc = gsasl_register (ctx, &_gsasl_gssapi_mechanism);
00131   if (rc != GSASL_OK)
00132     return rc;
00133 #endif /* USE_GSSAPI */
00134
00135 #ifdef USE_GS2
00136   rc = gsasl_register (ctx, &_gsasl_gs2_krb5_mechanism);
00137   if (rc != GSASL_OK)
00138     return rc;
00139 #endif /* USE_GSSAPI */
00140
00141   return GSASL_OK;
00142 }
00143
00156 int
00157 gsasl_init (Gsasl **ctx)
00158 {
00159   int rc;
00160
```

```
00161   if (gc_init () != GC_OK)
00162     return GSASL_CRYPTO_ERROR;
00163
00164   *ctx = (Gsasl *) calloc (1, sizeof (**ctx));
00165   if (*ctx == NULL)
00166     return GSASL_MALLOC_ERROR;
00167
00168   rc = register_builtin_mechs (*ctx);
00169   if (rc != GSASL_OK)
00170     {
00171       gsasl_done (*ctx);
00172       return rc;
00173     }
00174
00175   return GSASL_OK;
00176 }
```

## 5.195 internal.h File Reference

```
#include "gsasl.h"
#include <stdlib.h>
#include <string.h>
```

### Data Structures

- struct Gsasl
- struct Gsasl_session

## 5.196 internal.h

Go to the documentation of this file.

```
00001 /* internal.h --- Internal header with hidden library handle structures.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef INTERNAL_H
00023 # define INTERNAL_H
00024
00025 /* Get specifications. */
00026 # include "gsasl.h"
00027
00028 /* Get malloc, free, ... */
00029 # include <stdlib.h>
00030
00031 /* Get strlen, strcpy, ... */
00032 # include <string.h>
00033
00034 /* Main library handle. */
00035 struct Gsasl
00036 {
00037   size_t n_client_mechs;
```

```
00038   Gsasl_mechanism *client_mechs;
00039   size_t n_server_mechs;
00040   Gsasl_mechanism *server_mechs;
00041   /* Callback. */
00042   Gsasl_callback_function cb;
00043   void *application_hook;
00044 };
00045
00046 /* Per-session library handle. */
00047 struct Gsasl_session
00048 {
00049   Gsasl *ctx;
00050   int clientp;
00051   Gsasl_mechanism *mech;
00052   void *mech_data;
00053   void *application_hook;
00054
00055   /* Properties. */
00056   char *anonymous_token;
00057   char *authid;
00058   char *authzid;
00059   char *password;
00060   char *passcode;
00061   char *pin;
00062   char *suggestedpin;
00063   char *service;
00064   char *hostname;
00065   char *gssapi_display_name;
00066   char *realm;
00067   char *digest_md5_hashed_password;
00068   char *qops;
00069   char *qop;
00070   char *scram_iter;
00071   char *scram_salt;
00072   char *scram_salted_password;
00073   char *scram_serverkey;
00074   char *scram_storedkey;
00075   char *cb_tls_unique;
00076   char *cb_tls_exporter;
00077   char *saml20_idp_identifier;
00078   char *saml20_redirect_url;
00079   char *openid20_redirect_url;
00080   char *openid20_outcome_data;
00081   /* If you add anything here, remember to change change
00082      gsasl_finish() in xfinish.c and map() in property.c.  */
00083 };
00084
00085 #endif /* INTERNAL_H */
```

## 5.197 listmech.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- int gsasl_client_mechlist (Gsasl *ctx, char **out)
- int gsasl_server_mechlist (Gsasl *ctx, char **out)

### 5.197.1 Function Documentation

#### 5.197.1.1 gsasl_client_mechlist()

```
int gsasl_client_mechlist (
            Gsasl * ctx,
            char ** out )
```

gsasl_client_mechlist:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *out* | newly allocated output character array. |

Return a newly allocated string containing SASL names, separated by space, of mechanisms supported by the libgsasl client. @out is allocated by this function, and it is the responsibility of caller to deallocate it.

Return value: Returns GSASL_OK if successful, or error code.

Definition at line 74 of file listmech.c.

#### 5.197.1.2 gsasl_server_mechlist()

```
int gsasl_server_mechlist (
             Gsasl * ctx,
             char ** out )
```

gsasl_server_mechlist:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *out* | newly allocated output character array. |

Return a newly allocated string containing SASL names, separated by space, of mechanisms supported by the libgsasl server. @out is allocated by this function, and it is the responsibility of caller to deallocate it.

Return value: Returns GSASL_OK if successful, or error code.

Definition at line 93 of file listmech.c.

## 5.198 listmech.c

Go to the documentation of this file.

```
00001 /* listmech.c --- List active client and server mechanisms.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
```

```
00024
00025 static int
00026 _gsasl_listmech (Gsasl *ctx,
00027                  Gsasl_mechanism *mechs,
00028                  size_t n_mechs, char **out, int clientp)
00029 {
00030   Gsasl_session *sctx;
00031   char *list;
00032   size_t i;
00033   int rc;
00034
00035   list = calloc (n_mechs + 1, GSASL_MAX_MECHANISM_SIZE + 1);
00036   if (!list)
00037     return GSASL_MALLOC_ERROR;
00038
00039   for (i = 0; i < n_mechs; i++)
00040     {
00041       if (clientp)
00042         rc = gsasl_client_start (ctx, mechs[i].name, &sctx);
00043       else
00044         rc = gsasl_server_start (ctx, mechs[i].name, &sctx);
00045
00046       if (rc == GSASL_OK)
00047         {
00048           gsasl_finish (sctx);
00049
00050           strcat (list, mechs[i].name);
00051           if (i < n_mechs - 1)
00052             strcat (list, " ");
00053         }
00054     }
00055
00056   *out = list;
00057
00058   return GSASL_OK;
00059 }
00060
00073 int
00074 gsasl_client_mechlist (Gsasl *ctx, char **out)
00075 {
00076   return _gsasl_listmech (ctx, ctx->client_mechs, ctx->n_client_mechs,
00077                           out, 1);
00078 }
00079
00092 int
00093 gsasl_server_mechlist (Gsasl *ctx, char **out)
00094 {
00095   return _gsasl_listmech (ctx, ctx->server_mechs, ctx->n_server_mechs,
00096                           out, 0);
00097 }
```

## 5.199 md5pwd.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- int gsasl_simple_getpass (const char *filename, const char *username, char **key)

### 5.199.1 Function Documentation

#### 5.199.1.1 gsasl_simple_getpass()

```
int gsasl_simple_getpass (
            const char * filename,
            const char * username,
            char ** key )
```

gsasl_simple_getpass:

**Parameters**

| | |
|---|---|
| *filename* | filename of file containing passwords. |
| *username* | username string. |
| *key* | newly allocated output character array. |

Retrieve password for user from specified file. The buffer @key contain the password if this function is successful. The caller is responsible for deallocating it.

The file should be on the UoW "MD5 Based Authentication" format, which means it is in text format with comments denoted by # first on the line, with user entries looking as "usernameTABpassword". This function removes CR and LF at the end of lines before processing. TAB, CR, and LF denote ASCII values 9, 13, and 10, respectively.

Return value: Return GSASL_OK if output buffer contains the password, GSASL_AUTHENTICATION_ERROR if the user could not be found, or other error code.

Definition at line 47 of file md5pwd.c.

## 5.200 md5pwd.c

Go to the documentation of this file.

```
00001 /* md5pwd.c --- Find passwords in UoW imapd MD5 type password files.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00046 int
00047 gsasl_simple_getpass (const char *filename, const char *username, char **key)
00048 {
00049   size_t userlen = strlen (username);
00050   char *line = NULL;
00051   size_t n = 0;
00052   FILE *fh;
00053
00054   fh = fopen (filename, "r");
00055   if (fh)
00056     {
00057       while (!feof (fh))
00058         {
00059           if (getline (&line, &n, fh) < 0)
00060             break;
00061
00062           if (line[0] == '#')
00063             continue;
00064
00065           if (line[strlen (line) - 1] == '\r')
00066             line[strlen (line) - 1] = '\0';
00067           if (line[strlen (line) - 1] == '\n')
00068             line[strlen (line) - 1] = '\0';
00069
00070           if (strncmp (line, username, userlen) == 0 && line[userlen] == '\t')
00071             {
```

```
00072               *key = malloc (strlen (line) - userlen);
00073               if (!*key)
00074                 {
00075                   free (line);
00076                   return GSASL_MALLOC_ERROR;
00077                 }
00078
00079               strcpy (*key, line + userlen + 1);
00080
00081               free (line);
00082
00083               fclose (fh);
00084
00085               return GSASL_OK;
00086             }
00087         }
00088
00089       fclose (fh);
00090     }
00091
00092   free (line);
00093
00094   return GSASL_AUTHENTICATION_ERROR;
00095 }
```

## 5.201 mechname.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- const char * gsasl_mechanism_name (Gsasl_session *sctx)

### 5.201.1 Function Documentation

#### 5.201.1.1 gsasl_mechanism_name()

```
const char * gsasl_mechanism_name (
             Gsasl_session * sctx )
```

gsasl_mechanism_name:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |

This function returns the name of the SASL mechanism used in the session. The pointer must not be deallocated by the caller.

Return value: Returns a zero terminated character array with the name of the SASL mechanism, or NULL if not known.

Since: 0.2.28

Definition at line 38 of file mechname.c.

## 5.202 mechname.c

Go to the documentation of this file.

```
00001 /* mechname.c --- Get name of SASL mechanism used in a session.
00002  * Copyright (C) 2008-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00037 const char *
00038 gsasl_mechanism_name (Gsasl_session *sctx)
00039 {
00040   if (!sctx || !sctx->mech)
00041     return NULL;
00042   return sctx->mech->name;
00043 }
```

## 5.203 mechtools.c File Reference

```
#include <config.h>
#include "mechtools.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <gsasl.h>
#include <gc.h>
```

### Functions

- int _gsasl_parse_gs2_header (const char *data, size_t len, char **authzid, size_t *headerlen)
- int _gsasl_gs2_generate_header (bool nonstd, char cbflag, const char *cbname, const char *authzid, size_t extralen, const char *extra, char **gs2h, size_t *gs2hlen)
- void _gsasl_hex_encode (const char *in, size_t inlen, char *out)
- void _gsasl_hex_decode (const char *hexstr, char *bin)
- bool _gsasl_hex_p (const char *hexstr)
- int _gsasl_hash (Gsasl_hash hash, const char *in, size_t inlen, char *outhash)
- int _gsasl_hmac (Gsasl_hash hash, const char *key, size_t keylen, const char *in, size_t inlen, char *outhash)
- int _gsasl_pbkdf2 (Gsasl_hash hash, const char *password, size_t passwordlen, const char *salt, size_t saltlen, unsigned int c, char *dk, size_t dklen)

## 5.203.1 Function Documentation

### 5.203.1.1 _gsasl_gs2_generate_header()

```
int _gsasl_gs2_generate_header (
             bool nonstd,
             char cbflag,
             const char * cbname,
             const char * authzid,
             size_t extralen,
             const char * extra,
             char ** gs2h,
             size_t * gs2hlen )
```

Definition at line 165 of file mechtools.c.

### 5.203.1.2 _gsasl_hash()

```
int _gsasl_hash (
             Gsasl_hash hash,
             const char * in,
             size_t inlen,
             char * outhash )
```

Definition at line 295 of file mechtools.c.

### 5.203.1.3 _gsasl_hex_decode()

```
void _gsasl_hex_decode (
             const char * hexstr,
             char * bin )
```

Definition at line 255 of file mechtools.c.

### 5.203.1.4 _gsasl_hex_encode()

```
void _gsasl_hex_encode (
             const char * in,
             size_t inlen,
             char * out )
```

Definition at line 220 of file mechtools.c.

### 5.203.1.5 _gsasl_hex_p()

```
bool _gsasl_hex_p (
             const char * hexstr )
```

Definition at line 267 of file mechtools.c.

#### 5.203.1.6 _gsasl_hmac()

```
int _gsasl_hmac (
              Gsasl_hash hash,
              const char * key,
              size_t keylen,
              const char * in,
              size_t inlen,
              char * outhash )
```

Definition at line 328 of file mechtools.c.

#### 5.203.1.7 _gsasl_parse_gs2_header()

```
int _gsasl_parse_gs2_header (
              const char * data,
              size_t len,
              char ** authzid,
              size_t * headerlen )
```

Definition at line 96 of file mechtools.c.

#### 5.203.1.8 _gsasl_pbkdf2()

```
int _gsasl_pbkdf2 (
              Gsasl_hash hash,
              const char * password,
              size_t passwordlen,
              const char * salt,
              size_t saltlen,
              unsigned int c,
              char * dk,
              size_t dklen )
```

Definition at line 367 of file mechtools.c.

## 5.204 mechtools.c

Go to the documentation of this file.

```
00001 /* mechtools.c --- Helper functions available for use by any mechanism.
00002  * Copyright (C) 2010-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
```

```
00020  */
00021
00022 #include <config.h>
00023
00024 /* Get specification. */
00025 #include "mechtools.h"
00026
00027 /* Get strcmp. */
00028 #include <string.h>
00029
00030 /* Get malloc, free. */
00031 #include <stdlib.h>
00032
00033 /* Get asprintf. */
00034 #include <stdio.h>
00035
00036 /* Get error codes. */
00037 #include <gsasl.h>
00038
00039 /* Gnulib gc.h */
00040 #include <gc.h>
00041
00042 /* Create in AUTHZID a newly allocated copy of STR where =2C is
00043    replaced with , and =3D is replaced with =.  Return GSASL_OK on
00044    success, GSASL_MALLOC_ERROR on memory errors, GSASL_PARSE_ERRORS if
00045    string contains any unencoded ',' or incorrectly encoded
00046    sequence.  */
00047 static int
00048 unescape_authzid (const char *str, size_t len, char **authzid)
00049 {
00050   char *p;
00051
00052   if (memchr (str, ',', len) != NULL)
00053     return GSASL_MECHANISM_PARSE_ERROR;
00054
00055   p = *authzid = malloc (len + 1);
00056   if (!p)
00057     return GSASL_MALLOC_ERROR;
00058
00059   while (len > 0 && *str)
00060     {
00061       if (len >= 3 && str[0] == '=' && str[1] == '2' && str[2] == 'C')
00062         {
00063           *p++ = ',';
00064           str += 3;
00065           len -= 3;
00066         }
00067       else if (len >= 3 && str[0] == '=' && str[1] == '3' && str[2] == 'D')
00068         {
00069           *p++ = '=';
00070           str += 3;
00071           len -= 3;
00072         }
00073       else if (str[0] == '=')
00074         {
00075           free (*authzid);
00076           *authzid = NULL;
00077           return GSASL_MECHANISM_PARSE_ERROR;
00078         }
00079       else
00080         {
00081           *p++ = *str;
00082           str++;
00083           len--;
00084         }
00085     }
00086   *p = '\0';
00087
00088   return GSASL_OK;
00089 }
00090
00091 /* Parse the GS2 header containing flags and authorization identity.
00092    Put authorization identity (or NULL) in AUTHZID and length of
00093    header in HEADERLEN.  Return GSASL_OK on success or an error
00094    code.*/
00095 int
00096 _gsasl_parse_gs2_header (const char *data, size_t len,
00097                          char **authzid, size_t *headerlen)
00098 {
00099   const char *authzid_endptr;
00100
00101   if (len < 3)
00102     return GSASL_MECHANISM_PARSE_ERROR;
00103
00104   if (strncmp (data, "n,", 3) == 0)
00105     {
00106       *headerlen = 3;
```

```
00107       *authzid = NULL;
00108     }
00109   else if (strncmp (data, "n,a=", 4) == 0 &&
00110            (authzid_endptr = memchr (data + 4, ',', len - 4)))
00111     {
00112       int res;
00113
00114       res = unescape_authzid (data + 4, authzid_endptr - (data + 4), authzid);
00115       if (res != GSASL_OK)
00116         return res;
00117
00118       *headerlen = authzid_endptr - data + 1;
00119     }
00120   else
00121     return GSASL_MECHANISM_PARSE_ERROR;
00122
00123   return GSASL_OK;
00124 }
00125
00126 /* Return newly allocated copy of STR with all occurrences of ','
00127    replaced with =2C and '=' with '=3D', or return NULL on memory
00128    allocation errors.  */
00129 static char *
00130 escape_authzid (const char *str)
00131 {
00132   char *out = malloc (strlen (str) * 3 + 1);
00133   char *p = out;
00134
00135   if (!out)
00136     return NULL;
00137
00138   while (*str)
00139     {
00140       if (*str == ',')
00141         {
00142           memcpy (p, "=2C", 3);
00143           p += 3;
00144         }
00145       else if (*str == '=')
00146         {
00147           memcpy (p, "=3D", 3);
00148           p += 3;
00149         }
00150       else
00151         {
00152           *p = *str;
00153           p++;
00154         }
00155       str++;
00156     }
00157   *p = '\0';
00158
00159   return out;
00160 }
00161
00162 /* Generate a newly allocated GS2 header, escaping authzid
00163    appropriately, and appending EXTRA. */
00164 int
00165 _gsasl_gs2_generate_header (bool nonstd, char cbflag,
00166                             const char *cbname, const char *authzid,
00167                             size_t extralen, const char *extra,
00168                             char **gs2h, size_t *gs2hlen)
00169 {
00170   int elen = extralen;
00171   char *gs2cbflag;
00172   int len;
00173
00174   if (cbflag == 'p')
00175     len = asprintf (&gs2cbflag, "p=%s", cbname);
00176   else if (cbflag == 'n')
00177     len = asprintf (&gs2cbflag, "n");
00178   else if (cbflag == 'y')
00179     len = asprintf (&gs2cbflag, "y");
00180   else
00181     /* internal caller error */
00182     return GSASL_MECHANISM_PARSE_ERROR;
00183
00184   if (len <= 0 || gs2cbflag == NULL)
00185     return GSASL_MALLOC_ERROR;
00186
00187   if (authzid)
00188     {
00189       char *escaped_authzid = escape_authzid (authzid);
00190
00191       if (!escaped_authzid)
00192         {
00193           free (gs2cbflag);
```

```
00194           return GSASL_MALLOC_ERROR;
00195         }
00196
00197       len = asprintf (gs2h, "%s%s,a=%s,%.*s", nonstd ? "F," : "",
00198                       gs2cbflag, escaped_authzid, elen, extra);
00199
00200       free (escaped_authzid);
00201     }
00202   else
00203     len = asprintf (gs2h, "%s%s,,%.*s", nonstd ? "F," : "", gs2cbflag,
00204                     elen, extra);
00205
00206   free (gs2cbflag);
00207
00208   if (len <= 0 || *gs2h == NULL)
00209     return GSASL_MALLOC_ERROR;
00210
00211   *gs2hlen = len;
00212
00213   return GSASL_OK;
00214 }
00215
00216 /* Hex encode binary octet array IN of INLEN length, putting the hex
00217    encoded string in OUT which must have room for the data and
00218    terminating zero, i.e., 2*INLEN+1. */
00219 void
00220 _gsasl_hex_encode (const char *in, size_t inlen, char *out)
00221 {
00222   static const char trans[] = "0123456789abcdef";
00223
00224   while (inlen--)
00225     {
00226       unsigned char c = *in++;
00227       *out++ = trans[(c >> 4) & 0xf];
00228       *out++ = trans[c & 0xf];
00229     }
00230
00231   *out = '\0';
00232 }
00233
00234 static char
00235 hexdigit_to_char (char hexdigit)
00236 {
00237   if (hexdigit >= '0' && hexdigit <= '9')
00238     return hexdigit - '0';
00239   if (hexdigit >= 'a' && hexdigit <= 'f')
00240     return hexdigit - 'a' + 10;
00241   return 0;
00242 }
00243
00244 static char
00245 hex_to_char (char u, char l)
00246 {
00247   return (char) (((unsigned char) hexdigit_to_char (u)) * 16
00248                  + hexdigit_to_char (l));
00249 }
00250
00251 /* Hex decode string HEXSTR containing only hex "0-9A-F" characters
00252    into binary buffer BIN which must have room for data, i.e., strlen
00253    (hexstr)/2. */
00254 void
00255 _gsasl_hex_decode (const char *hexstr, char *bin)
00256 {
00257   while (*hexstr)
00258     {
00259       *bin = hex_to_char (hexstr[0], hexstr[1]);
00260       hexstr += 2;
00261       bin++;
00262     }
00263 }
00264
00265 /* Return whether string contains hex "0-9a-f" symbols only. */
00266 bool
00267 _gsasl_hex_p (const char *hexstr)
00268 {
00269   static const char hexalpha[] = "0123456789abcdef";
00270
00271   for (; *hexstr; hexstr++)
00272     if (strchr (hexalpha, *hexstr) == NULL)
00273       return false;
00274
00275   return true;
00276 }
00277
00278 /*
00279  * _gsasl_hash:
00280  * @hash: a %Gsasl_hash hash algorithm identifier, e.g. #GSASL_HASH_SHA256.
```

```
00281  * @in: input character array of data to hash.
00282  * @inlen: length of input character array of data to hash.
00283  * @outhash: buffer to hold hash of data.
00284  *
00285  * Compute hash of data using the @hash algorithm.  The @outhash
00286  * buffer must have room to hold the size of @hash's output; a safe
00287  * value that have room for all possible outputs is
00288  * %GSASL_HASH_MAX_SIZE.
00289  *
00290  * Return value: Returns %GSASL_OK iff successful.
00291  *
00292  * Since: 1.10
00293  **/
00294 int
00295 _gsasl_hash (Gsasl_hash hash, const char *in, size_t inlen, char *outhash)
00296 {
00297   int rc;
00298
00299   if (hash == GSASL_HASH_SHA1)
00300     rc = gc_sha1 (in, inlen, outhash);
00301   else if (hash == GSASL_HASH_SHA256)
00302     rc = gc_sha256 (in, inlen, outhash);
00303   else
00304     rc = GSASL_CRYPTO_ERROR;
00305
00306   return rc;
00307 }
00308
00309 /*
00310  * _gsasl_hmac:
00311  * @hash: a %Gsasl_hash hash algorithm identifier, e.g. #GSASL_HASH_SHA256.
00312  * @key: input character array with key to use.
00313  * @keylen: length of input character array with key to use.
00314  * @in: input character array of data to hash.
00315  * @inlen: length of input character array of data to hash.
00316  * @outhash: buffer to hold keyed hash of data.
00317  *
00318  * Compute keyed checksum of data using HMAC for the @hash algorithm.
00319  * The @outhash buffer must have room to hold the size of @hash's
00320  * output; a safe value that have room for all possible outputs is
00321  * %GSASL_HASH_MAX_SIZE.
00322  *
00323  * Return value: Returns %GSASL_OK iff successful.
00324  *
00325  * Since: 1.10
00326  **/
00327 int
00328 _gsasl_hmac (Gsasl_hash hash,
00329              const char *key, size_t keylen,
00330              const char *in, size_t inlen, char *outhash)
00331 {
00332   int rc;
00333
00334   if (hash == GSASL_HASH_SHA1)
00335     rc = gc_hmac_sha1 (key, keylen, in, inlen, outhash);
00336   else if (hash == GSASL_HASH_SHA256)
00337     rc = gc_hmac_sha256 (key, keylen, in, inlen, outhash);
00338   else
00339     rc = GSASL_CRYPTO_ERROR;
00340
00341   return rc;
00342 }
00343
00344 /*
00345  * gsasl_pbkdf2:
00346  * @hash: a %Gsasl_hash hash algorithm identifier.
00347  * @password: input character array with password to use.
00348  * @passwordlen: length of @password.
00349  * @salt: input character array with salt, typically a short string.
00350  * @saltlen: length of @salt.
00351  * @c: iteration count, typically larger than 4096.
00352  * @dk: output buffer, must be able to hold @dklen.
00353  * @dklen: length of output buffer, or 0 to indicate @hash output size.
00354  *
00355  * Hash and salt password according to PBKDF2 algorithm with the @hash
00356  * function used in HMAC.  This function can be used to prepare SCRAM
00357  * SaltedPassword values for the %GSASL_SCRAM_SALTED_PASSWORD
00358  * property.  Note that password should normally be prepared using
00359  * gsasl_saslprep(GSASL_ALLOW_UNASSIGNED) before calling this
00360  * function.
00361  *
00362  * Return value: Returns %GSASL_OK if successful, or error code.
00363  *
00364  * Since: 1.10
00365  **/
00366 int
00367 _gsasl_pbkdf2 (Gsasl_hash hash,
```

```
00368                 const char *password, size_t passwordlen,
00369                 const char *salt, size_t saltlen,
00370                 unsigned int c, char *dk, size_t dklen)
00371 {
00372   int rc;
00373   Gc_hash gch;
00374
00375   switch (hash)
00376     {
00377     case GSASL_HASH_SHA1:
00378       if (dklen == 0)
00379         dklen = GSASL_HASH_SHA1_SIZE;
00380       gch = GC_SHA1;
00381       break;
00382
00383     case GSASL_HASH_SHA256:
00384       if (dklen == 0)
00385         dklen = GSASL_HASH_SHA256_SIZE;
00386       gch = GC_SHA256;
00387       break;
00388
00389     default:
00390       return GSASL_CRYPTO_ERROR;
00391     }
00392
00393   rc = gc_pbkdf2_hmac (gch, password, passwordlen,
00394                        salt, saltlen, c, dk, dklen);
00395   if (rc != GC_OK)
00396     return GSASL_CRYPTO_ERROR;
00397
00398   return GSASL_OK;
00399 }
```

## 5.205 mechtools.h File Reference

```
#include <stddef.h>
#include <stdbool.h>
#include <gsasl.h>
```

### Functions

- int _gsasl_parse_gs2_header (const char *data, size_t len, char **authzid, size_t *headerlen)
- int _gsasl_gs2_generate_header (bool nonstd, char cbflag, const char *cbname, const char *authzid, size_t extralen, const char *extra, char **gs2h, size_t *gs2hlen)
- void _gsasl_hex_encode (const char *in, size_t inlen, char *out)
- void _gsasl_hex_decode (const char *hexstr, char *bin)
- bool _gsasl_hex_p (const char *hexstr)
- int _gsasl_hash (Gsasl_hash hash, const char *in, size_t inlen, char *out)
- int _gsasl_hmac (Gsasl_hash hash, const char *key, size_t keylen, const char *in, size_t inlen, char *outhash)
- int _gsasl_pbkdf2 (Gsasl_hash hash, const char *password, size_t passwordlen, const char *salt, size_t saltlen, unsigned int c, char *dk, size_t dklen)

### 5.205.1 Function Documentation

#### 5.205.1.1 _gsasl_gs2_generate_header()

```
int _gsasl_gs2_generate_header (
            bool nonstd,
            char cbflag,
            const char * cbname,
```

```
              const char * authzid,
              size_t extralen,
              const char * extra,
              char ** gs2h,
              size_t * gs2hlen )  [extern]
```

Definition at line 165 of file mechtools.c.

#### 5.205.1.2 _gsasl_hash()

```
int _gsasl_hash (
              Gsasl_hash hash,
              const char * in,
              size_t inlen,
              char * out )  [extern]
```

Definition at line 295 of file mechtools.c.

#### 5.205.1.3 _gsasl_hex_decode()

```
void _gsasl_hex_decode (
              const char * hexstr,
              char * bin )  [extern]
```

Definition at line 255 of file mechtools.c.

#### 5.205.1.4 _gsasl_hex_encode()

```
void _gsasl_hex_encode (
              const char * in,
              size_t inlen,
              char * out )  [extern]
```

Definition at line 220 of file mechtools.c.

#### 5.205.1.5 _gsasl_hex_p()

```
bool _gsasl_hex_p (
              const char * hexstr )  [extern]
```

Definition at line 267 of file mechtools.c.

#### 5.205.1.6 _gsasl_hmac()

```
int _gsasl_hmac (
              Gsasl_hash hash,
              const char * key,
              size_t keylen,
              const char * in,
              size_t inlen,
              char * outhash )  [extern]
```

Definition at line 328 of file mechtools.c.

#### 5.205.1.7 _gsasl_parse_gs2_header()

```
int _gsasl_parse_gs2_header (
             const char * data,
             size_t len,
             char ** authzid,
             size_t * headerlen )  [extern]
```

Definition at line 96 of file mechtools.c.

#### 5.205.1.8 _gsasl_pbkdf2()

```
int _gsasl_pbkdf2 (
             Gsasl_hash hash,
             const char * password,
             size_t passwordlen,
             const char * salt,
             size_t saltlen,
             unsigned int c,
             char * dk,
             size_t dklen )  [extern]
```

Definition at line 367 of file mechtools.c.

## 5.206 mechtools.h

Go to the documentation of this file.

```
00001 /* mechtools.h --- Helper functions available for use by any mechanism.
00002  * Copyright (C) 2010-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #ifndef MECHTOOLS_H
00023 # define MECHTOOLS_H
00024
00025 /* Get size_t. */
00026 # include <stddef.h>
00027
00028 /* Get bool. */
00029 # include <stdbool.h>
00030
00031 # include <gsasl.h>
00032
00033 extern int _gsasl_parse_gs2_header (const char *data, size_t len,
00034                                     char **authzid, size_t *headerlen);
00035
00036 extern int _gsasl_gs2_generate_header (bool nonstd, char cbflag,
00037                                        const char *cbname,
00038                                        const char *authzid, size_t extralen,
00039                                        const char *extra, char **gs2h,
00040                                        size_t *gs2hlen);
```

```
00041
00042 extern void _gsasl_hex_encode (const char *in, size_t inlen, char *out);
00043 extern void _gsasl_hex_decode (const char *hexstr, char *bin);
00044 extern bool _gsasl_hex_p (const char *hexstr);
00045
00046 extern int _gsasl_hash (Gsasl_hash hash,
00047                         const char *in, size_t inlen, char *out);
00048 extern int _gsasl_hmac (Gsasl_hash hash,
00049                         const char *key, size_t keylen,
00050                         const char *in, size_t inlen, char *outhash);
00051
00052 extern int _gsasl_pbkdf2 (Gsasl_hash hash,
00053                           const char *password,
00054                           size_t passwordlen,
00055                           const char *salt,
00056                           size_t saltlen,
00057                           unsigned int c, char *dk, size_t dklen);
00058
00059 #endif
```

## 5.207 property.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- void gsasl_property_free (Gsasl_session *sctx, Gsasl_property prop)
- int gsasl_property_set (Gsasl_session *sctx, Gsasl_property prop, const char *data)
- int gsasl_property_set_raw (Gsasl_session *sctx, Gsasl_property prop, const char *data, size_t len)
- const char * gsasl_property_fast (Gsasl_session *sctx, Gsasl_property prop)
- const char * gsasl_property_get (Gsasl_session *sctx, Gsasl_property prop)

### 5.207.1 Function Documentation

#### 5.207.1.1 gsasl_property_fast()

```
const char * gsasl_property_fast (
            Gsasl_session * sctx,
            Gsasl_property prop )
```

gsasl_property_fast:

**Parameters**

| | |
|---|---|
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type, indicating the type of data in @data. |

Retrieve the data stored in the session handle for given property @prop.

The pointer is to live data, and must not be deallocated or modified in any way.

This function will not invoke the application callback.

Return value: Return property value, if known, or NULL if no value known.

Since: 0.2.0

Definition at line 261 of file property.c.

#### 5.207.1.2 gsasl_property_free()

```
void gsasl_property_free (
             Gsasl_session * sctx,
             Gsasl_property prop )
```

gsasl_property_free:

**Parameters**

| | |
|---|---|
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type to clear |

Deallocate associated data with property @prop in session handle. After this call, gsasl_property_fast(@sctx, @prop) will always return NULL.

Since: 2.0.0

Definition at line 158 of file property.c.

#### 5.207.1.3 gsasl_property_get()

```
const char * gsasl_property_get (
             Gsasl_session * sctx,
             Gsasl_property prop )
```

gsasl_property_get:

**Parameters**

| | |
|---|---|
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type, indicating the type of data in @data. |

Retrieve the data stored in the session handle for given property @prop, possibly invoking the application callback to get the value.

The pointer is to live data, and must not be deallocated or modified in any way.

This function will invoke the application callback, using gsasl_callback(), when a property value is not known.

Return value: Return data for property, or NULL if no value known.

Since: 0.2.0

Definition at line 291 of file property.c.

#### 5.207.1.4 gsasl_property_set()

```
int gsasl_property_set (
             Gsasl_session * sctx,
             Gsasl_property prop,
             const char * data )
```

gsasl_property_set:

**Parameters**

| | |
|---|---|
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type, indicating the type of data in @data. |
| *data* | zero terminated character string to store. |

Make a copy of @data and store it in the session handle for the indicated property @prop.

You can immediately deallocate @data after calling this function, without affecting the data stored in the session handle.

Return value: GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Since: 0.2.0

Definition at line 188 of file property.c.

#### 5.207.1.5 gsasl_property_set_raw()

```
int gsasl_property_set_raw (
             Gsasl_session * sctx,
             Gsasl_property prop,
             const char * data,
             size_t len )
```

gsasl_property_set_raw:

**Parameters**

| | |
|---|---|
| *sctx* | session handle. |
| *prop* | enumerated value of Gsasl_property type, indicating the type of data in @data. |
| *data* | character string to store. |
| *len* | length of character string to store. |

Make a copy of @len sized @data and store a zero terminated version of it in the session handle for the indicated property @prop.

You can immediately deallocate @data after calling this function, without affecting the data stored in the session handle.

Except for the length indicator, this function is identical to gsasl_property_set.

Return value: GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Since: 0.2.0

Definition at line 217 of file property.c.

# 5.208 property.c

Go to the documentation of this file.

```
00001 /* property.c --- Callback property handling.
00002  * Copyright (C) 2004-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00025 static char **
00026 map (Gsasl_session *sctx, Gsasl_property prop)
00027 {
00028   char **p = NULL;
00029
00030   if (!sctx)
00031     return NULL;
00032
00033   switch (prop)
00034     {
00035     case GSASL_ANONYMOUS_TOKEN:
00036       p = &sctx->anonymous_token;
00037       break;
00038
00039     case GSASL_SERVICE:
00040       p = &sctx->service;
00041       break;
00042
00043     case GSASL_HOSTNAME:
00044       p = &sctx->hostname;
00045       break;
00046
00047     case GSASL_AUTHID:
00048       p = &sctx->authid;
00049       break;
00050
00051     case GSASL_AUTHZID:
00052       p = &sctx->authzid;
00053       break;
00054
00055     case GSASL_PASSWORD:
00056       p = &sctx->password;
00057       break;
00058
00059     case GSASL_PASSCODE:
00060       p = &sctx->passcode;
00061       break;
00062
00063     case GSASL_PIN:
00064       p = &sctx->pin;
00065       break;
00066
00067     case GSASL_SUGGESTED_PIN:
00068       p = &sctx->suggestedpin;
00069       break;
00070
00071     case GSASL_GSSAPI_DISPLAY_NAME:
00072       p = &sctx->gssapi_display_name;
00073       break;
00074
00075     case GSASL_REALM:
00076       p = &sctx->realm;
00077       break;
00078
00079     case GSASL_DIGEST_MD5_HASHED_PASSWORD:
00080       p = &sctx->digest_md5_hashed_password;
00081       break;
00082
```

```
00083     case GSASL_QOPS:
00084       p = &sctx->qops;
00085       break;
00086
00087     case GSASL_QOP:
00088       p = &sctx->qop;
00089       break;
00090
00091     case GSASL_SCRAM_ITER:
00092       p = &sctx->scram_iter;
00093       break;
00094
00095     case GSASL_SCRAM_SALT:
00096       p = &sctx->scram_salt;
00097       break;
00098
00099     case GSASL_SCRAM_SALTED_PASSWORD:
00100       p = &sctx->scram_salted_password;
00101       break;
00102
00103     case GSASL_SCRAM_SERVERKEY:
00104       p = &sctx->scram_serverkey;
00105       break;
00106
00107     case GSASL_SCRAM_STOREDKEY:
00108       p = &sctx->scram_storedkey;
00109       break;
00110
00111     case GSASL_CB_TLS_UNIQUE:
00112       p = &sctx->cb_tls_unique;
00113       break;
00114
00115     case GSASL_CB_TLS_EXPORTER:
00116       p = &sctx->cb_tls_exporter;
00117       break;
00118
00119     case GSASL_SAML20_IDP_IDENTIFIER:
00120       p = &sctx->saml20_idp_identifier;
00121       break;
00122
00123     case GSASL_SAML20_REDIRECT_URL:
00124       p = &sctx->saml20_redirect_url;
00125       break;
00126
00127     case GSASL_OPENID20_REDIRECT_URL:
00128       p = &sctx->openid20_redirect_url;
00129       break;
00130
00131     case GSASL_OPENID20_OUTCOME_DATA:
00132       p = &sctx->openid20_outcome_data;
00133       break;
00134
00135       /* If you add anything here, remember to change change
00136          gsasl_finish() in xfinish.c and Gsasl_session in
00137          internal.h.  */
00138
00139     default:
00140       break;
00141     }
00142
00143   return p;
00144 }
00145
00157 void
00158 gsasl_property_free (Gsasl_session *sctx, Gsasl_property prop)
00159 {
00160   char **p = map (sctx, prop);
00161
00162   if (p)
00163     {
00164       free (*p);
00165       *p = NULL;
00166     }
00167 }
00168
00187 int
00188 gsasl_property_set (Gsasl_session *sctx, Gsasl_property prop,
00189                     const char *data)
00190 {
00191   return gsasl_property_set_raw (sctx, prop, data, data ? strlen (data) : 0);
00192 }
00193
00216 int
00217 gsasl_property_set_raw (Gsasl_session *sctx, Gsasl_property prop,
00218                         const char *data, size_t len)
00219 {
00220   char **p = map (sctx, prop);
```

```
00221
00222   if (p)
00223     {
00224       free (*p);
00225       if (data)
00226         {
00227           *p = malloc (len + 1);
00228           if (!*p)
00229             return GSASL_MALLOC_ERROR;
00230
00231           memcpy (*p, data, len);
00232           (*p)[len] = '\0';
00233         }
00234       else
00235         *p = NULL;
00236     }
00237
00238   return GSASL_OK;
00239 }
00240
00260 const char *
00261 gsasl_property_fast (Gsasl_session *sctx, Gsasl_property prop)
00262 {
00263   char **p = map (sctx, prop);
00264
00265   if (p)
00266     return *p;
00267
00268   return NULL;
00269 }
00270
00290 const char *
00291 gsasl_property_get (Gsasl_session *sctx, Gsasl_property prop)
00292 {
00293   const char *p = gsasl_property_fast (sctx, prop);
00294
00295   if (!p)
00296     {
00297       gsasl_callback (NULL, sctx, prop);
00298       p = gsasl_property_fast (sctx, prop);
00299     }
00300
00301   return p;
00302 }
```

## 5.209 register.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- int gsasl_register (Gsasl *ctx, const Gsasl_mechanism *mech)

### 5.209.1 Function Documentation

#### 5.209.1.1 gsasl_register()

```
int gsasl_register (
            Gsasl * ctx,
            const Gsasl_mechanism * mech )
```

gsasl_register:

**Parameters**

| | |
|---|---|
| *ctx* | pointer to libgsasl handle. |
| *mech* | plugin structure with information about plugin. |

This function initialize given mechanism, and if successful, add it to the list of plugins that is used by the library.

Return value: GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Since: 0.2.0

Definition at line 38 of file register.c.

## 5.210 register.c

Go to the documentation of this file.

```
00001 /* register.c --- Initialize and register SASL plugin in global context.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00037 int
00038 gsasl_register (Gsasl *ctx, const Gsasl_mechanism *mech)
00039 {
00040   Gsasl_mechanism *tmp;
00041
00042 #ifdef USE_CLIENT
00043   if (mech->client.init == NULL || mech->client.init (ctx) == GSASL_OK)
00044     {
00045       tmp = realloc (ctx->client_mechs,
00046                      sizeof (*ctx->client_mechs) * (ctx->n_client_mechs + 1));
00047       if (tmp == NULL)
00048         return GSASL_MALLOC_ERROR;
00049
00050       memcpy (&tmp[ctx->n_client_mechs], mech, sizeof (*mech));
00051
00052       ctx->client_mechs = tmp;
00053       ctx->n_client_mechs++;
00054     }
00055 #endif
00056
00057 #ifdef USE_SERVER
00058   if (mech->server.init == NULL || mech->server.init (ctx) == GSASL_OK)
00059     {
00060       tmp = realloc (ctx->server_mechs,
00061                      sizeof (*ctx->server_mechs) * (ctx->n_server_mechs + 1));
00062       if (tmp == NULL)
00063         return GSASL_MALLOC_ERROR;
00064
00065       memcpy (&tmp[ctx->n_server_mechs], mech, sizeof (*mech));
00066
00067       ctx->server_mechs = tmp;
00068       ctx->n_server_mechs++;
00069     }
00070 #endif
00071
00072   return GSASL_OK;
00073 }
```

# 5.211 saslprep.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- int gsasl_saslprep (const char *in, Gsasl_saslprep_flags flags _GL_UNUSED, char **out, int *stringpreprc _GL_UNUSED)

## 5.211.1 Function Documentation

### 5.211.1.1 gsasl_saslprep()

```
int gsasl_saslprep (
            const char * in,
            Gsasl_saslprep_flags flags _GL_UNUSED,
            char ** out,
            int *stringpreprc _GL_UNUSED )
```

Definition at line 86 of file saslprep.c.

# 5.212 saslprep.c

Go to the documentation of this file.

```
00001 /* saslprep.c --- Internationalized SASL string processing.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00025 #if HAVE_LIBIDN
00026
00027 # include <stringprep.h>
00028
00029 # if defined HAVE_PR29_H && defined HAVE_PR29_8Z
00030 #  include <pr29.h>
00031
00032 # endif
00033
00049 int
00050 gsasl_saslprep (const char *in, Gsasl_saslprep_flags flags,
00051                 char **out, int *stringpreprc)
00052 {
00053   int rc;
```

```
00054
00055   rc = stringprep_profile (in, out, "SASLprep",
00056                            (flags & GSASL_ALLOW_UNASSIGNED)
00057                            ? STRINGPREP_NO_UNASSIGNED : 0);
00058
00059   if (stringpreprc)
00060     *stringpreprc = rc;
00061
00062   if (rc != STRINGPREP_OK)
00063     {
00064       *out = NULL;
00065       return GSASL_SASLPREP_ERROR;
00066     }
00067
00068 # if defined HAVE_PR29_8Z && defined HAVE_PR29_H
00069   if (pr29_8z (*out) != PR29_SUCCESS)
00070     {
00071       free (*out);
00072       *out = NULL;
00073       if (stringpreprc)
00074         *stringpreprc = STRINGPREP_NFKC_FAILED;
00075
00076       return GSASL_SASLPREP_ERROR;
00077     }
00078 # endif
00079
00080   return GSASL_OK;
00081 }
00082
00083 #else /* HAVE_LIBIDN */
00084
00085 int
00086 gsasl_saslprep (const char *in, Gsasl_saslprep_flags flags _GL_UNUSED,
00087                 char **out, int *stringpreprc _GL_UNUSED)
00088 {
00089   size_t i, inlen = strlen (in);
00090
00091   for (i = 0; i < inlen; i++)
00092     if (in[i] & 0x80)
00093       {
00094         *out = NULL;
00095         return GSASL_SASLPREP_ERROR;
00096       }
00097
00098   *out = malloc (inlen + 1);
00099   if (!*out)
00100     return GSASL_MALLOC_ERROR;
00101   strcpy (*out, in);
00102
00103   return GSASL_OK;
00104 }
00105
00106 #endif
```

## 5.213 suggest.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- int gsasl_mechanism_name_p (const char *mech)
- const char * gsasl_client_suggest_mechanism (Gsasl *ctx, const char *mechlist)

### 5.213.1 Function Documentation

#### 5.213.1.1 gsasl_client_suggest_mechanism()

```
const char * gsasl_client_suggest_mechanism (
            Gsasl * ctx,
            const char * mechlist )
```

gsasl_client_suggest_mechanism:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *mechlist* | input character array with SASL mechanism names, separated by invalid characters (e.g. SPC). |

Given a list of mechanisms, suggest which to use.

Return value: Returns name of "best" SASL mechanism supported by the libgsasl client which is present in the input string, or NULL if no supported mechanism is found.

Definition at line 87 of file suggest.c.

#### 5.213.1.2 gsasl_mechanism_name_p()

```
int gsasl_mechanism_name_p (
            const char * mech )
```

gsasl_mechanism_name_p:

**Parameters**

| | |
|---|---|
| *mech* | input variable with mechanism name string. |

Check if the mechanism name string @mech follows syntactical rules. It does not check that the name is registered with IANA. It does not check that the mechanism name is actually implemented and supported.

SASL mechanisms are named by strings, from 1 to 20 characters in length, consisting of upper-case letters, digits, hyphens, and/or underscores.

Returns: non-zero when mechanism name string @mech conforms to rules, zero when it does not meet the requirements.

Since: 2.0.0

Definition at line 52 of file suggest.c.

## 5.214 suggest.c

Go to the documentation of this file.

```
00001 /* suggest.c --- Suggest client mechanism to use, from a set of mechanisms.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
```

```
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00025 /*
00026  * _GSASL_VALID_MECHANISM_CHARACTERS:
00027  *
00028  * A zero-terminated character array, or string, with all ASCII
00029  * characters that may be used within a SASL mechanism name.
00030  **/
00031 static const char *_GSASL_VALID_MECHANISM_CHARACTERS =
00032   "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-_";
00033
00051 int
00052 gsasl_mechanism_name_p (const char *mech)
00053 {
00054   size_t l;
00055
00056   if (mech == NULL)
00057     return 0;
00058
00059   l = strlen (mech);
00060
00061   if (l < GSASL_MIN_MECHANISM_SIZE)
00062     return 0;
00063
00064   if (l > GSASL_MAX_MECHANISM_SIZE)
00065     return 0;
00066
00067   while (*mech)
00068     if (strchr (_GSASL_VALID_MECHANISM_CHARACTERS, *mech++) == NULL)
00069       return 0;
00070
00071   return 1;
00072 }
00073
00086 const char *
00087 gsasl_client_suggest_mechanism (Gsasl *ctx, const char *mechlist)
00088 {
00089   size_t mechlist_len, target_mech, i;
00090
00091   mechlist_len = mechlist ? strlen (mechlist) : 0;
00092   target_mech = ctx->n_client_mechs;      /* ~ no target */
00093
00094   for (i = 0; i < mechlist_len;)
00095     {
00096       size_t len;
00097
00098       len = strspn (mechlist + i, _GSASL_VALID_MECHANISM_CHARACTERS);
00099       if (!len)
00100         ++i;
00101       else
00102         {
00103           size_t j;
00104
00105           /* Assumption: the mechs array is sorted by preference
00106            * from low security to high security. */
00107           for (j = (target_mech < ctx->n_client_mechs ? target_mech + 1 : 0);
00108                j < ctx->n_client_mechs; ++j)
00109             {
00110               if ((strlen (ctx->client_mechs[j].name) == len)
00111                   && (strncmp (ctx->client_mechs[j].name, mechlist + i,
00112                                len) == 0))
00113                 {
00114                   Gsasl_session *sctx;
00115
00116                   if (gsasl_client_start (ctx, ctx->client_mechs[j].name,
00117                                           &sctx) == GSASL_OK)
00118                     {
00119                       gsasl_finish (sctx);
00120                       target_mech = j;
00121                     }
00122
00123                   break;
00124                 }
00125             }
00126           i += len + 1;
00127         }
00128     }
00129
00130   return target_mech < ctx->n_client_mechs ?
00131     ctx->client_mechs[target_mech].name : NULL;
00132 }
```

## 5.215 supportp.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- int gsasl_client_support_p (Gsasl *ctx, const char *name)
- int gsasl_server_support_p (Gsasl *ctx, const char *name)

### 5.215.1 Function Documentation

#### 5.215.1.1 gsasl_client_support_p()

```
int gsasl_client_support_p (
            Gsasl * ctx,
            const char * name )
```

gsasl_client_support_p:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *name* | name of SASL mechanism. |

Decide whether there is client-side support for a specified mechanism.

Return value: Returns 1 if the libgsasl client supports the named mechanism, otherwise 0.

Definition at line 49 of file supportp.c.

#### 5.215.1.2 gsasl_server_support_p()

```
int gsasl_server_support_p (
            Gsasl * ctx,
            const char * name )
```

gsasl_server_support_p:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *name* | name of SASL mechanism. |

Decide whether there is server-side support for a specified mechanism.

Return value: Returns 1 if the libgsasl server supports the named mechanism, otherwise 0.

Definition at line 66 of file supportp.c.

## 5.216 supportp.c

Go to the documentation of this file.

```
00001 /* supportp.c --- Tell if a specific mechanism is supported.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00025 static int
00026 _gsasl_support_p (Gsasl_mechanism *mechs, size_t n_mechs, const char *name)
00027 {
00028   size_t i;
00029
00030   for (i = 0; i < n_mechs; i++)
00031     if (name && strcmp (name, mechs[i].name) == 0)
00032       return 1;
00033
00034   return 0;
00035 }
00036
00048 int
00049 gsasl_client_support_p (Gsasl *ctx, const char *name)
00050 {
00051   return _gsasl_support_p (ctx->client_mechs, ctx->n_client_mechs, name);
00052 }
00053
00065 int
00066 gsasl_server_support_p (Gsasl *ctx, const char *name)
00067 {
00068   return _gsasl_support_p (ctx->server_mechs, ctx->n_server_mechs, name);
00069 }
```

## 5.217 version.c File Reference

```
#include <config.h>
#include "internal.h"
#include <string.h>
```

### Functions

- const char * gsasl_check_version (const char *req_version)

### 5.217.1 Function Documentation

#### 5.217.1.1 gsasl_check_version()

```
const char * gsasl_check_version (
            const char * req_version )
```

gsasl_check_version:

**Parameters**

| | |
|---|---|
| *req_version* | version string to compare with, or NULL. |

Check GNU SASL Library version.

See GSASL_VERSION for a suitable @req_version string.

This function is one of few in the library that can be used without a successful call to gsasl_init().

Return value: Check that the version of the library is at minimum the one given as a string in @req_version and return the actual version string of the library; return NULL if the condition is not met. If NULL is passed to this function no check is done and only the version string is returned.

Definition at line 45 of file version.c.

## 5.218 version.c

Go to the documentation of this file.

```
00001 /* version.c -- Version handling.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00025 #include <string.h>             /* for strverscmp */
00026
00044 const char *
00045 gsasl_check_version (const char *req_version)
00046 {
00047   if (!req_version || strverscmp (req_version, GSASL_VERSION) <= 0)
00048     return GSASL_VERSION;
00049
00050   return NULL;
00051 }
```

## 5.219 xcode.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- int gsasl_encode (Gsasl_session *sctx, const char *input, size_t input_len, char **output, size_t *output_↩ len)
- int gsasl_decode (Gsasl_session *sctx, const char *input, size_t input_len, char **output, size_t *output_↩ len)

## 5.219.1 Function Documentation

### 5.219.1.1 gsasl_decode()

```
int gsasl_decode (
              Gsasl_session * sctx,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

gsasl_decode:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |
| *input* | input byte array. |
| *input_len* | size of input byte array. |
| *output* | newly allocated output byte array. |
| *output_len* | pointer to output variable with size of output byte array. |

Decode data according to negotiated SASL mechanism. This might mean that data is integrity or privacy protected.

The @output buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl↩ _free(@output).

Return value: Returns GSASL_OK if encoding was successful, otherwise an error code.

Definition at line 98 of file xcode.c.

### 5.219.1.2 gsasl_encode()

```
int gsasl_encode (
              Gsasl_session * sctx,
              const char * input,
              size_t input_len,
              char ** output,
              size_t * output_len )
```

gsasl_encode:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |
| *input* | input byte array. |
| *input_len* | size of input byte array. |
| *output* | newly allocated output byte array. |
| *output_len* | pointer to output variable with size of output byte array. |

Encode data according to negotiated SASL mechanism. This might mean that data is integrity or privacy protected.

The @output buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl↩ _free(@output).

Return value: Returns GSASL_OK if encoding was successful, otherwise an error code.

Definition at line 65 of file xcode.c.

## 5.220 xcode.c

Go to the documentation of this file.

```
00001 /* xcode.c --- Encode and decode application payload in libgsasl session.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00025 static int
00026 _gsasl_code (Gsasl_session *sctx,
00027              Gsasl_code_function code,
00028              const char *input, size_t input_len,
00029              char **output, size_t *output_len)
00030 {
00031
00032   if (code == NULL)
00033     {
00034       *output_len = input_len;
00035       *output = malloc (*output_len);
00036       if (!*output)
00037         return GSASL_MALLOC_ERROR;
00038
00039       memcpy (*output, input, input_len);
00040       return GSASL_OK;
00041     }
00042
00043   return code (sctx, sctx->mech_data, input, input_len, output, output_len);
00044 }
00045
00064 int
00065 gsasl_encode (Gsasl_session *sctx,
00066               const char *input, size_t input_len,
00067               char **output, size_t *output_len)
00068 {
00069   Gsasl_code_function code;
00070
00071   if (sctx->clientp)
00072     code = sctx->mech->client.encode;
00073   else
00074     code = sctx->mech->server.encode;
00075
00076   return _gsasl_code (sctx, code, input, input_len, output, output_len);
00077 }
00078
00097 int
00098 gsasl_decode (Gsasl_session *sctx,
00099               const char *input, size_t input_len,
00100               char **output, size_t *output_len)
00101 {
00102   Gsasl_code_function code;
```

```
00103
00104   if (sctx->clientp)
00105     code = sctx->mech->client.decode;
00106   else
00107     code = sctx->mech->server.decode;
00108
00109   return _gsasl_code (sctx, code, input, input_len, output, output_len);
00110 }
```

## 5.221 xfinish.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- void gsasl_finish (Gsasl_session *sctx)

### 5.221.1 Function Documentation

#### 5.221.1.1 gsasl_finish()

```
void gsasl_finish (
            Gsasl_session * sctx )
```

gsasl_finish:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |

Destroy a libgsasl client or server handle. The handle must not be used with other libgsasl functions after this call.

Definition at line 33 of file xfinish.c.

## 5.222 xfinish.c

Go to the documentation of this file.

```
00001 /* xfinish.c --- Finish libgsasl session.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
```

```
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00032 void
00033 gsasl_finish (Gsasl_session *sctx)
00034 {
00035   if (sctx == NULL)
00036     return;
00037
00038   if (sctx->clientp)
00039     {
00040       if (sctx->mech && sctx->mech->client.finish)
00041         sctx->mech->client.finish (sctx, sctx->mech_data);
00042     }
00043   else
00044     {
00045       if (sctx->mech && sctx->mech->server.finish)
00046         sctx->mech->server.finish (sctx, sctx->mech_data);
00047     }
00048
00049   free (sctx->anonymous_token);
00050   free (sctx->authid);
00051   free (sctx->authzid);
00052   free (sctx->password);
00053   free (sctx->passcode);
00054   free (sctx->pin);
00055   free (sctx->suggestedpin);
00056   free (sctx->service);
00057   free (sctx->hostname);
00058   free (sctx->gssapi_display_name);
00059   free (sctx->realm);
00060   free (sctx->digest_md5_hashed_password);
00061   free (sctx->qops);
00062   free (sctx->qop);
00063   free (sctx->scram_iter);
00064   free (sctx->scram_salt);
00065   free (sctx->scram_salted_password);
00066   free (sctx->scram_serverkey);
00067   free (sctx->scram_storedkey);
00068   free (sctx->cb_tls_unique);
00069   free (sctx->cb_tls_exporter);
00070   free (sctx->saml20_idp_identifier);
00071   free (sctx->saml20_redirect_url);
00072   free (sctx->openid20_redirect_url);
00073   free (sctx->openid20_outcome_data);
00074   /* If you add anything here, remember to change change map() in
00075      property.c and Gsasl_session in internal.h.  */
00076
00077   free (sctx);
00078 }
```

## 5.223 xstart.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- int gsasl_client_start (Gsasl *ctx, const char *mech, Gsasl_session **sctx)
- int gsasl_server_start (Gsasl *ctx, const char *mech, Gsasl_session **sctx)

### 5.223.1 Function Documentation

#### 5.223.1.1 gsasl_client_start()

```
int gsasl_client_start (
             Gsasl * ctx,
```

```
            const char * mech,
            Gsasl_session ** sctx )
```

gsasl_client_start:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *mech* | name of SASL mechanism. |
| *sctx* | pointer to client handle. |

This functions initiates a client SASL authentication. This function must be called before any other gsasl_client_*() function is called.

Return value: Returns GSASL_OK if successful, or error code.

Definition at line 119 of file xstart.c.

#### 5.223.1.2 gsasl_server_start()

```
int gsasl_server_start (
            Gsasl * ctx,
            const char * mech,
            Gsasl_session ** sctx )
```

gsasl_server_start:

**Parameters**

| | |
|---|---|
| *ctx* | libgsasl handle. |
| *mech* | name of SASL mechanism. |
| *sctx* | pointer to server handle. |

This functions initiates a server SASL authentication. This function must be called before any other gsasl_server_*() function is called.

Return value: Returns GSASL_OK if successful, or error code.

Definition at line 137 of file xstart.c.

## 5.224 xstart.c

Go to the documentation of this file.

```
00001 /* xstart.c --- Start libgsasl session.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
```

```
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00025 static Gsasl_mechanism *
00026 find_mechanism (const char *mech, size_t n_mechs, Gsasl_mechanism *mechs)
00027 {
00028   size_t i;
00029
00030   if (mech == NULL)
00031     return NULL;
00032
00033   for (i = 0; i < n_mechs; i++)
00034     if (strcmp (mech, mechs[i].name) == 0)
00035       return &mechs[i];
00036
00037   return NULL;
00038 }
00039
00040 static int
00041 setup (Gsasl *ctx,
00042        const char *mech,
00043        Gsasl_session *sctx,
00044        size_t n_mechs, Gsasl_mechanism *mechs, int clientp)
00045 {
00046   Gsasl_mechanism *mechptr = NULL;
00047   int res;
00048
00049   mechptr = find_mechanism (mech, n_mechs, mechs);
00050   if (mechptr == NULL)
00051     return GSASL_UNKNOWN_MECHANISM;
00052
00053   sctx->ctx = ctx;
00054   sctx->mech = mechptr;
00055   sctx->clientp = clientp;
00056
00057   if (clientp)
00058     {
00059       if (sctx->mech->client.start)
00060         res = sctx->mech->client.start (sctx, &sctx->mech_data);
00061       else if (!sctx->mech->client.step)
00062         res = GSASL_NO_CLIENT_CODE;
00063       else
00064         res = GSASL_OK;
00065     }
00066   else
00067     {
00068       if (sctx->mech->server.start)
00069         res = sctx->mech->server.start (sctx, &sctx->mech_data);
00070       else if (!sctx->mech->server.step)
00071         res = GSASL_NO_SERVER_CODE;
00072       else
00073         res = GSASL_OK;
00074     }
00075   if (res != GSASL_OK)
00076     return res;
00077
00078   return GSASL_OK;
00079 }
00080
00081 static int
00082 start (Gsasl *ctx,
00083        const char *mech,
00084        Gsasl_session **sctx,
00085        size_t n_mechs, Gsasl_mechanism *mechs, int clientp)
00086 {
00087   Gsasl_session *out;
00088   int res;
00089
00090   out = calloc (1, sizeof (*out));
00091   if (out == NULL)
00092     return GSASL_MALLOC_ERROR;
00093
00094   res = setup (ctx, mech, out, n_mechs, mechs, clientp);
00095   if (res != GSASL_OK)
00096     {
00097       gsasl_finish (out);
00098       return res;
```

```
00099     }
00100
00101   *sctx = out;
00102
00103   return GSASL_OK;
00104 }
00105
00118 int
00119 gsasl_client_start (Gsasl *ctx, const char *mech, Gsasl_session **sctx)
00120 {
00121   return start (ctx, mech, sctx, ctx->n_client_mechs, ctx->client_mechs, 1);
00122 }
00123
00136 int
00137 gsasl_server_start (Gsasl *ctx, const char *mech, Gsasl_session **sctx)
00138 {
00139   return start (ctx, mech, sctx, ctx->n_server_mechs, ctx->server_mechs, 0);
00140 }
```

## 5.225 xstep.c File Reference

```
#include <config.h>
#include "internal.h"
```

### Functions

- int gsasl_step (Gsasl_session *sctx, const char *input, size_t input_len, char **output, size_t *output_len)
- int gsasl_step64 (Gsasl_session *sctx, const char *b64input, char **b64output)

### 5.225.1 Function Documentation

#### 5.225.1.1 gsasl_step()

```
int gsasl_step (
            Gsasl_session * sctx,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )
```

gsasl_step:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl session handle. |
| *input* | input byte array. |
| *input_len* | size of input byte array. |
| *output* | newly allocated output byte array. |
| *output_len* | pointer to output variable with size of output byte array. |

Perform one step of SASL authentication. This reads data from the other end (from @input and @input_len), processes it (potentially invoking callbacks to the application), and writes data to server (into newly allocated variable @output and @output_len that indicate the length of @output).

The contents of the @output buffer is unspecified if this functions returns anything other than GSASL_OK or GSASL_NEEDS_MORE. If this function return GSASL_OK or GSASL_NEEDS_MORE, however, the @output buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl_free(@output).

Return value: Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

Definition at line 51 of file xstep.c.

#### 5.225.1.2 gsasl_step64()

```
int gsasl_step64 (
            Gsasl_session * sctx,
            const char * b64input,
            char ** b64output )
```

gsasl_step64:

**Parameters**

| | |
|---|---|
| *sctx* | libgsasl client handle. |
| *b64input* | input base64 encoded byte array. |
| *b64output* | newly allocated output base64 encoded byte array. |

This is a simple wrapper around gsasl_step() that base64 decodes the input and base64 encodes the output.

The contents of the @b64output buffer is unspecified if this functions returns anything other than GSASL↩ _OK or GSASL_NEEDS_MORE. If this function return GSASL_OK or GSASL_NEEDS_MORE, however, the @b64output buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl↩ _free(@b64output).

Return value: Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

Definition at line 86 of file xstep.c.

## 5.226 xstep.c

Go to the documentation of this file.

```
00001 /* xstep.c --- Perform one SASL authentication step.
00002  * Copyright (C) 2002-2026 Simon Josefsson
00003  *
00004  * This file is part of GNU SASL Library.
00005  *
00006  * GNU SASL Library is free software; you can redistribute it and/or
00007  * modify it under the terms of the GNU Lesser General Public License
00008  * as published by the Free Software Foundation; either version 2.1 of
00009  * the License, or (at your option) any later version.
00010  *
00011  * GNU SASL Library is distributed in the hope that it will be useful,
00012  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00013  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
00014  * Lesser General Public License for more details.
00015  *
00016  * You should have received a copy of the GNU Lesser General Public
00017  * License along with GNU SASL Library; if not, see
00018  * <https://www.gnu.org/licenses/>.
00019  *
```

```
00020  */
00021
00022 #include <config.h>
00023 #include "internal.h"
00024
00050 int
00051 gsasl_step (Gsasl_session *sctx,
00052             const char *input, size_t input_len,
00053             char **output, size_t *output_len)
00054 {
00055   Gsasl_step_function step;
00056
00057   if (sctx->clientp)
00058     step = sctx->mech->client.step;
00059   else
00060     step = sctx->mech->server.step;
00061
00062   return step (sctx, sctx->mech_data, input, input_len, output, output_len);
00063 }
00064
00085 int
00086 gsasl_step64 (Gsasl_session *sctx, const char *b64input, char **b64output)
00087 {
00088   size_t input_len = 0, output_len = 0;
00089   char *input = NULL, *output = NULL;
00090   int res;
00091
00092   if (b64input)
00093     {
00094       res = gsasl_base64_from (b64input, strlen (b64input),
00095                                &input, &input_len);
00096       if (res != GSASL_OK)
00097         return GSASL_BASE64_ERROR;
00098     }
00099
00100   res = gsasl_step (sctx, input, input_len, &output, &output_len);
00101
00102   free (input);
00103
00104   if (res == GSASL_OK || res == GSASL_NEEDS_MORE)
00105     {
00106       int tmpres = gsasl_base64_to (output, output_len, b64output, NULL);
00107
00108       free (output);
00109
00110       if (tmpres != GSASL_OK)
00111         return tmpres;
00112     }
00113
00114   return res;
00115 }
```

# Index