gsasl

2.2.1

Generated by Doxygen 1.9.1

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

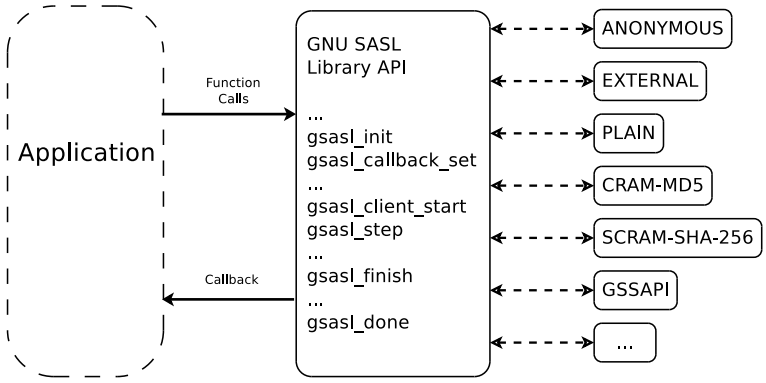

Figure 1.1 Logical overview

1.3 Control flow in application using the library

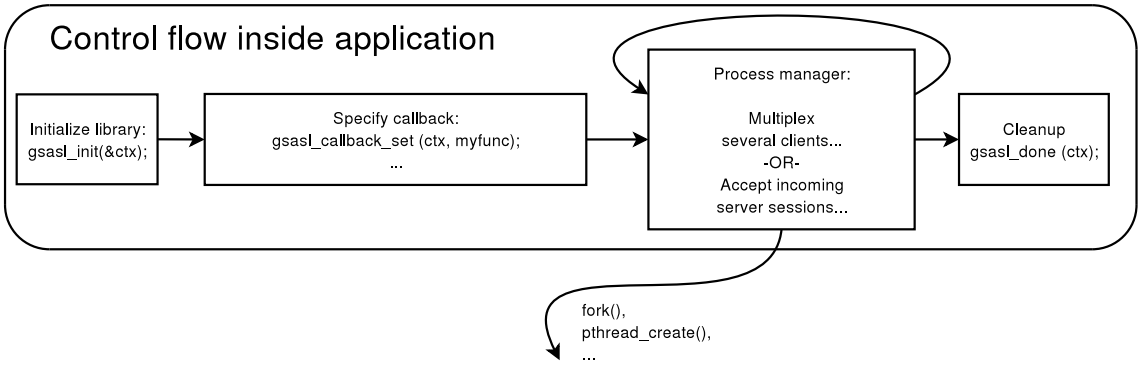

Figure 1.2 Control flow

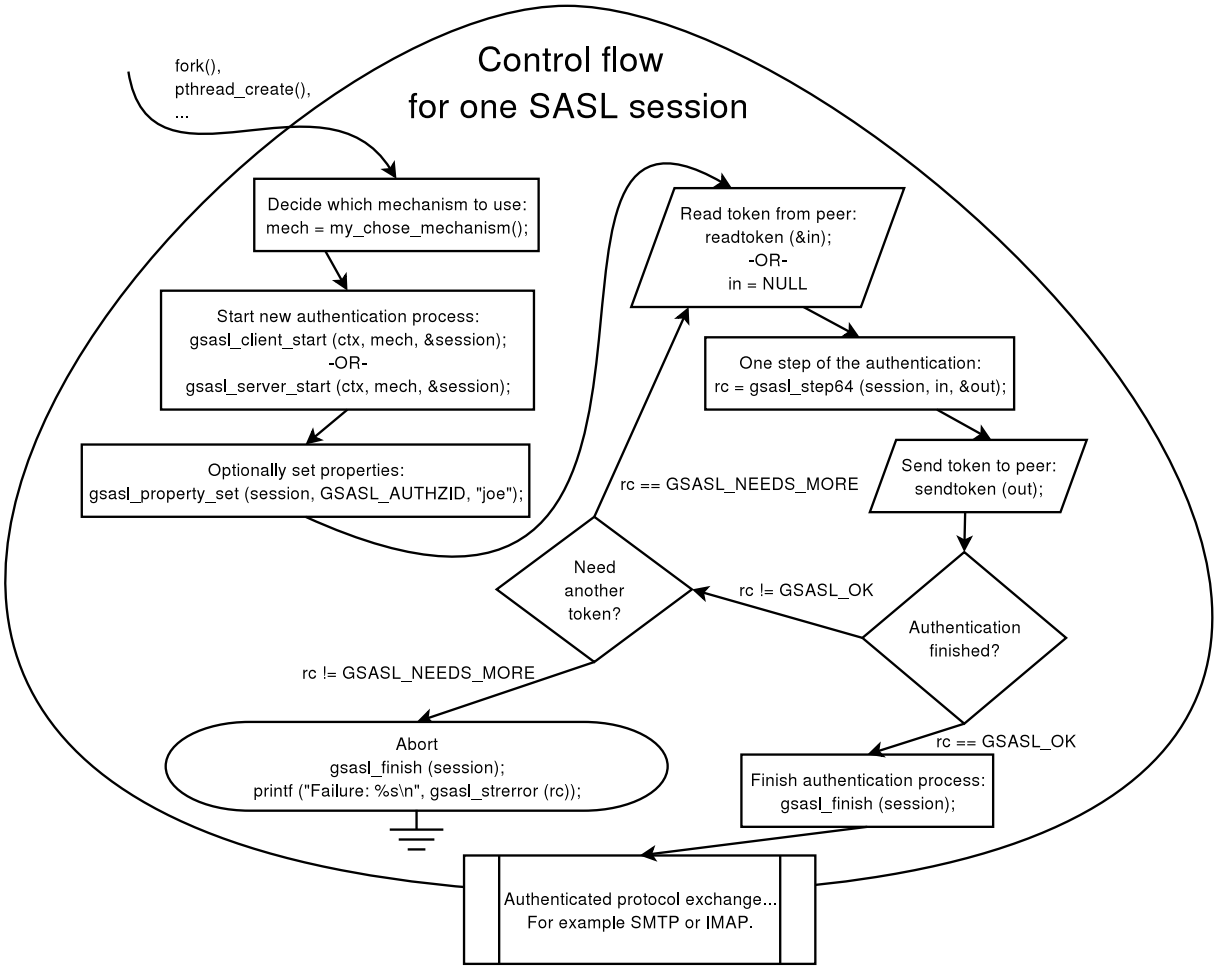

Figure 1.3 Control flow

1.4 Examples

```
/* client.c --- Example SASL client.
 * Copyright (C) 2004-2024 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 sucessful, 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) 2004-2024 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) 2004-2024 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) 2004-2024 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 50 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 59 of file digest-md5/client.c.

4.1.2.2 finish

digest_md5_finish _Gsasl_digest_md5_client_state::finish

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

4.1.2.3 kcc

char _Gsasl_digest_md5_client_state::kcc[DIGEST_MD5_LENGTH]

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

4.1.2.4 kcs

char _Gsasl_digest_md5_client_state::kcs[DIGEST_MD5_LENGTH]

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

4.1.2.5 kic

char _Gsasl_digest_md5_client_state::kic[DIGEST_MD5_LENGTH]

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

4.1.2.6 kis

char _Gsasl_digest_md5_client_state::kis[DIGEST_MD5_LENGTH]

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

4.1.2.7 readseqnum

unsigned long _Gsasl_digest_md5_client_state::readseqnum

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

4.1.2.8 response

```
digest_md5_response _Gsasl_digest_md5_client_state::response
```

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

4.1.2.9 secret

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

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

4.1.2.10 sendseqnum

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

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

4.1.2.11 step

```
int _Gsasl_digest_md5_client_state::step
```

Definition at line 52 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 51 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 60 of file digest-md5/server.c.

4.2.2.2 finish

digest_md5_finish _Gsasl_digest_md5_server_state::finish

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

4.2.2.3 kcc

char _Gsasl_digest_md5_server_state::kcc[DIGEST_MD5_LENGTH]

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

4.2.2.4 kcs

char _Gsasl_digest_md5_server_state::kcs[DIGEST_MD5_LENGTH]

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

4.2.2.5 kic

char _Gsasl_digest_md5_server_state::kic[DIGEST_MD5_LENGTH]

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

4.2.2.6 kis

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

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

4.2.2.7 readseqnum

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

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

4.2.2.8 response

```
digest_md5_response _Gsasl_digest_md5_server_state::response
```

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

4.2.2.9 secret

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

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

4.2.2.10 sendseqnum

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

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

4.2.2.11 step

```
int _Gsasl_digest_md5_server_state::step
```

Definition at line 53 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 37 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 42 of file gs2/client.c.

4.3.2.3 mech_oid

```
gss_OID _gsasl_gs2_client_state::mech_oid
```

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

4.3.2.4 service

```
gss_name_t _gsasl_gs2_client_state::service
```

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

4.3.2.5 step

```
int _gsasl_gs2_client_state::step
```

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

4.3.2.6 token

```
gss_buffer_desc _gsasl_gs2_client_state::token
```

Definition at line 44 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 41 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 45 of file gs2/server.c.

4.4.2.3 context

```
gss_ctx_id_t _Gsasl_gs2_server_state::context
```

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

4.4.2.4 cred

```
gss_cred_id_t _Gsasl_gs2_server_state::cred
```

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

4.4.2.5 mech_oid

```
gss_OID _Gsasl_gs2_server_state::mech_oid
```

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

4.4.2.6 step

```
int _Gsasl_gs2_server_state::step
```

Definition at line 44 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 37 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 41 of file gssapi/client.c.

4.5.2.2 qop

```
gss_qop_t _Gsasl_gssapi_client_state::qop
```

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

4.5.2.3 service

```
gss_name_t _Gsasl_gssapi_client_state::service
```

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

4.5.2.4 step

```
int _Gsasl_gssapi_client_state::step
```

Definition at line 39 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 37 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 40 of file gssapi/server.c.

4.6.2.2 context

```
gss_ctx_id_t _Gsasl_gssapi_server_state::context
```

Definition at line 42 of file gssapi/server.c.

4.6.2.3 cred

```
gss_cred_id_t _Gsasl_gssapi_server_state::cred
```

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

4.6.2.4 step

```
int _Gsasl_gssapi_server_state::step
```

Definition at line 39 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 34 of file login/client.c.

4.7.2 Field Documentation

4.7.2.1 step

```
int _Gsasl_login_client_state::step
```

Definition at line 36 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 34 of file login/server.c.

4.8.2 Field Documentation

4.8.2.1 password

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

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

4.8.2.2 step

```
int _Gsasl_login_server_state::step
```

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

4.8.2.3 username

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

Definition at line 37 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 36 of file ntlm.c.

4.9.2 Field Documentation

4.9.2.1 step

```
int _Gsasl_ntlm_state::step
```

Definition at line 38 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 82 of file digest-md5/tokens.h.

4.10.2 Field Documentation

4.10.2.1 ciphers

`int digest_md5_challenge::ciphers`

Definition at line 91 of file digest-md5/tokens.h.

4.10.2.2 nonce

`char* digest_md5_challenge::nonce`

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

4.10.2.3 nrealms

`size_t digest_md5_challenge::nrealms`

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

4.10.2.4 qops

```
int digest_md5_challenge::qops
```

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

4.10.2.5 realms

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

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

4.10.2.6 servermaxbuf

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

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

4.10.2.7 stale

```
int digest_md5_challenge::stale
```

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

4.10.2.8 utf8

```
int digest_md5_challenge::utf8
```

Definition at line 90 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 146 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 148 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 126 of file digest-md5/tokens.h.

4.12.2 Field Documentation

4.12.2.1 authzid

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

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

4.12.2.2 cipher

```
digest_md5_cipher digest_md5_response::cipher
```

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

4.12.2.3 clientmaxbuf

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

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

4.12.2.4 cnonce

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

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

4.12.2.5 digesturi

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

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

4.12.2.6 nc

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

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

4.12.2.7 nonce

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

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

4.12.2.8 qop

```
digest_md5_qop digest_md5_response::qop
```

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

4.12.2.9 realm

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

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

4.12.2.10 response

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

Definition at line 139 of file digest-md5/tokens.h.

4.12.2.11 username

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

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

4.12.2.12 utf8

```
int digest_md5_response::utf8
```

Definition at line 136 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 36 of file internal.h.

4.13.2 Field Documentation

4.13.2.1 application_hook

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

Definition at line 44 of file internal.h.

4.13.2.2 cb

```
Gsasl_callback_function Gsasl::cb
```

Definition at line 43 of file internal.h.

4.13.2.3 client_mechs

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

Definition at line 39 of file internal.h.

4.13.2.4 n_client_mechs

```
size_t Gsasl::n_client_mechs
```

Definition at line 38 of file internal.h.

4.13.2.5 n_server_mechs

```
size_t Gsasl::n_server_mechs
```

Definition at line 40 of file internal.h.

4.13.2.6 server_mechs

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

Definition at line 41 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 171 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 173 of file gsasl-mech.h.

4.14.2.2 name

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

Definition at line 173 of file gsasl-mech.h.

4.14.2.3 server

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

Definition at line 173 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().
decode	a Gsasl_code_function().

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

Definition at line 151 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 159 of file gsasl-mech.h.

4.15.2.2 done

```
Gsasl_done_function Gsasl_mechanism_functions::done
```

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

4.15.2.3 encode

```
Gsasl_code_function Gsasl_mechanism_functions::encode
```

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

4.15.2.4 finish

```
Gsasl_finish_function Gsasl_mechanism_functions::finish
```

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

4.15.2.5 init

Gsasl_init_function Gsasl_mechanism_functions::init

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

4.15.2.6 start

Gsasl_start_function Gsasl_mechanism_functions::start

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

4.15.2.7 step

Gsasl_step_function Gsasl_mechanism_functions::step

Definition at line 156 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 48 of file internal.h.

4.16.2 Field Documentation

4.16.2.1 anonymous_token

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

Definition at line 57 of file internal.h.

4.16.2.2 application_hook

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

Definition at line 54 of file internal.h.

4.16.2.3 authid

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

Definition at line 58 of file internal.h.

4.16.2.4 authzid

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

Definition at line 59 of file internal.h.

4.16.2.5 cb_tls_exporter

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

Definition at line 77 of file internal.h.

4.16.2.6 cb_tls_unique

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

Definition at line 76 of file internal.h.

4.16.2.7 clientp

```
int Gsasl_session::clientp
```

Definition at line 51 of file internal.h.

4.16.2.8 ctx

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

Definition at line 50 of file internal.h.

4.16.2.9 digest_md5_hashed_password

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

Definition at line 68 of file internal.h.

4.16.2.10 gssapi_display_name

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

Definition at line 66 of file internal.h.

4.16.2.11 hostname

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

Definition at line 65 of file internal.h.

4.16.2.12 mech

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

Definition at line 52 of file internal.h.

4.16.2.13 mech_data

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

Definition at line 53 of file internal.h.

4.16.2.14 openid20_outcome_data

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

Definition at line 81 of file internal.h.

4.16.2.15 openid20_redirect_url

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

Definition at line 80 of file internal.h.

4.16.2.16 passcode

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

Definition at line 61 of file internal.h.

4.16.2.17 password

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

Definition at line 60 of file internal.h.

4.16.2.18 pin

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

Definition at line 62 of file internal.h.

4.16.2.19 qop

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

Definition at line 70 of file internal.h.

4.16.2.20 qops

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

Definition at line 69 of file internal.h.

4.16.2.21 realm

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

Definition at line 67 of file internal.h.

4.16.2.22 saml20_idp_identifier

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

Definition at line 78 of file internal.h.

4.16.2.23 saml20_redirect_url

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

Definition at line 79 of file internal.h.

4.16.2.24 scram_iter

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

Definition at line 71 of file internal.h.

4.16.2.25 scram_salt

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

Definition at line 72 of file internal.h.

4.16.2.26 scram_salted_password

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

Definition at line 73 of file internal.h.

4.16.2.27 scram_serverkey

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

Definition at line 74 of file internal.h.

4.16.2.28 scram_storedkey

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

Definition at line 75 of file internal.h.

4.16.2.29 service

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

Definition at line 64 of file internal.h.

4.16.2.30 suggestedpin

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

Definition at line 63 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 40 of file openid20/client.c.

4.17.2 Field Documentation

4.17.2.1 step

```
int openid20_client_state::step
```

Definition at line 42 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 37 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 40 of file openid20/server.c.

4.18.2.2 step

```
int openid20_server_state::step
```

Definition at line 39 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 40 of file saml20/client.c.

4.19.2 Field Documentation

4.19.2.1 step

```
int saml20_client_state::step
```

Definition at line 42 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 37 of file saml20/server.c.

4.20.2 Field Documentation

4.20.2.1 step

```
int saml20_server_state::step
```

Definition at line 39 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 45 of file scram/tokens.h.

4.21.2 Field Documentation

4.21.2.1 cbind

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

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

4.21.2.2 nonce

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

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

4.21.2.3 proof

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

Definition at line 49 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 29 of file scram/tokens.h.

4.22.2 Field Documentation

4.22.2.1 authzid

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

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

4.22.2.2 cbflag

```
char scram_client_first::cbflag
```

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

4.22.2.3 cbname

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

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

4.22.2.4 client_nonce

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

Definition at line 35 of file scram/tokens.h.

4.22.2.5 username

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

Definition at line 34 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 47 of file scram/client.c.

4.23.2 Field Documentation

4.23.2.1 authmessage

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

Definition at line 54 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 52 of file scram/client.c.

4.23.2.4 cl

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

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

4.23.2.5 hash

```
Gsasl_hash scram_client_state::hash
```

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

4.23.2.6 plus

```
bool scram_client_state::plus
```

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

4.23.2.7 serversignature

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

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

4.23.2.8 sf

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

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

4.23.2.9 sl

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

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

4.23.2.10 step

```
int scram_client_state::step
```

Definition at line 51 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 52 of file scram/tokens.h.

4.24.2 Field Documentation

4.24.2.1 verifier

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

Definition at line 54 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 38 of file scram/tokens.h.

4.25.2 Field Documentation

4.25.2.1 iter

`size_t scram_server_first::iter`

Definition at line 42 of file scram/tokens.h.

4.25.2.2 nonce

`char* scram_server_first::nonce`

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

4.25.2.3 salt

`char* scram_server_first::salt`

Definition at line 41 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 51 of file scram/server.c.

4.26.2 Field Documentation

4.26.2.1 authmessage

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

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

4.26.2.2 cb

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

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

4.26.2.3 cbind

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

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

4.26.2.4 cblen

```
size_t scram_server_state::cblen
```

Definition at line 66 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 58 of file scram/server.c.

4.26.2.7 cl

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

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

4.26.2.8 clientproof

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

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

4.26.2.9 gs2header

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

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

4.26.2.10 hash

```
Gsasl_hash scram_server_state::hash
```

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

4.26.2.11 plus

```
bool scram_server_state::plus
```

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

4.26.2.12 serverkey

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

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

4.26.2.13 sf

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

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

4.26.2.14 sf_str

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

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

4.26.2.15 sl

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

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

4.26.2.16 snonce

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

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

4.26.2.17 step

```
int scram_server_state::step
```

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

4.26.2.18 storedkey

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

Definition at line 62 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 28 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 )
```

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 )
```

5.1.3 Variable Documentation

5.1.3.1 _gsasl_anonymous_mechanism

```
Gsasl_mechanism _gsasl_anonymous_mechanism  [extern]
```

Definition at line 28 of file anonymous/mechinfo.c.

5.2 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)

5.2.1 Function Documentation

5.2.1.1 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 75 of file base64.c.

5.2.1.2 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 45 of file base64.c.

5.2.1.3 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 144 of file base64.c.

5.2.1.4 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 111 of file base64.c.

5.3 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)

5.3.1 Function Documentation

5.3.1.1 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 71 of file callback.c.

5.3.1.2 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 120 of file callback.c.

5.3.1.3 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 100 of file callback.c.

5.3.1.4 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 45 of file callback.c.

5.3.1.5 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 160 of file callback.c.

5.3.1.6 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 140 of file callback.c.

5.4 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])

5.4.1 Macro Definition Documentation

5.4.1.1 DIGIT

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

Value:

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

Definition at line 61 of file challenge.c.

5.4.1.2 NONCELEN

```
#define NONCELEN 10
```

Definition at line 56 of file challenge.c.

5.4.1.3 TEMPLATE

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

Definition at line 57 of file challenge.c.

5.4.2 Function Documentation

5.4.2.1 cram_md5_challenge()

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

Definition at line 66 of file challenge.c.

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 26 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] )
```

Definition at line 66 of file challenge.c.

5.6 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)

5.6.1 Function Documentation

5.6.1.1 _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 32 of file anonymous/client.c.

5.7 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)

5.7.1 Function Documentation

5.7.1.1 _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 38 of file cram-md5/client.c.

5.8 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)

5.8.1 Macro Definition Documentation

5.8.1.1 CNONCE_ENTROPY_BYTES

```
#define CNONCE_ENTROPY_BYTES 16
```

Definition at line 48 of file digest-md5/client.c.

5.8.2 Typedef Documentation

5.8.2.1 _Gsasl_digest_md5_client_state

```
typedef struct _Gsasl_digest_md5_client_state _Gsasl_digest_md5_client_state
```

Definition at line 1 of file digest-md5/client.c.

5.8.3 Function Documentation

5.8.3.1 _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 329 of file digest-md5/client.c.

5.8.3.2 _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 305 of file digest-md5/client.c.

5.8.3.3 _gsasl_digest_md5_client_finish()

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

Definition at line 289 of file digest-md5/client.c.

5.8.3.4 _gsasl_digest_md5_client_start()

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

Definition at line 66 of file digest-md5/client.c.

5.8.3.5 _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 98 of file digest-md5/client.c.

5.9 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)

5.9.1 Function Documentation

5.9.1.1 _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 32 of file external/client.c.

5.10 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)

5.10.1 Typedef Documentation

5.10.1.1 _gsasl_gs2_client_state

```
typedef struct _gsasl_gs2_client_state _gsasl_gs2_client_state
```

Definition at line 1 of file gs2/client.c.

5.10.2 Function Documentation

5.10.2.1 _gsasl_gs2_client_finish()

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

Definition at line 316 of file gs2/client.c.

5.10.2.2 _gsasl_gs2_client_start()

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

Definition at line 52 of file gs2/client.c.

5.10.2.3 _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 235 of file gs2/client.c.

5.11 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)

5.11.1 Typedef Documentation

5.11.1.1 _Gsasl_gssapi_client_state

```
typedef struct _Gsasl_gssapi_client_state _Gsasl_gssapi_client_state
```

Definition at line 1 of file gssapi/client.c.

5.11.2 Function Documentation

5.11.2.1 _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 322 of file gssapi/client.c.

5.11.2.2 _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 267 of file gssapi/client.c.

5.11.2.3 _gsasl_gssapi_client_finish()

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

Definition at line 249 of file gssapi/client.c.

5.11.2.4 _gsasl_gssapi_client_start()

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

Definition at line 47 of file gssapi/client.c.

5.11.2.5 _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 66 of file gssapi/client.c.

5.12 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)

5.12.1 Function Documentation

5.12.1.1 _gsasl_login_client_finish()

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

Definition at line 103 of file login/client.c.

5.12.1.2 _gsasl_login_client_start()

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

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

5.12.1.3 _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 56 of file login/client.c.

5.13 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)

5.13.1 Macro Definition Documentation

5.13.1.1 ERR_PREFIX

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

5.13.2 Function Documentation

5.13.2.1 _gsasl_openid20_client_finish()

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

Definition at line 167 of file openid20/client.c.

5.13.2.2 _gsasl_openid20_client_start()

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

Definition at line 46 of file openid20/client.c.

5.13.2.3 _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 61 of file openid20/client.c.

5.14 client.c File Reference

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

Functions

- int _gasl_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)

5.14.1 Function Documentation

5.14.1.1 _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 35 of file plain/client.c.

5.15 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)

5.15.1 Function Documentation

5.15.1.1 _gsasl_saml20_client_finish()

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

Definition at line 120 of file saml20/client.c.

5.15.1.2 _gsasl_saml20_client_start()

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

Definition at line 46 of file saml20/client.c.

5.15.1.3 _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 60 of file saml20/client.c.

5.16 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)

5.16.1 Macro Definition Documentation

5.16.1.1 CNONCE_ENTROPY_BYTES

```
#define CNONCE_ENTROPY_BYTES 18
```

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

5.16.2 Function Documentation

5.16.2.1 _gsasl_scram_client_finish()

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

Definition at line 423 of file scram/client.c.

5.16.2.2 _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 125 of file scram/client.c.

5.17 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)

5.17.1 Macro Definition Documentation

5.17.1.1 PASSCODE

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

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

5.17.1.2 PIN

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

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

5.17.2 Function Documentation

5.17.2.1 _gsasl_securid_client_finish()

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

Definition at line 165 of file securid/client.c.

5.17.2.2 _gsasl_securid_client_start()

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

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

5.17.2.3 _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 54 of file securid/client.c.

5.18 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

5.18.1 Macro Definition Documentation

5.18.1.1 GSASL_CRAM_MD5_NAME

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

Definition at line 28 of file cram-md5.h.

5.18.2 Function Documentation

5.18.2.1 _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 )
```

5.18.2.2 _gsasl_cram_md5_server_finish()

```
void _gsasl_cram_md5_server_finish (
            Gsasl_session * sctx,
            void * mech_data )
```

5.18.2.3 _gsasl_cram_md5_server_start()

```
int _gsasl_cram_md5_server_start (
            Gsasl_session * sctx,
            void ** mech_data )
```

5.18.2.4 _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 )
```

Definition at line 66 of file cram-md5/server.c.

5.18.3 Variable Documentation

5.18.3.1 _gsasl_cram_md5_mechanism

```
Gsasl_mechanism _gsasl_cram_md5_mechanism  [extern]
```

Definition at line 28 of file cram-md5/mechinfo.c.

5.19 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)

5.19.1 Macro Definition Documentation

5.19.1.1 CLIENT_KEY

```
#define CLIENT_KEY "Client Key"
```

5.19.1.2 SERVER_KEY

```
#define SERVER_KEY "Server Key"
```

5.19.2 Function Documentation

5.19.2.1 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 73 of file crypto.c.

5.19.2.2 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 39 of file crypto.c.

5.19.2.3 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 55 of file crypto.c.

5.19.2.4 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 156 of file crypto.c.

5.19.2.5 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 104 of file crypto.c.

5.20 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

5.20.1 Macro Definition Documentation

5.20.1.1 GSASL_DIGEST_MD5_NAME

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

Definition at line 28 of file digest-md5.h.

5.20.2 Function Documentation

5.20.2.1 _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 )
```

5.20.2.2 _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 )
```

5.20.2.3 _gsasl_digest_md5_client_finish()

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

5.20.2.4 _gsasl_digest_md5_client_start()

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

5.20.2.5 _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 )
```

Definition at line 98 of file digest-md5/client.c.

5.20.2.6 _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 )
```

5.20.2.7 _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 )
```

5.20.2.8 _gsasl_digest_md5_server_finish()

```
void _gsasl_digest_md5_server_finish (
             Gsasl_session * sctx,
             void * mech_data )
```

5.20.2.9 _gsasl_digest_md5_server_start()

```
int _gsasl_digest_md5_server_start (
             Gsasl_session * sctx,
             void ** mech_data )
```

5.20.2.10 _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 )
```

Definition at line 146 of file digest-md5/server.c.

5.20.3 Variable Documentation

5.20.3.1 _gsasl_digest_md5_mechanism

```
Gsasl_mechanism _gsasl_digest_md5_mechanism [extern]
```

Definition at line 28 of file digest-md5/mechinfo.c.

5.21 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])

5.21.1 Macro Definition Documentation

5.21.1.1 HEXCHAR

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

Definition at line 57 of file digest.c.

5.21.2 Function Documentation

5.21.2.1 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 60 of file digest.c.

5.22 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])

5.22.1 Macro Definition Documentation

5.22.1.1 CRAM_MD5_DIGEST_LEN

```
#define CRAM_MD5_DIGEST_LEN 32
```

Definition at line 29 of file digest.h.

5.22.2 Function Documentation

5.22.2.1 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 60 of file digest.c.

5.23 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)

5.23.1 Macro Definition Documentation

5.23.1.1 A2_POST

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

Definition at line 47 of file digesthmac.c.

5.23.1.2 A2_PRE

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

Definition at line 46 of file digesthmac.c.

5.23.1.3 COLON

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

Definition at line 48 of file digesthmac.c.

5.23.1.4 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 56 of file digesthmac.c.

5.23.1.5 DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING_LEN

```
#define DERIVE_CLIENT_CONFIDENTIALITY_KEY_STRING_LEN 59
```

Definition at line 58 of file digesthmac.c.

5.23.1.6 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 50 of file digesthmac.c.

5.23.1.7 DERIVE_CLIENT_INTEGRITY_KEY_STRING_LEN

```
#define DERIVE_CLIENT_INTEGRITY_KEY_STRING_LEN 65
```

Definition at line 52 of file digesthmac.c.

5.23.1.8 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 59 of file digesthmac.c.

5.23.1.9 DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING_LEN

```
#define DERIVE_SERVER_CONFIDENTIALITY_KEY_STRING_LEN 59
```

Definition at line 61 of file digesthmac.c.

5.23.1.10 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 53 of file digesthmac.c.

5.23.1.11 DERIVE_SERVER_INTEGRITY_KEY_STRING_LEN

```
#define DERIVE_SERVER_INTEGRITY_KEY_STRING_LEN 65
```

Definition at line 55 of file digesthmac.c.

5.23.1.12 HEXCHAR

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

Definition at line 40 of file digesthmac.c.

5.23.1.13 MD5LEN

```
#define MD5LEN 16
```

Definition at line 49 of file digesthmac.c.

5.23.1.14 QOP_AUTH

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

Definition at line 42 of file digesthmac.c.

5.23.1.15 QOP_AUTH_CONF

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

Definition at line 44 of file digesthmac.c.

5.23.1.16 QOP_AUTH_INT

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

Definition at line 43 of file digesthmac.c.

5.23.2 Function Documentation

5.23.2.1 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 77 of file digesthmac.c.

5.24 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)

5.24.1 Function Documentation

5.24.1.1 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 )
```

5.25 done.c File Reference

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

Functions

- void gsasl_done (Gsasl *ctx)

5.25.1 Function Documentation

5.25.1.1 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 34 of file done.c.

5.26 doxygen.c File Reference

5.27 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)

5.27.1 Macro Definition Documentation

5.27.1.1 _

```
#define _(
              String ) dgettext (PACKAGE, String)
```

Definition at line 28 of file error.c.

5.27.1.2 ERR

```
#define ERR(
              name,
              desc ) { name, #name, desc }
```

Definition at line 32 of file error.c.

5.27.1.3 gettext_noop

```
#define gettext_noop(
              String ) String
```

Definition at line 29 of file error.c.

5.27.1.4 N_

```
#define N_(
              String ) gettext_noop (String)
```

Definition at line 30 of file error.c.

5.27.2 Function Documentation

5.27.2.1 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 185 of file error.c.

5.27.2.2 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 223 of file error.c.

5.27.3 Variable Documentation

5.27.3.1 description

```
const char* description
```

Definition at line 39 of file error.c.

5.27.3.2 name

```
const char* name
```

Definition at line 38 of file error.c.

5.27.3.3 rc

```
int rc
```

Definition at line 37 of file error.c.

5.28 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

5.28.1 Macro Definition Documentation

5.28.1.1 GSASL_EXTERNAL_NAME

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

Definition at line 28 of file external.h.

5.28.2 Function Documentation

5.28.2.1 _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 )
```

5.28.2.2 _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 )
```

5.28.3 Variable Documentation

5.28.3.1 _gsasl_external_mechanism

```
Gsasl_mechanism _gsasl_external_mechanism  [extern]
```

Definition at line 28 of file external/mechinfo.c.

5.29 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)

5.29.1 Function Documentation

5.29.1.1 digest_md5_free_challenge()

```
void digest_md5_free_challenge (
            digest_md5_challenge * c )
```

Definition at line 35 of file digest-md5/free.c.

5.29.1.2 digest_md5_free_finish()

```
void digest_md5_free_finish (
            digest_md5_finish * f )
```

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

5.29.1.3 digest_md5_free_response()

```
void digest_md5_free_response (
            digest_md5_response * r )
```

Definition at line 48 of file digest-md5/free.c.

5.30 free.c File Reference

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

Functions

- void gsasl_free (void *ptr)

5.30.1 Function Documentation

5.30.1.1 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 41 of file src/free.c.

5.31 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)

5.31.1 Function Documentation

5.31.1.1 digest_md5_free_challenge()

```
void digest_md5_free_challenge (
            digest_md5_challenge * c )
```

Definition at line 35 of file digest-md5/free.c.

5.31.1.2 digest_md5_free_finish()

```
void digest_md5_free_finish (
            digest_md5_finish * f )
```

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

5.31.1.3 digest_md5_free_response()

```
void digest_md5_free_response (
            digest_md5_response * r )
```

Definition at line 48 of file digest-md5/free.c.

5.32 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)

5.32.1 Function Documentation

5.32.1.1 digest_md5_getsubopt()

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

Definition at line 44 of file getsubopt.c.

5.33 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

5.33.1 Macro Definition Documentation

5.33.1.1 GSASL_GS2_KRB5_NAME

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

Definition at line 28 of file gs2.h.

5.33.2 Function Documentation

5.33.2.1 _gsasl_gs2_client_finish()

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

Definition at line 316 of file gs2/client.c.

5.33.2.2 _gsasl_gs2_client_start()

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

Definition at line 52 of file gs2/client.c.

5.33.2.3 _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 235 of file gs2/client.c.

5.33.2.4 _gsasl_gs2_server_finish()

```
void _gsasl_gs2_server_finish (
             Gsasl_session * sctx,
             void * mech_data )
```

Definition at line 299 of file gs2/server.c.

5.33.2.5 _gsasl_gs2_server_start()

```
int _gsasl_gs2_server_start (
             Gsasl_session * sctx,
             void ** mech_data )
```

Definition at line 119 of file gs2/server.c.

5.33.2.6 _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 )
```

Definition at line 162 of file gs2/server.c.

5.33.3 Variable Documentation

5.33.3.1 _gsasl_gs2_krb5_mechanism

```
Gsasl_mechanism _gsasl_gs2_krb5_mechanism  [extern]
```

Definition at line 28 of file gs2/mechinfo.c.

5.34 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)

5.34.1 Function Documentation

5.34.1.1 gs2_get_oid()

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

Definition at line 38 of file gs2helper.c.

5.35 gs2helper.h File Reference

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

Functions

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

5.35.1 Function Documentation

5.35.1.1 gs2_get_oid()

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

Definition at line 38 of file gs2helper.c.

5.36 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)

5.36.1 Typedef Documentation

5.36.1.1 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 134 of file gsasl-mech.h.

5.36.1.2 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 58 of file gsasl-mech.h.

5.36.1.3 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 112 of file gsasl-mech.h.

5.36.1.4 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 49 of file gsasl-mech.h.

5.36.1.5 Gsasl_mechanism

```
typedef struct Gsasl_mechanism Gsasl_mechanism
```

Definition at line 134 of file gsasl-mech.h.

5.36.1.6 Gsasl_mechanism_functions

```
typedef struct Gsasl_mechanism_functions Gsasl_mechanism_functions
```

Definition at line 134 of file gsasl-mech.h.

5.36.1.7 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 70 of file gsasl-mech.h.

5.36.1.8 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 100 of file gsasl-mech.h.

5.36.2 Function Documentation

5.36.2.1 gsasl_register()

```
_GSASL_API 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 39 of file register.c.

5.37 gsasl-version.h File Reference

Macros

- #define GSASL_VERSION "2.2.1"
- #define GSASL_VERSION_MAJOR 2
- #define GSASL_VERSION_MINOR 2
- #define GSASL_VERSION_PATCH 1
- #define GSASL_VERSION_NUMBER 0x020201

5.37.1 Macro Definition Documentation

5.37.1.1 GSASL_VERSION

```
#define GSASL_VERSION "2.2.1"
```

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 42 of file gsasl-version.h.

5.37.1.2 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 53 of file gsasl-version.h.

5.37.1.3 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 64 of file gsasl-version.h.

5.37.1.4 GSASL_VERSION_NUMBER

```
#define GSASL_VERSION_NUMBER 0x020201
```

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 86 of file gsasl-version.h.

5.37.1.5 GSASL_VERSION_PATCH

```
#define GSASL_VERSION_PATCH 1
```

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 75 of file gsasl-version.h.

5.38 gsasl.h File Reference

```
#include <stdio.h>
#include <stddef.h>
#include <unistd.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)

5.38.1 Macro Definition Documentation

5.38.1.1 _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 49 of file gsasl.h.

5.38.2 Typedef Documentation

5.38.2.1 Gsasl

```
typedef struct Gsasl Gsasl
```

Gsasl:

Handle to global library context.

Definition at line 1 of file gsasl.h.

5.38.2.2 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 286 of file gsasl.h.

5.38.2.3 Gsasl_session

```
typedef struct Gsasl_session Gsasl_session
```

Gsasl_session:

Handle to SASL session context.

Definition at line 1 of file gsasl.h.

5.38.3 Enumeration Type Documentation

5.38.3.1 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 427 of file gsasl.h.

5.38.3.2 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 447 of file gsasl.h.

5.38.3.3 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 298 of file gsasl.h.

5.38.3.4 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.

Parameters

GSASL_SCRAM_SERVERKEY	Hex-encoded SCRAM ServerKey derived from users' passowrd.
GSASL_SCRAM_STOREDKEY	Hex-encoded SCRAM StoredKey derived from users' passowrd.
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.
GSASL_VALIDATE_GSSAPI	Request for validation of GSSAPI/GS2.
GSASL_VALIDATE_SECURID	Reqest for validation of SecurID.
GSASL_VALIDATE_SAML20	Reqest for validation of SAML20.
GSASL_VALIDATE_OPENID20	Reqest 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	

Enumerator

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 221 of file gsasl.h.

5.38.3.5 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 316 of file gsasl.h.

5.38.3.6 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).
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 requeted.
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.

Parameters

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	
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	

Enumerator

GSASL_GSSAPI_TEST_OID_SET_MEMBER_ERROR	
GSASL_GSSAPI_RELEASE_OID_SET_ERROR	

Definition at line 127 of file gsasl.h.

5.38.3.7 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 330 of file gsasl.h.

5.38.4 Function Documentation

5.38.4.1 gsasl_base64_from()

```
_GSASL_API 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 75 of file base64.c.

5.38.4.2 gsasl_base64_to()

```
_GSASL_API 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 45 of file base64.c.

5.38.4.3 gsasl_callback()

```
_GSASL_API 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 71 of file callback.c.

5.38.4.4 gsasl_callback_hook_get()

```
_GSASL_API 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 120 of file callback.c.

5.38.4.5 gsasl_callback_hook_set()

```
_GSASL_API 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 100 of file callback.c.

5.38.4.6 gsasl_callback_set()

```
_GSASL_API 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 45 of file callback.c.

5.38.4.7 gsasl_check_version()

```
_GSASL_API 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 46 of file version.c.

5.38.4.8 gsasl_client_mechlist()

```
_GSASL_API 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 75 of file listmech.c.

5.38.4.9 gsasl_client_start()

```
_GSASL_API 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 120 of file xstart.c.

5.38.4.10 gsasl_client_suggest_mechanism()

```
_GSASL_API 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 88 of file suggest.c.

5.38.4.11 gsasl_client_support_p()

```
_GSASL_API 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 50 of file supportp.c.

5.38.4.12 gsasl_decode()

```
_GSASL_API int gsasl_decode (
            Gsasl_session * sctx,
```

```
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )
```

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 99 of file xcode.c.

5.38.4.13 gsasl_done()

```
_GSASL_API 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 34 of file done.c.

5.38.4.14 gsasl_encode()

```
_GSASL_API int gsasl_encode (
            Gsasl_session * sctx,
            const char * input,
            size_t input_len,
            char ** output,
            size_t * output_len )
```

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 66 of file xcode.c.

5.38.4.15 gsasl_finish()

```
_GSASL_API 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 34 of file xfinish.c.

5.38.4.16 gsasl_free()

```
_GSASL_API 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 41 of file src/free.c.

5.38.4.17 gsasl_hash_length()

```
_GSASL_API 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 73 of file crypto.c.

5.38.4.18 gsasl_hex_from()

```
_GSASL_API 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 144 of file base64.c.

5.38.4.19 gsasl_hex_to()

```
_GSASL_API 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 111 of file base64.c.

5.38.4.20 gsasl_init()

```
_GSASL_API 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 158 of file init.c.

5.38.4.21 gsasl_mechanism_name()

```
_GSASL_API 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 39 of file mechname.c.

5.38.4.22 gsasl_mechanism_name_p()

```
_GSASL_API 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 53 of file suggest.c.

5.38.4.23 gsasl_nonce()

```
_GSASL_API 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 39 of file crypto.c.

5.38.4.24 gsasl_property_fast()

```
_GSASL_API 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 262 of file property.c.

5.38.4.25 gsasl_property_free()

```
_GSASL_API 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 159 of file property.c.

5.38.4.26 gsasl_property_get()

```
_GSASL_API 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 292 of file property.c.

5.38.4.27 gsasl_property_set()

```
_GSASL_API 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 189 of file property.c.

5.38.4.28 gsasl_property_set_raw()

```
_GSASL_API int gsasl_property_set_raw (
             Gsasl_session * sctx,
             Gsasl_property prop,
             const char * data,
             size_t len )
```

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 218 of file property.c.

5.38.4.29 gsasl_random()

```
_GSASL_API 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 55 of file crypto.c.

5.38.4.30 gsasl_saslprep()

```
_GSASL_API int gsasl_saslprep (
             const char * in,
             Gsasl_saslprep_flags flags,
             char ** out,
             int * stringpreprc )
```

5.38.4.31 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 )
```

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 156 of file crypto.c.

5.38.4.32 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 )
```

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 104 of file crypto.c.

5.38.4.33 gsasl_server_mechlist()

```
_GSASL_API 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 94 of file listmech.c.

5.38.4.34 gsasl_server_start()

```
_GSASL_API 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 138 of file xstart.c.

5.38.4.35 gsasl_server_support_p()

```
_GSASL_API 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 67 of file supportp.c.

5.38.4.36 gsasl_session_hook_get()

```
_GSASL_API 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 160 of file callback.c.

5.38.4.37 gsasl_session_hook_set()

```
_GSASL_API 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 140 of file callback.c.

5.38.4.38 gsasl_simple_getpass()

```
_GSASL_API 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 48 of file md5pwd.c.

5.38.4.39 gsasl_step()

```
_GSASL_API int gsasl_step (
             Gsasl_session * sctx,
             const char * input,
             size_t input_len,
             char ** output,
             size_t * output_len )
```

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 52 of file xstep.c.

5.38.4.40 gsasl_step64()

```
_GSASL_API 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 87 of file xstep.c.

5.38.4.41 gsasl_strerror()

```
_GSASL_API 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 185 of file error.c.

5.38.4.42 gsasl_strerror_name()

```
_GSASL_API 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 223 of file error.c.

5.39 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)

5.39.1 Function Documentation

5.39.1.1 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 158 of file init.c.

5.40 internal.h File Reference

```
#include "gsasl.h"
#include <stdlib.h>
#include <string.h>
```

Data Structures

- struct Gsasl
- struct Gsasl_session

5.41 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)

5.41.1 Function Documentation

5.41.1.1 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 75 of file listmech.c.

5.41.1.2 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 94 of file listmech.c.

5.42 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

5.42.1 Macro Definition Documentation

5.42.1.1 GSASL_LOGIN_NAME

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

Definition at line 28 of file login.h.

5.42.2 Function Documentation

5.42.2.1 _gsasl_login_client_finish()

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

5.42.2.2 _gsasl_login_client_start()

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

5.42.2.3 _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 )
```

5.42.2.4 _gsasl_login_server_finish()

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

5.42.2.5 _gsasl_login_server_start()

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

5.42.2.6 _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 )
```

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

5.42.3 Variable Documentation

5.42.3.1 _gsasl_login_mechanism

```
Gsasl_mechanism _gsasl_login_mechanism  [extern]
```

Definition at line 28 of file login/mechinfo.c.

5.43 md5pwd.c File Reference

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

Functions

- int gsasl_simple_getpass (const char *filename, const char *username, char **key)

5.43.1 Function Documentation

5.43.1.1 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 48 of file md5pwd.c.

5.44 mechinfo.c File Reference

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

Variables

- Gsasl_mechanism _gsasl_anonymous_mechanism

5.44.1 Variable Documentation

5.44.1.1 _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 28 of file anonymous/mechinfo.c.

5.45 mechinfo.c File Reference

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

Variables

- Gsasl_mechanism _gsasl_cram_md5_mechanism

5.45.1 Variable Documentation

5.45.1.1 _gsasl_cram_md5_mechanism

```
Gsasl_mechanism _gsasl_cram_md5_mechanism
```

Definition at line 28 of file cram-md5/mechinfo.c.

5.46 mechinfo.c File Reference

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

Variables

- Gsasl_mechanism _gsasl_digest_md5_mechanism

5.46.1 Variable Documentation

5.46.1.1 _gsasl_digest_md5_mechanism

```
Gsasl_mechanism _gsasl_digest_md5_mechanism
```

Definition at line 28 of file digest-md5/mechinfo.c.

5.47 mechinfo.c File Reference

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

Variables

- Gsasl_mechanism _gsasl_external_mechanism

5.47.1 Variable Documentation

5.47.1.1 _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 28 of file external/mechinfo.c.

5.48 mechinfo.c File Reference

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

Variables

- Gsasl_mechanism _gsasl_gs2_krb5_mechanism

5.48.1 Variable Documentation

5.48.1.1 _gsasl_gs2_krb5_mechanism

```
Gsasl_mechanism _gsasl_gs2_krb5_mechanism
```

Definition at line 28 of file gs2/mechinfo.c.

5.49 mechinfo.c File Reference

```
#include <config.h>
#include "x-gssapi.h"
```

Variables

- Gsasl_mechanism _gsasl_gssapi_mechanism

5.49.1 Variable Documentation

5.49.1.1 _gsasl_gssapi_mechanism

```
Gsasl_mechanism _gsasl_gssapi_mechanism
```

Definition at line 28 of file gssapi/mechinfo.c.

5.50 mechinfo.c File Reference

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

Variables

- Gsasl_mechanism _gsasl_login_mechanism

5.50.1 Variable Documentation

5.50.1.1 _gsasl_login_mechanism

```
Gsasl_mechanism _gsasl_login_mechanism
```

Definition at line 28 of file login/mechinfo.c.

5.51 mechinfo.c File Reference

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

Variables

- Gsasl_mechanism _gsasl_ntlm_mechanism

5.51.1 Variable Documentation

5.51.1.1 _gsasl_ntlm_mechanism

```
Gsasl_mechanism _gsasl_ntlm_mechanism
```

Definition at line 28 of file ntlm/mechinfo.c.

5.52 mechinfo.c File Reference

```
#include <config.h>
#include "openid20.h"
```

Variables

- Gsasl_mechanism _gsasl_openid20_mechanism

5.52.1 Variable Documentation

5.52.1.1 _gsasl_openid20_mechanism

```
Gsasl_mechanism _gsasl_openid20_mechanism
```

Definition at line 28 of file openid20/mechinfo.c.

5.53 mechinfo.c File Reference

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

Variables

- Gsasl_mechanism _gsasl_plain_mechanism

5.53.1 Variable Documentation

5.53.1.1 _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 28 of file plain/mechinfo.c.

5.54 mechinfo.c File Reference

```
#include <config.h>
#include "saml20.h"
```

Variables

- Gsasl_mechanism _gsasl_saml20_mechanism

5.54.1 Variable Documentation

5.54.1.1 _gsasl_saml20_mechanism

```
Gsasl_mechanism _gsasl_saml20_mechanism
```

Definition at line 28 of file saml20/mechinfo.c.

5.55 mechinfo.c File Reference

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

5.56 mechinfo.c File Reference

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

Variables

- Gsasl_mechanism _gsasl_securid_mechanism

5.56.1 Variable Documentation

5.56.1.1 _gsasl_securid_mechanism

```
Gsasl_mechanism _gsasl_securid_mechanism
```

Definition at line 28 of file securid/mechinfo.c.

5.57 mechname.c File Reference

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

Functions

- const char * gsasl_mechanism_name (Gsasl_session *sctx)

5.57.1 Function Documentation

5.57.1.1 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 39 of file mechname.c.

5.58 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)

5.58.1 Function Documentation

5.58.1.1 _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 166 of file mechtools.c.

5.58.1.2 _gsasl_hash()

```
int _gsasl_hash (
             Gsasl_hash hash,
             const char * in,
             size_t inlen,
             char * outhash )
```

Definition at line 296 of file mechtools.c.

5.58.1.3 _gsasl_hex_decode()

```
void _gsasl_hex_decode (
             const char * hexstr,
             char * bin )
```

Definition at line 256 of file mechtools.c.

5.58.1.4 _gsasl_hex_encode()

```
void _gsasl_hex_encode (
             const char * in,
             size_t inlen,
             char * out )
```

Definition at line 221 of file mechtools.c.

5.58.1.5 _gsasl_hex_p()

```
bool _gsasl_hex_p (
             const char * hexstr )
```

Definition at line 268 of file mechtools.c.

5.58.1.6 _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 329 of file mechtools.c.

5.58.1.7 _gsasl_parse_gs2_header()

```
int _gsasl_parse_gs2_header (
             const char * data,
             size_t len,
             char ** authzid,
             size_t * headerlen )
```

Definition at line 97 of file mechtools.c.

5.58.1.8 _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 368 of file mechtools.c.

5.59 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)

5.59.1 Function Documentation

5.59.1.1 _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 166 of file mechtools.c.

5.59.1.2 _gsasl_hash()

```
int _gsasl_hash (
             Gsasl_hash hash,
             const char * in,
             size_t inlen,
             char * out )
```

Definition at line 296 of file mechtools.c.

5.59.1.3 _gsasl_hex_decode()

```
void _gsasl_hex_decode (
             const char * hexstr,
             char * bin )
```

Definition at line 256 of file mechtools.c.

5.59.1.4 _gsasl_hex_encode()

```
void _gsasl_hex_encode (
             const char * in,
             size_t inlen,
             char * out )
```

Definition at line 221 of file mechtools.c.

5.59.1.5 _gsasl_hex_p()

```
bool _gsasl_hex_p (
             const char * hexstr )
```

Definition at line 268 of file mechtools.c.

5.59.1.6 _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 329 of file mechtools.c.

5.59.1.7 _gsasl_parse_gs2_header()

```
int _gsasl_parse_gs2_header (
             const char * data,
             size_t len,
             char ** authzid,
             size_t * headerlen )
```

Definition at line 97 of file mechtools.c.

5.59.1.8 _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 368 of file mechtools.c.

5.60 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)

5.60.1 Function Documentation

5.60.1.1 latin1toutf8()

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

Definition at line 39 of file nonascii.c.

5.60.1.2 utf8tolatin1ifpossible()

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

Definition at line 67 of file nonascii.c.

5.61 nonascii.h File Reference

Functions

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

5.61.1 Function Documentation

5.61.1.1 latin1toutf8()

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

Definition at line 39 of file nonascii.c.

5.61.1.2 utf8tolatin1ifpossible()

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

Definition at line 67 of file nonascii.c.

5.62 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)

5.62.1 Typedef Documentation

5.62.1.1 _Gsasl_ntlm_state

```
typedef struct _Gsasl_ntlm_state _Gsasl_ntlm_state
```

Definition at line 1 of file ntlm.c.

5.62.2 Function Documentation

5.62.2.1 _gsasl_ntlm_client_finish()

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

Definition at line 165 of file ntlm.c.

5.62.2.2 _gsasl_ntlm_client_start()

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

Definition at line 43 of file ntlm.c.

5.62.2.3 _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 59 of file ntlm.c.

5.63 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

5.63.1 Macro Definition Documentation

5.63.1.1 GSASL_OPENID20_NAME

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

Definition at line 28 of file openid20.h.

5.63.2 Function Documentation

5.63.2.1 _gsasl_openid20_client_finish()

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

5.63.2.2 _gsasl_openid20_client_start()

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

5.63.2.3 _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 61 of file openid20/client.c.

5.63.2.4 _gsasl_openid20_server_finish()

```
void _gsasl_openid20_server_finish (
             Gsasl_session * sctx,
             void * mech_data )
```

5.63.2.5 _gsasl_openid20_server_start()

```
int _gsasl_openid20_server_start (
             Gsasl_session * sctx,
             void ** mech_data )
```

5.63.2.6 _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 )
```

Definition at line 59 of file openid20/server.c.

5.63.3 Variable Documentation

5.63.3.1 _gsasl_openid20_mechanism

```
Gsasl_mechanism _gsasl_openid20_mechanism  [extern]
```

Definition at line 28 of file openid20/mechinfo.c.

5.64 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)

5.64.1 Macro Definition Documentation

5.64.1.1 DEFAULT_ALGORITHM

```
#define DEFAULT_ALGORITHM "md5-sess"
```

Definition at line 38 of file digest-md5/parser.c.

5.64.1.2 DEFAULT_CHARSET

```
#define DEFAULT_CHARSET "utf-8"
```

Definition at line 37 of file digest-md5/parser.c.

5.64.2 Enumeration Type Documentation

5.64.2.1 anonymous enum

```
anonymous enum
```

Enumerator

CHALLENGE_REALM	
CHALLENGE_NONCE	
CHALLENGE_QOP	
CHALLENGE_STALE	
CHALLENGE_MAXBUF	
CHALLENGE_CHARSET	
CHALLENGE_ALGORITHM	
CHALLENGE_CIPHER	

Definition at line 40 of file digest-md5/parser.c.

5.64.2.2 anonymous enum

```
anonymous enum
```

Enumerator

QOP_AUTH	
QOP_AUTH_INT	
QOP_AUTH_CONF	

Definition at line 67 of file digest-md5/parser.c.

5.64.2.3 anonymous enum

```
anonymous enum
```

Enumerator

CIPHER_DES	
CIPHER_3DES	
CIPHER_RC4	
CIPHER_RC4_40	
CIPHER_RC4_56	
CIPHER_AES_CBC	

Definition at line 87 of file digest-md5/parser.c.

5.64.2.4 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 313 of file digest-md5/parser.c.

5.64.2.5 anonymous enum

```
anonymous enum
```

Enumerator

RESPONSEAUTH_RSPAUTH	

Definition at line 519 of file digest-md5/parser.c.

5.64.3 Function Documentation

5.64.3.1 digest_md5_parse_challenge()

```
int digest_md5_parse_challenge (
            const char * challenge,
            size_t len,
            digest_md5_challenge * out )
```

Definition at line 567 of file digest-md5/parser.c.

5.64.3.2 digest_md5_parse_finish()

```
int digest_md5_parse_finish (
            const char * finish,
            size_t len,
            digest_md5_finish * out )
```

Definition at line 601 of file digest-md5/parser.c.

5.64.3.3 digest_md5_parse_response()

```
int digest_md5_parse_response (
            const char * response,
            size_t len,
            digest_md5_response * out )
```

Definition at line 584 of file digest-md5/parser.c.

5.65 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)

5.65.1 Function Documentation

5.65.1.1 scram_parse_client_final()

```
int scram_parse_client_final (
            const char * str,
            size_t len,
            struct scram_client_final * cl )
```

Definition at line 329 of file scram/parser.c.

5.65.1.2 scram_parse_client_first()

```
int scram_parse_client_first (
            const char * str,
            size_t len,
            struct scram_client_first * cf )
```

Definition at line 76 of file scram/parser.c.

5.65.1.3 scram_parse_server_final()

```
int scram_parse_server_final (
            const char * str,
            size_t len,
            struct scram_server_final * sl )
```

Definition at line 459 of file scram/parser.c.

5.65.1.4 scram_parse_server_first()

```
int scram_parse_server_first (
            const char * str,
            size_t len,
            struct scram_server_first * sf )
```

Definition at line 218 of file scram/parser.c.

5.66 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)

5.66.1 Function Documentation

5.66.1.1 digest_md5_getsubopt()

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

Definition at line 44 of file getsubopt.c.

5.66.1.2 digest_md5_parse_challenge()

```
int digest_md5_parse_challenge (
            const char * challenge,
            size_t len,
            digest_md5_challenge * out )
```

Definition at line 567 of file digest-md5/parser.c.

5.66.1.3 digest_md5_parse_finish()

```
int digest_md5_parse_finish (
            const char * finish,
            size_t len,
            digest_md5_finish * out )
```

Definition at line 601 of file digest-md5/parser.c.

5.66.1.4 digest_md5_parse_response()

```
int digest_md5_parse_response (
            const char * response,
            size_t len,
            digest_md5_response * out )
```

Definition at line 584 of file digest-md5/parser.c.

5.67 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)

5.67.1 Function Documentation

5.67.1.1 scram_parse_client_final()

```
int scram_parse_client_final (
            const char * str,
            size_t len,
            struct scram_client_final * cl )
```

Definition at line 329 of file scram/parser.c.

5.67.1.2 scram_parse_client_first()

```
int scram_parse_client_first (
            const char * str,
            size_t len,
            struct scram_client_first * cf )
```

Definition at line 76 of file scram/parser.c.

5.67.1.3 scram_parse_server_final()

```
int scram_parse_server_final (
            const char * str,
            size_t len,
            struct scram_server_final * sl )
```

Definition at line 459 of file scram/parser.c.

5.67.1.4 scram_parse_server_first()

```
int scram_parse_server_first (
            const char * str,
            size_t len,
            struct scram_server_first * cf )
```

Definition at line 218 of file scram/parser.c.

5.68 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

5.68.1 Macro Definition Documentation

5.68.1.1 GSASL_PLAIN_NAME

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

Definition at line 28 of file plain.h.

5.68.2 Function Documentation

5.68.2.1 _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 )
```

5.68.2.2 _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 )
```

5.68.3 Variable Documentation

5.68.3.1 _gsasl_plain_mechanism

```
Gsasl_mechanism _gsasl_plain_mechanism  [extern]
```

Definition at line 28 of file plain/mechinfo.c.

5.69 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)

5.69.1 Function Documentation

5.69.1.1 digest_md5_print_challenge()

```
char* digest_md5_print_challenge (
            digest_md5_challenge * c )
```

Definition at line 72 of file digest-md5/printer.c.

5.69.1.2 digest_md5_print_finish()

```
char* digest_md5_print_finish (
            digest_md5_finish * finish )
```

Definition at line 385 of file digest-md5/printer.c.

5.69.1.3 digest_md5_print_response()

```
char* digest_md5_print_response (
            digest_md5_response * r )
```

Definition at line 241 of file digest-md5/printer.c.

5.70 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)

5.70.1 Function Documentation

5.70.1.1 scram_print_client_final()

```
int scram_print_client_final (
            struct scram_client_final * cl,
            char ** out )
```

Definition at line 145 of file scram/printer.c.

5.70.1.2 scram_print_client_first()

```
int scram_print_client_first (
            struct scram_client_first * cf,
            char ** out )
```

Definition at line 77 of file scram/printer.c.

5.70.1.3 scram_print_server_final()

```
int scram_print_server_final (
            struct scram_server_final * sl,
            char ** out )
```

Definition at line 165 of file scram/printer.c.

5.70.1.4 scram_print_server_first()

```
int scram_print_server_first (
            struct scram_server_first * sf,
            char ** out )
```

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

5.71 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)

5.71.1 Function Documentation

5.71.1.1 digest_md5_print_challenge()

```
char* digest_md5_print_challenge (
            digest_md5_challenge * challenge )
```

Definition at line 72 of file digest-md5/printer.c.

5.71.1.2 digest_md5_print_finish()

```
char* digest_md5_print_finish (
            digest_md5_finish * out )
```

Definition at line 385 of file digest-md5/printer.c.

5.71.1.3 digest_md5_print_response()

```
char* digest_md5_print_response (
            digest_md5_response * response )
```

Definition at line 241 of file digest-md5/printer.c.

5.72 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)

5.72.1 Function Documentation

5.72.1.1 scram_print_client_final()

```
int scram_print_client_final (
            struct scram_client_final * cl,
            char ** out )
```

Definition at line 145 of file scram/printer.c.

5.72.1.2 scram_print_client_first()

```
int scram_print_client_first (
            struct scram_client_first * cf,
            char ** out )
```

Definition at line 77 of file scram/printer.c.

5.72.1.3 scram_print_server_final()

```
int scram_print_server_final (
            struct scram_server_final * sl,
            char ** out )
```

Definition at line 165 of file scram/printer.c.

5.72.1.4 scram_print_server_first()

```
int scram_print_server_first (
            struct scram_server_first * cf,
            char ** out )
```

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

5.73 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)

5.73.1 Function Documentation

5.73.1.1 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 262 of file property.c.

5.73.1.2 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 159 of file property.c.

5.73.1.3 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 292 of file property.c.

5.73.1.4 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 189 of file property.c.

5.73.1.5 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 218 of file property.c.

5.74 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)

5.74.1 Function Documentation

5.74.1.1 digest_md5_qops2qopstr()

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

Definition at line 90 of file qop.c.

5.74.1.2 digest_md5_qopstr2qops()

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

Definition at line 35 of file qop.c.

5.75 qop.h File Reference

Functions

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

5.75.1 Function Documentation

5.75.1.1 digest_md5_qops2qopstr()

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

Definition at line 90 of file qop.c.

5.75.1.2 digest_md5_qopstr2qops()

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

Definition at line 35 of file qop.c.

5.76 register.c File Reference

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

Functions

- int gsasl_register (Gsasl *ctx, const Gsasl_mechanism *mech)

5.76.1 Function Documentation

5.76.1.1 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 39 of file register.c.

5.77 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

5.77.1 Macro Definition Documentation

5.77.1.1 GSASL_SAML20_NAME

```
#define GSASL_SAML20_NAME "SAML20"
```

Definition at line 28 of file saml20.h.

5.77.2 Function Documentation

5.77.2.1 _gsasl_saml20_client_finish()

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

5.77.2.2 _gsasl_saml20_client_start()

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

5.77.2.3 _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 )
```

Definition at line 60 of file saml20/client.c.

5.77.2.4 _gsasl_saml20_server_finish()

```
void _gsasl_saml20_server_finish (
             Gsasl_session * sctx,
             void * mech_data )
```

5.77.2.5 _gsasl_saml20_server_start()

```
int _gsasl_saml20_server_start (
             Gsasl_session * sctx,
             void ** mech_data )
```

5.77.2.6 _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 )
```

Definition at line 57 of file saml20/server.c.

5.77.3 Variable Documentation

5.77.3.1 _gsasl_saml20_mechanism

```
Gsasl_mechanism _gsasl_saml20_mechanism  [extern]
```

Definition at line 28 of file saml20/mechinfo.c.

5.78 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)

5.78.1 Function Documentation

5.78.1.1 gsasl_saslprep()

```
int gsasl_saslprep (
            const char * in,
            Gsasl_saslprep_flags flags _GL_UNUSED,
            char ** out,
            int *stringpreprc _GL_UNUSED )
```

Definition at line 87 of file saslprep.c.

5.79 scram.h File Reference

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

5.80 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

5.80.1 Macro Definition Documentation

5.80.1.1 GSASL_SECURID_NAME

```
#define GSASL_SECURID_NAME "SECURID"
```

Definition at line 28 of file securid.h.

5.80.2 Function Documentation

5.80.2.1 _gsasl_securid_client_finish()

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

5.80.2.2 _gsasl_securid_client_start()

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

5.80.2.3 _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 54 of file securid/client.c.

5.80.2.4 _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 )
```

5.80.3 Variable Documentation

5.80.3.1 _gsasl_securid_mechanism

```
Gsasl_mechanism _gsasl_securid_mechanism  [extern]
```

Definition at line 28 of file securid/mechinfo.c.

5.81 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)

5.81.1 Function Documentation

5.81.1.1 _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 29 of file anonymous/server.c.

5.82 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)

5.82.1 Macro Definition Documentation

5.82.1.1 MD5LEN

```
#define MD5LEN 16
```

Definition at line 40 of file cram-md5/server.c.

5.82.2 Function Documentation

5.82.2.1 _gsasl_cram_md5_server_finish()

```
void _gsasl_cram_md5_server_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 130 of file cram-md5/server.c.

5.82.2.2 _gsasl_cram_md5_server_start()

```
int _gsasl_cram_md5_server_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 43 of file cram-md5/server.c.

5.82.2.3 _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 66 of file cram-md5/server.c.

5.83 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)

5.83.1 Macro Definition Documentation

5.83.1.1 NONCE_ENTROPY_BYTES

```
#define NONCE_ENTROPY_BYTES 16
```

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

5.83.2 Typedef Documentation

5.83.2.1 _Gsasl_digest_md5_server_state

```
typedef struct _Gsasl_digest_md5_server_state _Gsasl_digest_md5_server_state
```

Definition at line 1 of file digest-md5/server.c.

5.83.3 Function Documentation

5.83.3.1 _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 389 of file digest-md5/server.c.

5.83.3.2 _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 365 of file digest-md5/server.c.

5.83.3.3 _gsasl_digest_md5_server_finish()

```
void _gsasl_digest_md5_server_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 349 of file digest-md5/server.c.

5.83.3.4 _gsasl_digest_md5_server_start()

```
int _gsasl_digest_md5_server_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 67 of file digest-md5/server.c.

5.83.3.5 _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 146 of file digest-md5/server.c.

5.84 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)

5.84.1 Function Documentation

5.84.1.1 _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 32 of file external/server.c.

5.85 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)

5.85.1 Typedef Documentation

5.85.1.1 _Gsasl_gs2_server_state

```
typedef struct _Gsasl_gs2_server_state _Gsasl_gs2_server_state
```

Definition at line 1 of file gs2/server.c.

5.85.2 Function Documentation

5.85.2.1 _gsasl_gs2_server_finish()

```
void _gsasl_gs2_server_finish (
             Gsasl_session * sctx,
             void * mech_data )
```

Definition at line 299 of file gs2/server.c.

5.85.2.2 _gsasl_gs2_server_start()

```
int _gsasl_gs2_server_start (
             Gsasl_session * sctx,
             void ** mech_data )
```

Definition at line 119 of file gs2/server.c.

5.85.2.3 _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 162 of file gs2/server.c.

5.86 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)

5.86.1 Typedef Documentation

5.86.1.1 _Gsasl_gssapi_server_state

```
typedef struct _Gsasl_gssapi_server_state _Gsasl_gssapi_server_state
```

Definition at line 1 of file gssapi/server.c.

5.86.2 Function Documentation

5.86.2.1 _gsasl_gssapi_server_finish()

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

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

5.86.2.2 _gsasl_gssapi_server_start()

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

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

5.86.2.3 _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 65 of file gssapi/server.c.

5.87 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)

5.87.1 Macro Definition Documentation

5.87.1.1 CHALLENGE_PASSWORD

```
#define CHALLENGE_PASSWORD "Password"
```

Definition at line 42 of file login/server.c.

5.87.1.2 CHALLENGE_USERNAME

```
#define CHALLENGE_USERNAME "User Name"
```

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

5.87.2 Function Documentation

5.87.2.1 _gsasl_login_server_finish()

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

Definition at line 148 of file login/server.c.

5.87.2.2 _gsasl_login_server_start()

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

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

5.87.2.3 _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 59 of file login/server.c.

5.88 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)

5.88.1 Function Documentation

5.88.1.1 _gsasl_openid20_server_finish()

```
void _gsasl_openid20_server_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 191 of file openid20/server.c.

5.88.1.2 _gsasl_openid20_server_start()

```
int _gsasl_openid20_server_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 44 of file openid20/server.c.

5.88.1.3 _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 59 of file openid20/server.c.

5.89 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)

5.89.1 Function Documentation

5.89.1.1 _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 35 of file plain/server.c.

5.90 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)

5.90.1 Function Documentation

5.90.1.1 _gsasl_saml20_server_finish()

```
void _gsasl_saml20_server_finish (
             Gsasl_session *sctx _GL_UNUSED,
             void * mech_data )
```

Definition at line 141 of file saml20/server.c.

5.90.1.2 _gsasl_saml20_server_start()

```
int _gsasl_saml20_server_start (
             Gsasl_session *sctx _GL_UNUSED,
             void ** mech_data )
```

Definition at line 43 of file saml20/server.c.

5.90.1.3 _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 57 of file saml20/server.c.

5.91 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)

5.91.1 Macro Definition Documentation

5.91.1.1 DEFAULT_SALT_BYTES

```
#define DEFAULT_SALT_BYTES 12
```

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

5.91.1.2 SNONCE_ENTROPY_BYTES

```
#define SNONCE_ENTROPY_BYTES 18
```

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

5.91.2 Function Documentation

5.91.2.1 _gsasl_scram_server_finish()

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

Definition at line 581 of file scram/server.c.

5.91.2.2 _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 169 of file scram/server.c.

5.92 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)

5.92.1 Macro Definition Documentation

5.92.1.1 PASSCODE

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

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

5.92.1.2 PIN

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

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

5.92.2 Function Documentation

5.92.2.1 _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 38 of file securid/server.c.

5.93 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])

5.93.1 Macro Definition Documentation

5.93.1.1 C2I

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

Value:

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

Definition at line 114 of file session.c.

5.93.1.2 MAC_DATA_LEN

```
#define MAC_DATA_LEN 4
```

Definition at line 39 of file session.c.

5.93.1.3 MAC_HMAC_LEN

```
#define MAC_HMAC_LEN 10
```

Definition at line 40 of file session.c.

5.93.1.4 MAC_MSG_TYPE

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

Definition at line 41 of file session.c.

5.93.1.5 MAC_MSG_TYPE_LEN

```
#define MAC_MSG_TYPE_LEN 2
```

Definition at line 42 of file session.c.

5.93.1.6 MAC_SEQNUM_LEN

```
#define MAC_SEQNUM_LEN 4
```

Definition at line 43 of file session.c.

5.93.1.7 MD5LEN

```
#define MD5LEN 16
```

Definition at line 37 of file session.c.

5.93.1.8 SASL_INTEGRITY_PREFIX_LENGTH

```
#define SASL_INTEGRITY_PREFIX_LENGTH 4
```

Definition at line 38 of file session.c.

5.93.2 Function Documentation

5.93.2.1 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 120 of file session.c.

5.93.2.2 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 46 of file session.c.

5.94 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])

5.94.1 Function Documentation

5.94.1.1 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] )
```

Definition at line 120 of file session.c.

5.94.1.2 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] )
```

Definition at line 46 of file session.c.

5.95 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)

Variables

- const char * _GSASL_VALID_MECHANISM_CHARACTERS

5.95.1 Function Documentation

5.95.1.1 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 88 of file suggest.c.

5.95.1.2 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 53 of file suggest.c.

5.95.2 Variable Documentation

5.95.2.1 _GSASL_VALID_MECHANISM_CHARACTERS

```
const char* _GSASL_VALID_MECHANISM_CHARACTERS
```

Initial value:

```
=
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-_"
```

Definition at line 32 of file suggest.c.

5.96 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)

5.96.1 Function Documentation

5.96.1.1 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 50 of file supportp.c.

5.96.1.2 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 67 of file supportp.c.

5.97 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)

5.97.1 Function Documentation

5.97.1.1 main()

```
int main (
            void  )
```

Definition at line 37 of file test-parser.c.

5.98 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)

5.98.1 Function Documentation

5.98.1.1 scram_free_client_final()

```
void scram_free_client_final (
            struct scram_client_final * cl )
```

Definition at line 55 of file tokens.c.

5.98.1.2 scram_free_client_first()

```
void scram_free_client_first (
            struct scram_client_first * cf )
```

Definition at line 35 of file tokens.c.

5.98.1.3 scram_free_server_final()

```
void scram_free_server_final (
            struct scram_server_final * sl )
```

Definition at line 65 of file tokens.c.

5.98.1.4 scram_free_server_first()

```
void scram_free_server_first (
            struct scram_server_first * sf )
```

Definition at line 46 of file tokens.c.

5.99 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 }

5.99.1 Macro Definition Documentation

5.99.1.1 DIGEST_MD5_LENGTH

```
#define DIGEST_MD5_LENGTH 16
```

Definition at line 30 of file digest-md5/tokens.h.

5.99.1.2 DIGEST_MD5_RESPONSE_LENGTH

```
#define DIGEST_MD5_RESPONSE_LENGTH 32
```

Definition at line 95 of file digest-md5/tokens.h.

5.99.2 Typedef Documentation

5.99.2.1 digest_md5_challenge

```
typedef struct digest_md5_challenge digest_md5_challenge
```

Definition at line 1 of file digest-md5/tokens.h.

5.99.2.2 digest_md5_cipher

```
typedef enum digest_md5_cipher digest_md5_cipher
```

Definition at line 1 of file digest-md5/tokens.h.

5.99.2.3 digest_md5_finish

```
typedef struct digest_md5_finish digest_md5_finish
```

Definition at line 1 of file digest-md5/tokens.h.

5.99.2.4 digest_md5_qop

```
typedef enum digest_md5_qop digest_md5_qop
```

Definition at line 1 of file digest-md5/tokens.h.

5.99.2.5 digest_md5_response

```
typedef struct digest_md5_response digest_md5_response
```

Definition at line 1 of file digest-md5/tokens.h.

5.99.3 Enumeration Type Documentation

5.99.3.1 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 42 of file digest-md5/tokens.h.

5.99.3.2 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 33 of file digest-md5/tokens.h.

5.100 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)

5.100.1 Function Documentation

5.100.1.1 scram_free_client_final()

```
void scram_free_client_final (
            struct scram_client_final * cl )
```

Definition at line 55 of file tokens.c.

5.100.1.2 scram_free_client_first()

```
void scram_free_client_first (
            struct scram_client_first * cf )
```

Definition at line 35 of file tokens.c.

5.100.1.3 scram_free_server_final()

```
void scram_free_server_final (
            struct scram_server_final * sl )
```

Definition at line 65 of file tokens.c.

5.100.1.4 scram_free_server_first()

```
void scram_free_server_first (
            struct scram_server_first * sf )
```

Definition at line 46 of file tokens.c.

5.101 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)

5.101.1 Function Documentation

5.101.1.1 set_saltedpassword()

```
int set_saltedpassword (
            Gsasl_session * sctx,
            Gsasl_hash hash,
            const char * hashbuf )
```

Definition at line 31 of file tools.c.

5.102 tools.h File Reference

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

Functions

- int set_saltedpassword (Gsasl_session *sctx, Gsasl_hash hash, const char *hashbuf)

5.102.1 Function Documentation

5.102.1.1 set_saltedpassword()

```
int set_saltedpassword (
            Gsasl_session * sctx,
            Gsasl_hash hash,
            const char * hashbuf )
```

Definition at line 31 of file tools.c.

5.103 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)

5.103.1 Function Documentation

5.103.1.1 digest_md5_validate()

```
int digest_md5_validate (
            digest_md5_challenge * c,
            digest_md5_response * r )
```

Definition at line 114 of file digest-md5/validate.c.

5.103.1.2 digest_md5_validate_challenge()

```
int digest_md5_validate_challenge (
            digest_md5_challenge * c )
```

Definition at line 32 of file digest-md5/validate.c.

5.103.1.3 digest_md5_validate_finish()

```
int digest_md5_validate_finish (
            digest_md5_finish * f )
```

Definition at line 101 of file digest-md5/validate.c.

5.103.1.4 digest_md5_validate_response()

```
int digest_md5_validate_response (
            digest_md5_response * r )
```

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

5.104 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)

5.104.1 Function Documentation

5.104.1.1 scram_valid_client_final()

```
bool scram_valid_client_final (
            struct scram_client_final * cl )
```

Definition at line 104 of file scram/validate.c.

5.104.1.2 scram_valid_client_first()

```
bool scram_valid_client_first (
            struct scram_client_first * cf )
```

Definition at line 32 of file scram/validate.c.

5.104.1.3 scram_valid_server_final()

```
bool scram_valid_server_final (
            struct scram_server_final * sl )
```

Definition at line 134 of file scram/validate.c.

5.104.1.4 scram_valid_server_first()

```
bool scram_valid_server_first (
            struct scram_server_first * sf )
```

Definition at line 79 of file scram/validate.c.

5.105 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)

5.105.1 Function Documentation

5.105.1.1 digest_md5_validate()

```
int digest_md5_validate (
            digest_md5_challenge * c,
            digest_md5_response * r )
```

Definition at line 114 of file digest-md5/validate.c.

5.105.1.2 digest_md5_validate_challenge()

```
int digest_md5_validate_challenge (
            digest_md5_challenge * c )
```

Definition at line 32 of file digest-md5/validate.c.

5.105.1.3 digest_md5_validate_finish()

```
int digest_md5_validate_finish (
            digest_md5_finish * f )
```

Definition at line 101 of file digest-md5/validate.c.

5.105.1.4 digest_md5_validate_response()

```
int digest_md5_validate_response (
            digest_md5_response * r )
```

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

5.106 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)

5.106.1 Function Documentation

5.106.1.1 scram_valid_client_final()

```
bool scram_valid_client_final (
            struct scram_client_final * cl )
```

Definition at line 104 of file scram/validate.c.

5.106.1.2 scram_valid_client_first()

```
bool scram_valid_client_first (
            struct scram_client_first * cf )
```

Definition at line 32 of file scram/validate.c.

5.106.1.3 scram_valid_server_final()

```
bool scram_valid_server_final (
            struct scram_server_final * sl )
```

Definition at line 134 of file scram/validate.c.

5.106.1.4 scram_valid_server_first()

```
bool scram_valid_server_first (
            struct scram_server_first * sf )
```

Definition at line 79 of file scram/validate.c.

5.107 version.c File Reference

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

Functions

- const char * gsasl_check_version (const char *req_version)

5.107.1 Function Documentation

5.107.1.1 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 46 of file version.c.

5.108 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

5.108.1 Macro Definition Documentation

5.108.1.1 GSASL_GSSAPI_NAME

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

Definition at line 28 of file x-gssapi.h.

5.108.2 Function Documentation

5.108.2.1 _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 322 of file gssapi/client.c.

5.108.2.2 _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 267 of file gssapi/client.c.

5.108.2.3 _gsasl_gssapi_client_finish()

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

Definition at line 249 of file gssapi/client.c.

5.108.2.4 _gsasl_gssapi_client_start()

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

Definition at line 47 of file gssapi/client.c.

5.108.2.5 _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 66 of file gssapi/client.c.

5.108.2.6 _gsasl_gssapi_server_finish()

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

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

5.108.2.7 _gsasl_gssapi_server_start()

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

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

5.108.2.8 _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 )
```

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

5.108.3 Variable Documentation

5.108.3.1 _gsasl_gssapi_mechanism

```
Gsasl_mechanism _gsasl_gssapi_mechanism  [extern]
```

Definition at line 28 of file gssapi/mechinfo.c.

5.109 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

5.109.1 Macro Definition Documentation

5.109.1.1 GSASL_NTLM_NAME

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

Definition at line 28 of file x-ntlm.h.

5.109.2 Function Documentation

5.109.2.1 _gsasl_ntlm_client_finish()

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

5.109.2.2 _gsasl_ntlm_client_start()

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

5.109.2.3 _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 59 of file ntlm.c.

5.109.3 Variable Documentation

5.109.3.1 _gsasl_ntlm_mechanism

```
Gsasl_mechanism _gsasl_ntlm_mechanism  [extern]
```

Definition at line 28 of file ntlm/mechinfo.c.

5.110 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)

5.110.1 Function Documentation

5.110.1.1 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 99 of file xcode.c.

5.110.1.2 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 66 of file xcode.c.

5.111 xfinish.c File Reference

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

Functions

- void gsasl_finish (Gsasl_session *sctx)

5.111.1 Function Documentation

5.111.1.1 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 34 of file xfinish.c.

5.112 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)

5.112.1 Function Documentation

5.112.1.1 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 120 of file xstart.c.

5.112.1.2 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 138 of file xstart.c.

5.113 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)

5.113.1 Function Documentation

5.113.1.1 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 52 of file xstep.c.

5.113.1.2 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 87 of file xstep.c.

Index