GNU SASL API Reference Manual

COLLABORATORS			
	TITLE : GNU SASL API Reference Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 24, 2020	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

List of Figures

Chapter 1

GNU SASL API Reference Manual

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 ANONYMOUS, CRAM-MD5, DIGEST-MD5, EXTERNAL, GS2-KRB5, GSSAPI, LOGIN, NTLM, PLAIN, SCRAM-SHA-1, SCRAM-SHA-1-PLUS, SCRAM-SHA-256, SCRAM-SHA-256-PLUS, SAML20, OPENID20, and SECURID 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 utilizes 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 supports client and server mode, and the IMAP and SMTP protocols, fits in under 80kb on an Intel x86 platform, without any modifications to the code. (This figure was accurate as of version 1.1.)

The library is licensed under the GNU Lesser General Public License version 2.1 or later. The command-line application (src/), examples (examples/), self-test suite (tests/) are licensed under the GNU General Public License license version 3.0 or later. The documentation (doc/) is licensed under the GNU Free Documentation License version 1.3 or later.

A conceptual view of how your application, the library, and each mechanism interact is shown in Figure 1.1.

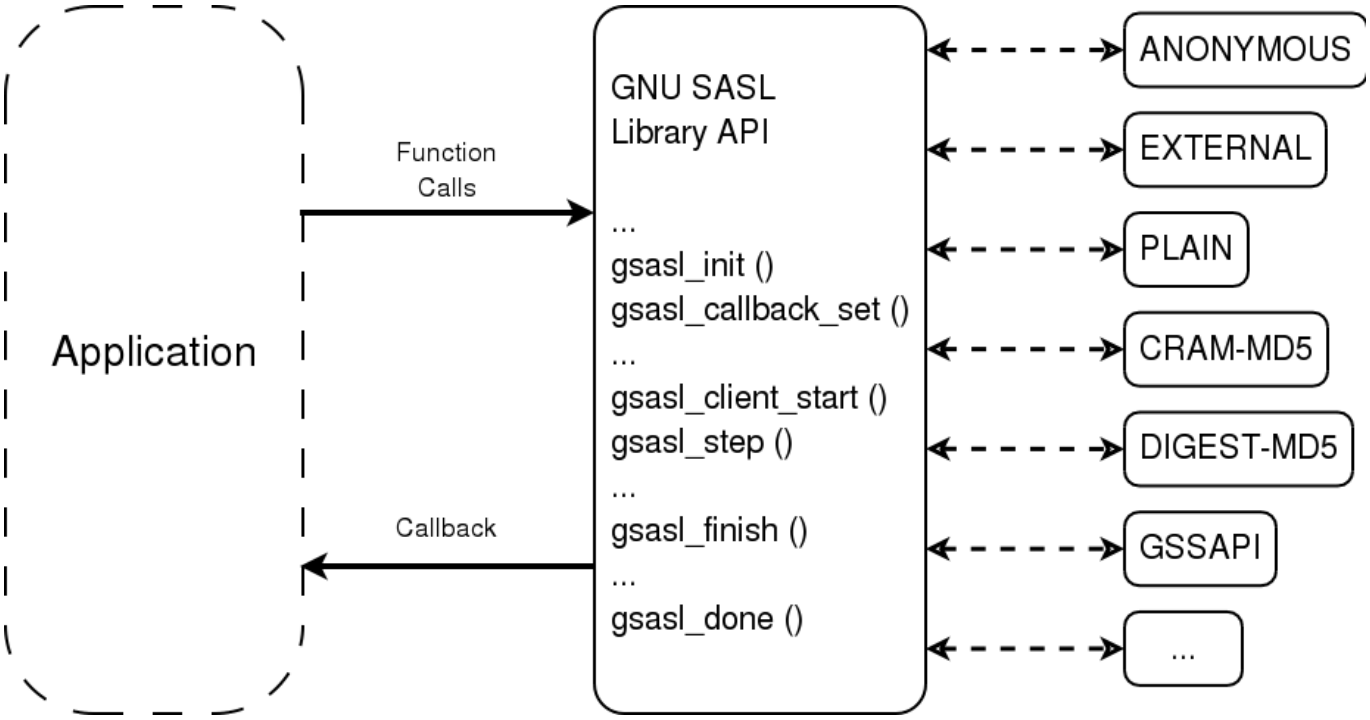

Figure 1.1: Illustration of separation between application and individual mechanism

The operation of an application using the library can best be understood in terms of a flow chart diagram, as shown in Figure 1.2. The details on how the actual negotiation are carried out are illustrated in Figure 1.3.

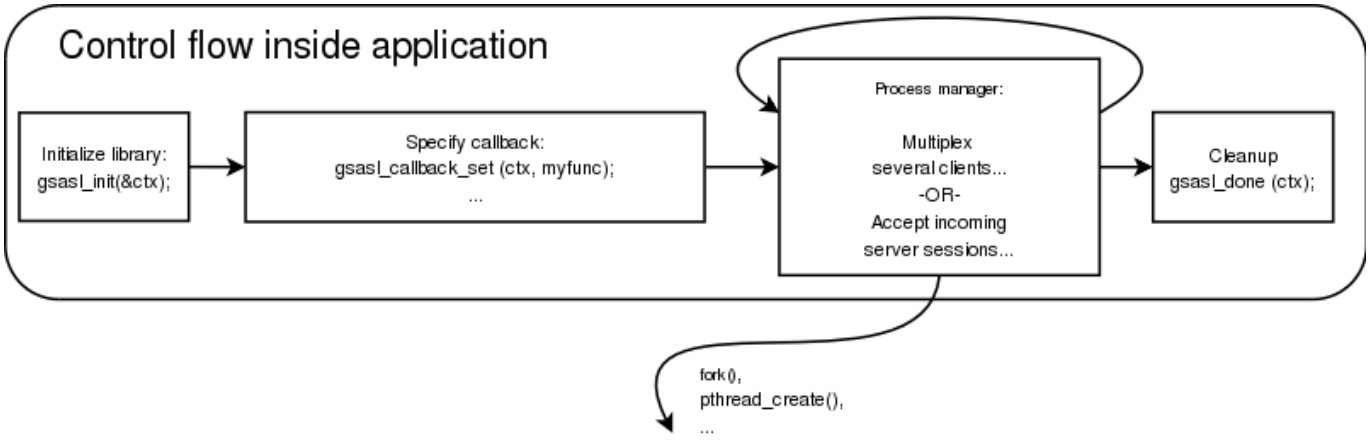

Figure 1.2: High-level control flow of SASL application

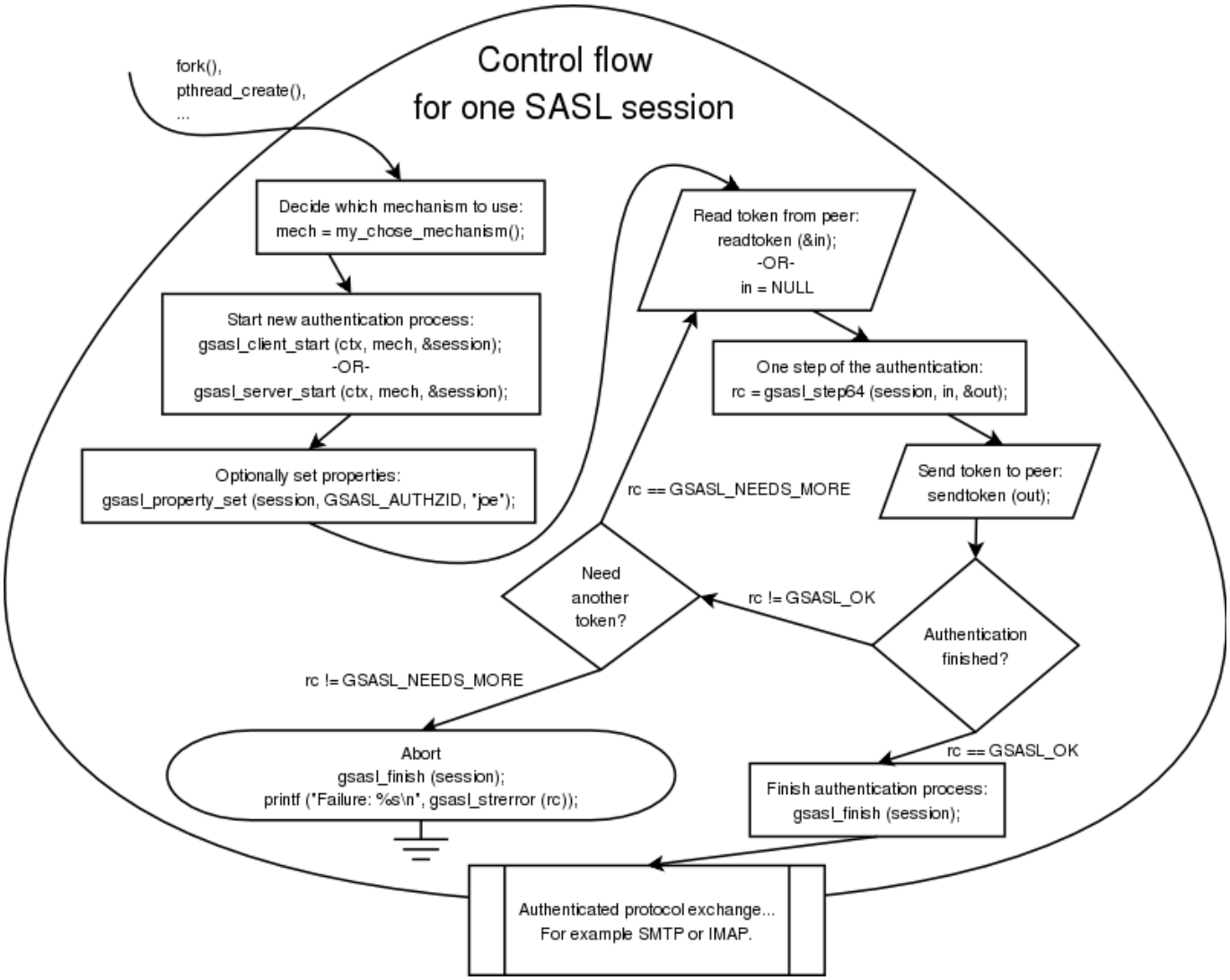

Figure 1.3: Low-level control flow of SASL application

1.1 gsasl

gsasl —

Functions

int	(*Gsasl_callback_function) ()
int	gsasl_init ()
void	gsasl_done ()
const char *	gsasl_check_version ()
void	gsasl_callback_set ()
int	gsasl_callback ()
void	gsasl_callback_hook_set ()
void *	gsasl_callback_hook_get ()
void	gsasl_session_hook_set ()
void *	gsasl_session_hook_get ()

void	gsasl_property_set ()
void	gsasl_property_set_raw ()
const char *	gsasl_property_get ()
const char *	gsasl_property_fast ()
int	gsasl_client_mechlist ()
int	gsasl_client_support_p ()
const char *	gsasl_client_suggest_mechanism ()
int	gsasl_server_mechlist ()
int	gsasl_server_support_p ()
int	gsasl_client_start ()
int	gsasl_server_start ()
int	gsasl_step ()
int	gsasl_step64 ()
void	gsasl_finish ()
int	gsasl_encode ()
int	gsasl_decode ()
const char *	gsasl_mechanism_name ()
const char *	gsasl_strerror ()
const char *	gsasl_strerror_name ()
int	gsasl_saslprep ()
int	gsasl_nonce ()
int	gsasl_random ()
size_t	gsasl_hash_length ()
int	gsasl_scram_secrets_from_salted_password ()
int	gsasl_scram_secrets_from_password ()
int	gsasl_simple_getpass ()
int	gsasl_base64_to ()
int	gsasl_base64_from ()
int	gsasl_hex_to ()
int	gsasl_hex_from ()
void	gsasl_free ()

Types and Values

#define	GSASL_API
#define	GSASL_VERSION
#define	GSASL_VERSION_MAJOR
#define	GSASL_VERSION_MINOR
#define	GSASL_VERSION_PATCH
#define	GSASL_VERSION_NUMBER
enum	Gsasl_rc
enum	Gsasl_qop
enum	Gsasl_cipher
enum	Gsasl_saslprep_flags
typedef	Gsasl
typedef	Gsasl_session
enum	Gsasl_property
enum	Gsasl_hash
enum	Gsasl_hash_length

Description

Functions

Gsasl_callback_function ()

```
int
(*Gsasl_callback_function) (Gsasl *ctx,
                            Gsasl_session *sctx,
                            Gsasl_property prop);
```

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.

Parameters

ctx	libgsasl handle.	
sctx	session handle, may be NULL.	
prop	enumerated value of Gsasl_property type.	

Returns

Any valid return code, the interpretation of which depend on the *prop* value.

Since: 0.2.0

gsasl_init ()

```
int
gsasl_init (Gsasl **ctx);
```

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().

Parameters

ctx	pointer to libgsasl handle.	

Returns

GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

gsasl_done ()

```
void
gsasl_done (Gsasl *ctx);
```

This function destroys a libgsasl handle. The handle must not be used with other libgsasl functions after this call.

Parameters

ctx	libgsasl handle.	

gsasl_check_version ()

```
const char~*
gsasl_check_version (const char *req_version);
```

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().

Parameters

req_version	version string to compare with, or NULL.	

Returns

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.

gsasl_callback_set ()

```
void
gsasl_callback_set (Gsasl *ctx,
                    Gsasl_callback_function cb);
```

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.

Parameters

ctx	handle received from gsasl_init().	
cb	pointer to function implemented by application.	

Since: 0.2.0

gsasl_callback ()

```
int
gsasl_callback (Gsasl *ctx,
                Gsasl_session *sctx,
                Gsasl_property prop);
```

Invoke the application callback. The *prop* value indicate what the callback is expected to do. For example, for GSASL_ANONYMOUS_ 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.

Note that if no callback has been set by the application, but the obsolete callback interface has been used, this function will translate the old callback interface into the new. This interface should be sufficient to invoke all callbacks, both new and old.

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.	

Returns

Returns whatever the application callback returns, or GSASL_NO_CALLBACK if no application was known.

Since: 0.2.0

gsasl_callback_hook_set ()

```
void
gsasl_callback_hook_set (Gsasl *ctx,
                         void *hook);
```

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.

Parameters

ctx	libgsasl handle.	
hook	opaque pointer to application specific data.	

Since: 0.2.0

gsasl_callback_hook_get ()

```
void~*
gsasl_callback_hook_get (Gsasl *ctx);
```

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.

Parameters

ctx	libgsasl handle.	

Returns

Returns the application specific data, or NULL.

Since: 0.2.0

gsasl_session_hook_set ()

```
void
gsasl_session_hook_set (Gsasl_session *sctx,
                        void *hook);
```

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.

Parameters

sctx	libgsasl session handle.	
hook	opaque pointer to application specific data.	

Since: 0.2.14

gsasl_session_hook_get ()

```
void~*
gsasl_session_hook_get (Gsasl_session *sctx);
```

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.

Parameters

sctx	libgsasl session handle.	

Returns

Returns the application specific data, or NULL.

Since: 0.2.14

gsasl_property_set ()

```
void
gsasl_property_set (Gsasl_session *sctx,
                    Gsasl_property prop,
                    const char *data);
```

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.

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.	

Since: 0.2.0

gsasl_property_set_raw ()

```
void
gsasl_property_set_raw (Gsasl_session *sctx,
                        Gsasl_property prop,
                        const char *data,
                        size_t len);
```

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.

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.	

Since: 0.2.0

gsasl_property_get ()

```
const char~*
gsasl_property_get (Gsasl_session *sctx,
                    Gsasl_property prop);
```

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.

If no value is known, and no callback is specified or if the callback fail to return data, and if any obsolete callback functions has been set by the application, this function will try to call these obsolete callbacks, and store the returned data as the corresponding property. This behaviour of this function will be removed when the obsolete callback interfaces are removed.

Parameters

sctx	session handle.	
prop	enumerated value of Gsasl_property type, indicating the type of data in *data* .	

Returns

Return data for property, or NULL if no value known.

Since: 0.2.0

gsasl_property_fast ()

```
const char~*
gsasl_property_fast (Gsasl_session *sctx,
                     Gsasl_property prop);
```

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.

Parameters

sctx	session handle.	
prop	enumerated value of Gsasl_property type, indicating the type of data in *data* .	

Returns

Return property value, if known, or NULL if no value known.

Since: 0.2.0

gsasl_client_mechlist ()

```
int
gsasl_client_mechlist (Gsasl *ctx,
                       char **out);
```

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.

Parameters

ctx	libgsasl handle.	
out	newly allocated output character array.	

Returns

Returns GSASL_OK if successful, or error code.

gsasl_client_support_p ()

```
int
gsasl_client_support_p (Gsasl *ctx,
                        const char *name);
```

Decide whether there is client-side support for a specified mechanism.

Parameters

ctx	libgsasl handle.	
name	name of SASL mechanism.	

Returns

Returns 1 if the libgsasl client supports the named mechanism, otherwise 0.

gsasl_client_suggest_mechanism ()

```
const char~*
gsasl_client_suggest_mechanism (Gsasl *ctx,
                                const char *mechlist);
```

Given a list of mechanisms, suggest which to use.

Parameters

ctx	libgsasl handle.	
mechlist	input character array with SASL mechanism names, separated by invalid characters (e.g. SPC).	

Returns

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.

gsasl_server_mechlist ()

```
int
gsasl_server_mechlist (Gsasl *ctx,
                       char **out);
```

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.

Parameters

ctx	libgsasl handle.	
out	newly allocated output character array.	

Returns

Returns GSASL_OK if successful, or error code.

gsasl_server_support_p ()

```
int
gsasl_server_support_p (Gsasl *ctx,
                        const char *name);
```

Decide whether there is server-side support for a specified mechanism.

Parameters

ctx	libgsasl handle.	
name	name of SASL mechanism.	

Returns

Returns 1 if the libgsasl server supports the named mechanism, otherwise 0.

gsasl_client_start ()

```
int
```

```
gsasl_client_start (Gsasl *ctx,
                    const char *mech,
                    Gsasl_session **sctx);
```

This functions initiates a client SASL authentication. This function must be called before any other gsasl_client_*() function is called.

Parameters

ctx	libgsasl handle.	
mech	name of SASL mechanism.	
sctx	pointer to client handle.	

Returns

Returns GSASL_OK if successful, or error code.

gsasl_server_start ()

```
int
gsasl_server_start (Gsasl *ctx,
                    const char *mech,
                    Gsasl_session **sctx);
```

This functions initiates a server SASL authentication. This function must be called before any other gsasl_server_*() function is called.

Parameters

ctx	libgsasl handle.	
mech	name of SASL mechanism.	
sctx	pointer to server handle.	

Returns

Returns GSASL_OK if successful, or error code.

gsasl_step ()

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

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 free (*output*).

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.	

Returns

Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

gsasl_step64 ()

```
int
gsasl_step64 (Gsasl_session *sctx,
              const char *b64input,
              char **b64output);
```

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_MOR 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 free (*b64output*).

Parameters

sctx	libgsasl client handle.	
b64input	input base64 encoded byte array.	
b64output	newly allocated output base64 encoded byte array.	

Returns

Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

gsasl_finish ()

```
void
gsasl_finish (Gsasl_session *sctx);
```

Destroy a libgsasl client or server handle. The handle must not be used with other libgsasl functions after this call.

Parameters

sctx	libgsasl session handle.	

gsasl_encode ()

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

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 free(*output*).

Parameters

sctx	libgsasl session handle.	
input	input byte array.	
input_len	size of input byte array.	
output	newly allocated output byte array.	
output_len	size of output byte array.	

Returns

Returns GSASL_OK if encoding was successful, otherwise an error code.

gsasl_decode ()

```
int
gsasl_decode (Gsasl_session *sctx,
              const char *input,
              size_t input_len,
              char **output,
              size_t *output_len);
```

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 free(*output*).

Parameters

sctx	libgsasl session handle.	
input	input byte array.	
input_len	size of input byte array.	
output	newly allocated output byte array.	
output_len	size of output byte array.	

Returns

Returns GSASL_OK if encoding was successful, otherwise an error code.

gsasl_mechanism_name ()

```
const char~*
gsasl_mechanism_name (Gsasl_session *sctx);
```

This function returns the name of the SASL mechanism used in the session. The pointer must not be deallocated by the caller.

Parameters

sctx	libgsasl session handle.	

Returns

Returns a zero terminated character array with the name of the SASL mechanism, or NULL if not known.

Since: 0.2.28

gsasl_strerror ()

```
const char~*
gsasl_strerror (int err);
```

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().

Parameters

err	libgsasl error code	

Returns

Returns a pointer to a statically allocated string containing an explanation of the error code *err* .

gsasl_strerror_name ()

```
const char~*
gsasl_strerror_name (int err);
```

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().

Parameters

err	libgsasl error code	

Returns

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

gsasl_saslprep ()

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

Prepare string using SASLprep. On success, the *out* variable must be deallocated by the caller.

Parameters

in	a UTF-8 encoded string.	
flags	any SASLprep flag, e.g., GSASL_ALLOW_UNASSIGNED.	
out	on exit, contains newly allocated output string.	
stringpreprc	if non-NULL, will hold precise stringprep return code.	

Returns

Returns GSASL_OK on success, or GSASL_SASLPREP_ERROR on error.

Since: 0.2.3

gsasl_nonce ()

```
int
gsasl_nonce (char *data,
             size_t datalen);
```

Store unpredictable data of given size in the provided buffer.

Parameters

data	output array to be filled with unpredictable random data.	
datalen	size of output array.	

Returns

Returns GSASL_OK iff successful.

gsasl_random ()

```
int
gsasl_random (char *data,
              size_t datalen);
```

Store cryptographically strong random data of given size in the provided buffer.

Parameters

data	output array to be filled with strong random data.	
datalen	size of output array.	

Returns

Returns GSASL_OK iff successful.

gsasl_hash_length ()

```
size_t
gsasl_hash_length (Gsasl_hash hash);
```

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.

Parameters

hash	a Gsasl_hash element, e.g., GSASL_HASH_SHA256.	

Returns

size of supplied Gsasl_hash element.

Since: 1.10

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);
```

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.

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.	

Returns

Returns GSASL_OK if successful, or error code.

Since: 1.10

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);
```

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.

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.	

Returns

Returns GSASL_OK if successful, or error code.

Since: 1.10

gsasl_simple_getpass ()

```
int
gsasl_simple_getpass (const char *filename,
                      const char *username,
                      char **key);
```

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.

Parameters

filename	filename of file containing passwords.	
username	username string.	
key	newly allocated output character array.	

Returns

Return GSASL_OK if output buffer contains the password, GSASL_AUTHENTICATION_ERROR if the user could not be found, or other error code.

gsasl_base64_to ()

```
int
gsasl_base64_to (const char *in,
                 size_t inlen,
                 char **out,
                 size_t *outlen);
```

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.

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.	

Returns

Returns GSASL_OK on success, or GSASL_MALLOC_ERROR if input was too large or memory allocation fail.

Since: 0.2.2

gsasl_base64_from ()

```
int
gsasl_base64_from (const char *in,
                   size_t inlen,
                   char **out,
                   size_t *outlen);
```

Decode Base64 data. The *out* buffer must be deallocated by the caller.

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	

Returns

Returns GSASL_OK on success, GSASL_BASE64_ERROR if input was invalid, and GSASL_MALLOC_ERROR on memory allocation errors.

Since: 0.2.2

gsasl_hex_to ()

```
int
gsasl_hex_to (const char *in,
              size_t inlen,
              char **out,
              size_t *outlen);
```

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.

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.	

Returns

Returns GSASL_OK on success, or GSASL_MALLOC_ERROR if input was too large or memory allocation fail.

Since: 1.10

gsasl_hex_from ()

```
int
gsasl_hex_from (const char *in,
                char **out,
                size_t *outlen);
```

Decode hex data. The *out* buffer must be deallocated by the caller.

Parameters

in	input byte array	
out	pointer to newly allocated output byte array	
outlen	pointer to size of newly allocated output byte array	

Returns

Returns GSASL_OK on success, GSASL_BASE64_ERROR if input was invalid, and GSASL_MALLOC_ERROR on memory allocation errors.

Since: 1.10

gsasl_free ()

```
void
gsasl_free (void *ptr);
```

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.

Parameters

ptr	memory pointer	

Since: 0.2.19

Types and Values

GSASL_API

```
#define GSASL_API __attribute__((__visibility__("default")))
```

GSASL_VERSION

```
#define GSASL_VERSION "1.9.2"
```

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.

GSASL_VERSION_MAJOR

```
#define GSASL_VERSION_MAJOR 1
```

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

GSASL_VERSION_MINOR

```
#define GSASL_VERSION_MINOR 9
```

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

GSASL_VERSION_PATCH

```
#define GSASL_VERSION_PATCH 2
```

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

GSASL_VERSION_NUMBER

```
#define GSASL_VERSION_NUMBER 0x010902
```

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

enum Gsasl_rc

Error codes for library functions.

Members

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_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_KERBEROS_V5_INIT_ERROR	Init error in KERBEROS_V5.
GSASL_KERBEROS_V5_INTERNAL_ERROR	General error in KERBEROS_V5.
GSASL_SHISHI_ERROR	Same as GSASL_KERBEROS_V5_INTERNAL_ERROR.
GSASL_SECURID_SERVER_NEED_ADDITIONAL_PASSCODE	SecurID mechanism needs an additional passcode.
GSASL_SECURID_SERVER_NEED_NEW_PIN	SecurID mechanism needs an new PIN.
GSASL_GSSAPI_ENCAPSULATE_TOKEN_ERROR	GSSAPI 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.

enum Gsasl_qop

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.

Members

GSASL_QOP_AUTH	Authentication only.
GSASL_QOP_AUTH_INT	Authentication and integrity.
GSASL_QOP_AUTH_CONF	Authentication, integrity and confidentiality.

enum Gsasl_cipher

Encryption types (DIGEST-MD5) for confidentiality services of application data. We recommend that you use TLS instead as it is generally more secure and have better chance of working.

Members

GSASL_CIPHER_DES	Cipher DES.
GSASL_CIPHER_3DES	Cipher 3DES.
GSASL_CIPHER_RC4	Cipher RC4.
GSASL_CIPHER_RC4_40	Cipher RC4 with 40-bit keys.
GSASL_CIPHER_RC4_56	Cipher RC4 with 56-bit keys.
GSASL_CIPHER_AES	Cipher AES.

enum Gsasl_saslprep_flags

Flags for the SASLprep function, see gsasl_saslprep(). For background, see the GNU Libidn documentation.

Members

GSASL_ALLOW_UNASSIGNED	Allow unassigned code points.

Gsasl

```
typedef struct Gsasl Gsasl;
```

Handle to global library context.

Gsasl_session

```
typedef struct Gsasl_session Gsasl_session;
```

Handle to SASL session context.

enum Gsasl_property

Callback/property types.

Members

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	GSSAPI credential principal name.
GSASL_PASSCODE	SecurID passcode.
GSASL_SUGGESTED_PIN	SecurID suggested PIN.
GSASL_PIN	SecurID PIN.
GSASL_REALM	User realm.

GSASL_DIGEST_MD5_HASHED_PASSWORD	Pre-computed hashed DIGEST-MD5 password, to avoid storing passwords in the clear.
GSASL_QOPS	Set of quality-of-protection values.
GSASL_QOP	Quality-of-protection value.
GSASL_SCRAM_ITER	Number of iterations in password-to-key hashing.
GSASL_SCRAM_SALT	Salt for password-to-key hashing.
GSASL_SCRAM_SALTED_PASSWORD	Hex-encoded hashed/salted password.

GSASL_SCRAM_SERVERKEY	Hex-encoded SCRAM ServerKey de-rived from users' pas-sowrd.
GSASL_SCRAM_STOREDKEY	Hex-encoded SCRAM Stored-Key de-rived from users' pas-sowrd.
GSASL_CB_TLS_UNIQUE	Base64 en-coded tls-unique chan-nel bind-ing.
GSASL_SAML20_IDP_IDENTIFIER	SAML20 user IdP URL.
GSASL_SAML20_REDIRECT_URL	SAML 2.0 URL to ac-cess in browser.
GSASL_OPENID20_REDIRECT_URL	OpenID 2.0 URL to ac-cess 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.

enum Gsasl_hash

Members

GSASL_HASH_SHA1	
GSASL_HASH_SHA256	

enum Gsasl_hash_length

Members

GSASL_HASH_SHA1_SIZE		
GSASL_HASH_SHA256_SIZE		
GSASL_HASH_MAX_SIZE		

1.2 gsasl-mech

gsasl-mech —

Functions

int	(*Gsasl_init_function) ()
void	(*Gsasl_done_function) ()
int	(*Gsasl_start_function) ()
int	(*Gsasl_step_function) ()
void	(*Gsasl_finish_function) ()
int	(*Gsasl_code_function) ()
int	gsasl_register ()

Types and Values

struct	Gsasl_mechanism_functions
struct	Gsasl_mechanism

Description

Functions

Gsasl_init_function ()

```
int
(*Gsasl_init_function) (Gsasl *ctx);
```

Gsasl_done_function ()

```
void
(*Gsasl_done_function) (Gsasl *ctx);
```

Gsasl_start_function ()

```
int
(*Gsasl_start_function) (Gsasl_session *sctx,
                         void **mech_data);
```

Gsasl_step_function ()

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

Gsasl_finish_function ()

```
void
(*Gsasl_finish_function) (Gsasl_session *sctx,
                          void *mech_data);
```

Gsasl_code_function ()

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

gsasl_register ()

```
int
gsasl_register (Gsasl *ctx,
                const Gsasl_mechanism *mech);
```

This function initialize given mechanism, and if successful, add it to the list of plugins that is used by the library.

Parameters

ctx	pointer to libgsasl handle.	
mech	plugin structure with information about plugin.	

Returns

GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Since: 0.2.0

Types and Values

struct Gsasl_mechanism_functions

```
struct Gsasl_mechanism_functions {
  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;
};
```

struct Gsasl_mechanism

```
struct Gsasl_mechanism {
  const char *name;

  struct Gsasl_mechanism_functions client;
  struct Gsasl_mechanism_functions server;
};
```

1.3 gsasl-compat

gsasl-compat —

Functions

int	gsasl_client_listmech ()
int	gsasl_server_listmech ()
int	gsasl_client_step ()
int	gsasl_client_step_base64 ()
int	gsasl_server_step ()
int	gsasl_server_step_base64 ()
void	gsasl_client_finish ()
void	gsasl_server_finish ()
Gsasl *	gsasl_client_ctx_get ()
Gsasl *	gsasl_server_ctx_get ()
void	gsasl_client_application_data_set ()
void *	gsasl_client_application_data_get ()
void	gsasl_server_application_data_set ()
void *	gsasl_server_application_data_get ()
int	gsasl_randomize ()
Gsasl *	gsasl_ctx_get ()
int	gsasl_encode_inline ()
int	gsasl_decode_inline ()
void	gsasl_application_data_set ()
void *	gsasl_application_data_get ()
void	gsasl_appinfo_set ()
void *	gsasl_appinfo_get ()
const char *	gsasl_server_suggest_mechanism ()
int	gsasl_base64_encode ()
int	gsasl_base64_decode ()
char *	gsasl_stringprep_nfkc ()
char *	gsasl_stringprep_saslprep ()

char *	gsasl_stringprep_trace ()
int	gsasl_md5pwd_get_password ()
int	(*Gsasl_client_callback_anonymous) ()
int	(*Gsasl_client_callback_authentication_id) ()
int	(*Gsasl_client_callback_authorization_id) ()
int	(*Gsasl_client_callback_password) ()
int	(*Gsasl_client_callback_passcode) ()
int	(*Gsasl_client_callback_pin) ()
int	(*Gsasl_client_callback_service) ()
Gsasl_qop	(*Gsasl_client_callback_qop) ()
size_t	(*Gsasl_client_callback_maxbuf) ()
int	(*Gsasl_client_callback_realm) ()
int	(*Gsasl_server_callback_retrieve) ()
int	(*Gsasl_server_callback_validate) ()
int	(*Gsasl_server_callback_gssapi) ()
int	(*Gsasl_server_callback_securid) ()
int	(*Gsasl_server_callback_cram_md5) ()
int	(*Gsasl_server_callback_digest_md5) ()
int	(*Gsasl_server_callback_service) ()
int	(*Gsasl_server_callback_external) ()
int	(*Gsasl_server_callback_anonymous) ()
int	(*Gsasl_server_callback_realm) ()
Gsasl_qop	(*Gsasl_server_callback_qop) ()
size_t	(*Gsasl_server_callback_maxbuf) ()
Gsasl_cipher	(*Gsasl_server_callback_cipher) ()
void	gsasl_client_callback_authorization_id_set ()
Gsasl_client_callback_authorization_id	gsasl_client_callback_authorization_id_get ()
void	gsasl_client_callback_authentication_id_set ()
Gsasl_client_callback_authentication_id	gsasl_client_callback_authentication_id_get ()
void	gsasl_client_callback_anonymous_set ()
Gsasl_client_callback_anonymous	gsasl_client_callback_anonymous_get ()
void	gsasl_client_callback_password_set ()
Gsasl_client_callback_password	gsasl_client_callback_password_get ()
void	gsasl_client_callback_passcode_set ()
Gsasl_client_callback_passcode	gsasl_client_callback_passcode_get ()
void	gsasl_client_callback_pin_set ()
Gsasl_client_callback_pin	gsasl_client_callback_pin_get ()
void	gsasl_client_callback_service_set ()
Gsasl_client_callback_service	gsasl_client_callback_service_get ()
void	gsasl_client_callback_qop_set ()
Gsasl_client_callback_qop	gsasl_client_callback_qop_get ()
void	gsasl_client_callback_maxbuf_set ()
Gsasl_client_callback_maxbuf	gsasl_client_callback_maxbuf_get ()
void	gsasl_client_callback_realm_set ()
Gsasl_client_callback_realm	gsasl_client_callback_realm_get ()
void	gsasl_server_callback_validate_set ()
Gsasl_server_callback_validate	gsasl_server_callback_validate_get ()
void	gsasl_server_callback_retrieve_set ()
Gsasl_server_callback_retrieve	gsasl_server_callback_retrieve_get ()
void	gsasl_server_callback_cram_md5_set ()
Gsasl_server_callback_cram_md5	gsasl_server_callback_cram_md5_get ()
void	gsasl_server_callback_digest_md5_set ()
Gsasl_server_callback_digest_md5	gsasl_server_callback_digest_md5_get ()
void	gsasl_server_callback_external_set ()
Gsasl_server_callback_external	gsasl_server_callback_external_get ()
void	gsasl_server_callback_anonymous_set ()

Gsasl_server_callback_anonymous	gsasl_server_callback_anonymous_get ()
void	gsasl_server_callback_realm_set ()
Gsasl_server_callback_realm	gsasl_server_callback_realm_get ()
void	gsasl_server_callback_qop_set ()
Gsasl_server_callback_qop	gsasl_server_callback_qop_get ()
void	gsasl_server_callback_maxbuf_set ()
Gsasl_server_callback_maxbuf	gsasl_server_callback_maxbuf_get ()
void	gsasl_server_callback_cipher_set ()
Gsasl_server_callback_cipher	gsasl_server_callback_cipher_get ()
void	gsasl_server_callback_securid_set ()
Gsasl_server_callback_securid	gsasl_server_callback_securid_get ()
void	gsasl_server_callback_gssapi_set ()
Gsasl_server_callback_gssapi	gsasl_server_callback_gssapi_get ()
void	gsasl_server_callback_service_set ()
Gsasl_server_callback_service	gsasl_server_callback_service_get ()
int	gsasl_md5 ()
int	gsasl_hmac_md5 ()
int	gsasl_sha1 ()
int	gsasl_hmac_sha1 ()

Description

Functions

gsasl_client_listmech ()

```
int
gsasl_client_listmech (Gsasl *ctx,
                       char *out,
                       size_t *outlen);
```

Warning

`gsasl_client_listmech` is deprecated and should not be used in newly-written code.
Use gsasl_client_mechlist() instead.

Write SASL names, separated by space, of mechanisms supported by the libgsasl client to the output array. To find out how large the output array must be, call this function with a NULL *out* parameter.

Parameters

ctx	libgsasl handle.	
out	output character array.	
outlen	input maximum size of output character array, on output contains actual length of output array.	

Returns

Returns GSASL_OK if successful, or error code.

gsasl_server_listmech ()

```
int
gsasl_server_listmech (Gsasl *ctx,
                       char *out,
                       size_t *outlen);
```

Warning
`gsasl_server_listmech` is deprecated and should not be used in newly-written code.
Use gsasl_server_mechlist() instead.

Write SASL names, separated by space, of mechanisms supported by the libgsasl server to the output array. To find out how large the output array must be, call this function with a NULL *out* parameter.

Parameters

ctx	libgsasl handle.	
out	output character array.	
outlen	input maximum size of output character array, on output contains actual length of output array.	

Returns

Returns GSASL_OK if successful, or error code.

gsasl_client_step ()

```
int
gsasl_client_step (Gsasl_session *sctx,
                   const char *input,
                   size_t input_len,
                   char *output,
                   size_t *output_len);
```

Warning
`gsasl_client_step` is deprecated and should not be used in newly-written code.
Use gsasl_step() instead.

Perform one step of SASL authentication in client. This reads data from server (specified with input and input_len), processes it (potentially invoking callbacks to the application), and writes data to server (into variables output and output_len).

The contents of the output buffer is unspecified if this functions returns anything other than GSASL_NEEDS_MORE.

Parameters

sctx	libgsasl client handle.	
input	input byte array.	
input_len	size of input byte array.	
output	output byte array.	
output_len	size of output byte array.	

Returns

Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

gsasl_client_step_base64 ()

```
int
gsasl_client_step_base64 (Gsasl_session *sctx,
                          const char *b64input,
                          char *b64output,
                          size_t b64output_len);
```

Warning
`gsasl_client_step_base64` is deprecated and should not be used in newly-written code.
Use gsasl_step64() instead.

This is a simple wrapper around gsasl_client_step() that base64 decodes the input and base64 encodes the output.

Parameters

sctx	libgsasl client handle.	
b64input	input base64 encoded byte array.	
b64output	output base64 encoded byte array.	
b64output_len	size of output base64 encoded byte array.	

Returns

See gsasl_client_step().

gsasl_server_step ()

```
int
gsasl_server_step (Gsasl_session *sctx,
                   const char *input,
                   size_t input_len,
                   char *output,
                   size_t *output_len);
```

Warning
`gsasl_server_step` is deprecated and should not be used in newly-written code.
Use gsasl_step() instead.

Perform one step of SASL authentication in server. This reads data from client (specified with input and input_len), processes it (potentially invoking callbacks to the application), and writes data to client (into variables output and output_len).

The contents of the output buffer is unspecified if this functions returns anything other than GSASL_NEEDS_MORE.

Parameters

sctx	libgsasl server handle.	
input	input byte array.	
input_len	size of input byte array.	
output	output byte array.	
output_len	size of output byte array.	

Returns

Returns GSASL_OK if authenticated terminated successfully, GSASL_NEEDS_MORE if more data is needed, or error code.

gsasl_server_step_base64 ()

```
int
gsasl_server_step_base64 (Gsasl_session *sctx,
                          const char *b64input,
                          char *b64output,
                          size_t b64output_len);
```

Warning
`gsasl_server_step_base64` is deprecated and should not be used in newly-written code.
Use gsasl_step64() instead.

This is a simple wrapper around gsasl_server_step() that base64 decodes the input and base64 encodes the output.

Parameters

sctx	libgsasl server handle.	
b64input	input base64 encoded byte array.	
b64output	output base64 encoded byte array.	
b64output_len	size of output base64 encoded byte array.	

Returns

See gsasl_server_step().

gsasl_client_finish ()

```
void
gsasl_client_finish (Gsasl_session *sctx);
```

Warning
`gsasl_client_finish` is deprecated and should not be used in newly-written code.
Use gsasl_finish() instead.

Destroy a libgsasl client handle. The handle must not be used with other libgsasl functions after this call.

Parameters

sctx	libgsasl client handle.	

gsasl_server_finish ()

```
void
gsasl_server_finish (Gsasl_session *sctx);
```

Warning
`gsasl_server_finish` is deprecated and should not be used in newly-written code.
Use gsasl_finish() instead.

Destroy a libgsasl server handle. The handle must not be used with other libgsasl functions after this call.

Parameters

sctx	libgsasl server handle.	

gsasl_client_ctx_get ()

```
Gsasl~*
gsasl_client_ctx_get (Gsasl_session *sctx);
```

Warning
`gsasl_client_ctx_get` is deprecated and should not be used in newly-written code.
This function is not useful with the new 0.2.0 API.

Get the libgsasl handle given a libgsasl client handle.

Parameters

sctx	libgsasl client handle

Returns

Returns the libgsasl handle given a libgsasl client handle.

gsasl_server_ctx_get ()

```
Gsasl~*
gsasl_server_ctx_get (Gsasl_session *sctx);
```

> **Warning**
> `gsasl_server_ctx_get` is deprecated and should not be used in newly-written code.
> This function is not useful with the new 0.2.0 API.

Get the libgsasl handle given a libgsasl server handle.

Parameters

sctx	libgsasl server handle

Returns

Returns the libgsasl handle given a libgsasl server handle.

gsasl_client_application_data_set ()

```
void
gsasl_client_application_data_set (Gsasl_session *sctx,
                                   void *application_data);
```

> **Warning**
> `gsasl_client_application_data_set` is deprecated and should not be used in newly-written code.
> Use gsasl_callback_hook_set() or gsasl_session_hook_set() instead.

Store application specific data in the libgsasl client handle. The application data can be later (for instance, inside a callback) be retrieved by calling gsasl_client_application_data_get(). It is normally used by the application to maintain state between the main program and the callback.

Parameters

sctx	libgsasl client handle.	
application_data	opaque pointer to application specific data.	

gsasl_client_application_data_get ()

```
void~*
gsasl_client_application_data_get (Gsasl_session *sctx);
```

Warning
`gsasl_client_application_data_get` is deprecated and should not be used in newly-written code. Use gsasl_callback_hook_get() or gsasl_session_hook_get() instead.

Retrieve application specific data from libgsasl client handle. The application data is set using gsasl_client_application_data_set(). It is normally used by the application to maintain state between the main program and the callback.

Parameters

sctx	libgsasl client handle.	

Returns

Returns the application specific data, or NULL.

gsasl_server_application_data_set ()

```
void
gsasl_server_application_data_set (Gsasl_session *sctx,
                                   void *application_data);
```

Warning
`gsasl_server_application_data_set` is deprecated and should not be used in newly-written code. Use gsasl_callback_hook_set() or gsasl_session_hook_set() instead.

Store application specific data in the libgsasl server handle. The application data can be later (for instance, inside a callback) be retrieved by calling gsasl_server_application_data_get(). It is normally used by the application to maintain state between the main program and the callback.

Parameters

sctx	libgsasl server handle.	
application_data	opaque pointer to application specific data.	

gsasl_server_application_data_get ()

```
void~*
gsasl_server_application_data_get (Gsasl_session *sctx);
```

Warning
`gsasl_server_application_data_get` is deprecated and should not be used in newly-written code.
Use gsasl_callback_hook_get() or gsasl_session_hook_get() instead.

Retrieve application specific data from libgsasl server handle. The application data is set using gsasl_server_application_data_set(). It is normally used by the application to maintain state between the main program and the callback.

Parameters

sctx	libgsasl server handle.	

Returns

Returns the application specific data, or NULL.

gsasl_randomize ()

```
int
gsasl_randomize (int strong,
                 char *data,
                 size_t datalen);
```

Warning
`gsasl_randomize` is deprecated and should not be used in newly-written code.
Use gsasl_random() or gsasl_nonce() instead.

Store cryptographically random data of given size in the provided buffer.

Parameters

strong	0 iff operation should not block, non-0 for very strong randomness.	
data	output array to be filled with random data.	
datalen	size of output array.	

Returns

Returns GSASL_OK iff successful.

gsasl_ctx_get ()

```
Gsasl~*
gsasl_ctx_get (Gsasl_session *sctx);
```

Warning
`gsasl_ctx_get` is deprecated and should not be used in newly-written code.
This function is not useful with the new 0.2.0 API.

Get the libgsasl handle given a libgsasl session handle.

Parameters

sctx	libgsasl session handle	

Returns

Returns the libgsasl handle given a libgsasl session handle.

gsasl_encode_inline ()

```
int
gsasl_encode_inline (Gsasl_session *sctx,
                     const char *input,
                     size_t input_len,
                     char *output,
                     size_t *output_len);
```

Warning
`gsasl_encode_inline` is deprecated and should not be used in newly-written code.
Use gsasl_encode() instead.

Encode data according to negotiated SASL mechanism. This might mean that data is integrity or privacy protected.

Parameters

sctx	libgsasl session handle.	
input	input byte array.	
input_len	size of input byte array.	
output	output byte array.	
output_len	size of output byte array.	

Returns

Returns GSASL_OK if encoding was successful, otherwise an error code.

Since: 0.2.0

gsasl_decode_inline ()

```
int
gsasl_decode_inline (Gsasl_session *sctx,
```

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

Warning
`gsasl_decode_inline` is deprecated and should not be used in newly-written code.
Use gsasl_decode() instead.

Decode data according to negotiated SASL mechanism. This might mean that data is integrity or privacy protected.

Parameters

sctx	libgsasl session handle.	
input	input byte array.	
input_len	size of input byte array.	
output	output byte array.	
output_len	size of output byte array.	

Returns

Returns GSASL_OK if encoding was successful, otherwise an error code.

Since: 0.2.0

gsasl_application_data_set ()

```
void
gsasl_application_data_set (Gsasl *ctx,
                            void *appdata);
```

Warning
`gsasl_application_data_set` is deprecated and should not be used in newly-written code.
Use gsasl_callback_hook_set() instead.

Store application specific data in the libgsasl handle. The application data can be later (for instance, inside a callback) be retrieved by calling gsasl_application_data_get(). It is normally used by the application to maintain state between the main program and the callback.

Parameters

ctx	libgsasl handle.	
appdata	opaque pointer to application specific data.	

gsasl_application_data_get ()

```
void~*
gsasl_application_data_get (Gsasl *ctx);
```

Warning
`gsasl_application_data_get` is deprecated and should not be used in newly-written code.
Use gsasl_callback_hook_get() instead.

Retrieve application specific data from libgsasl handle. The application data is set using gsasl_application_data_set(). It is normally used by the application to maintain state between the main program and the callback.

Parameters

ctx	libgsasl handle.	

Returns

Returns the application specific data, or NULL.

gsasl_appinfo_set ()

```
void
gsasl_appinfo_set (Gsasl_session *sctx,
                   void *appdata);
```

Warning
`gsasl_appinfo_set` is deprecated and should not be used in newly-written code.
Use gsasl_callback_hook_set() instead.

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_appinfo_get(). It is normally used by the application to maintain state between the main program and the callback.

Parameters

sctx	libgsasl session handle.	
appdata	opaque pointer to application specific data.	

gsasl_appinfo_get ()

```
void~*
gsasl_appinfo_get (Gsasl_session *sctx);
```

Warning
`gsasl_appinfo_get` is deprecated and should not be used in newly-written code.
Use gsasl_callback_hook_get() instead.

Retrieve application specific data from libgsasl session handle. The application data is set using gsasl_appinfo_set(). It is normally used by the application to maintain state between the main program and the callback.

Parameters

sctx	libgsasl session handle.	

Returns

Returns the application specific data, or NULL.

gsasl_server_suggest_mechanism ()

```
const char~*
gsasl_server_suggest_mechanism (Gsasl *ctx,
                                const char *mechlist);
```

Warning
`gsasl_server_suggest_mechanism` is deprecated and should not be used in newly-written code.
This function was never useful, since it is the client that chose which mechanism to use.

Get name of "best" SASL mechanism supported by the libgsasl server which is present in the input string.

Parameters

ctx	libgsasl handle.	
mechlist	input character array with SASL mechanism names, separated by invalid characters (e.g. SPC).	

Returns

Returns name of "best" SASL mechanism supported by the libgsasl server which is present in the input string.

gsasl_base64_encode ()

```
int
gsasl_base64_encode (char const *src,
                     size_t srclength,
                     char *target,
                     size_t targsize);
```

Warning
`gsasl_base64_encode` is deprecated and should not be used in newly-written code.
Use gsasl_base64_to() instead.

Encode data as base64. Converts characters, three at a time, starting at src into four base64 characters in the target area until the entire input buffer is encoded.

Parameters

src	input byte array	
srclength	size of input byte array	
target	output byte array	
targsize	size of output byte array	

Returns

Returns the number of data bytes stored at the target, or -1 on error.

gsasl_base64_decode ()

```
int
gsasl_base64_decode (char const *src,
                     char *target,
                     size_t targsize);
```

Warning
`gsasl_base64_decode` is deprecated and should not be used in newly-written code.
Use gsasl_base64_from() instead.

Decode Base64 data. Skips all whitespace anywhere. Converts characters, four at a time, starting at (or after) src from Base64 numbers into three 8 bit bytes in the target area.

Parameters

src	input byte array	
target	output byte array	
targsize	size of output byte array	

Returns

Returns the number of data bytes stored at the target, or -1 on error.

gsasl_stringprep_nfkc ()

```
char~*
gsasl_stringprep_nfkc (const char *in,
```

```
                        ssize_t len);
```

Warning
`gsasl_stringprep_nfkc` is deprecated and should not be used in newly-written code.
No replacement functionality in GNU SASL, use GNU Libidn instead. Note that in SASL, you most likely want to use SASLprep and not bare NFKC, see gsasl_saslprep().

Converts a string into canonical form, standardizing such issues as whether a character with an accent is represented as a base character and combining accent or as a single precomposed character.

The normalization mode is NFKC (ALL COMPOSE). It standardizes differences that do not affect the text content, such as the above-mentioned accent representation. It standardizes the "compatibility" characters in Unicode, such as SUPERSCRIPT THREE to the standard forms (in this case DIGIT THREE). Formatting information may be lost but for most text operations such characters should be considered the same. It returns a result with composed forms rather than a maximally decomposed form.

Parameters

in	a UTF-8 encoded string.	
len	length of *str* , in bytes, or -1 if *str* is nul-terminated.	

Returns

Return a newly allocated string, that is the NFKC normalized form of *str* , or NULL on error.

gsasl_stringprep_saslprep ()

```
char~*
gsasl_stringprep_saslprep (const char *in,
                           int *stringprep_rc);
```

Warning
`gsasl_stringprep_saslprep` is deprecated and should not be used in newly-written code.
Use gsasl_saslprep() instead.

Process a Unicode string for comparison, according to the "SASLprep" stringprep profile. This function is intended to be used by Simple Authentication and Security Layer (SASL) mechanisms (such as PLAIN, CRAM-MD5, and DIGEST-MD5) as well as other protocols exchanging user names and/or passwords.

Parameters

in	input ASCII or UTF-8 string with data to prepare according to SASLprep.	
stringprep_rc	pointer to output variable with stringprep error code, or NULL to indicate that you don't care about it.	

Returns

Return a newly allocated string that is the "SASLprep" processed form of the input string, or NULL on error, in which case *stringprep_rc* contain the stringprep library error code.

gsasl_stringprep_trace ()

```
char~*
gsasl_stringprep_trace (const char *in,
                        int *stringprep_rc);
```

Warning
`gsasl_stringprep_trace` is deprecated and should not be used in newly-written code.
No replacement functionality in GNU SASL, use GNU Libidn instead.

Process a Unicode string for use as trace information, according to the "trace" stringprep profile. The profile is designed for use with the SASL ANONYMOUS Mechanism.

Parameters

in	input ASCII or UTF-8 string with data to prepare according to "trace".	
stringprep_rc	pointer to output variable with stringprep error code, or NULL to indicate that you don't care about it.	

Returns

Return a newly allocated string that is the "trace" processed form of the input string, or NULL on error, in which case *stringprep_rc* contain the stringprep library error code.

gsasl_md5pwd_get_password ()

```
int
gsasl_md5pwd_get_password (const char *filename,
                           const char *username,
                           char *key,
                           size_t *keylen);
```

Warning
`gsasl_md5pwd_get_password` is deprecated and should not be used in newly-written code.
Use gsasl_simple_getpass() instead.

Retrieve password for user from specified file. To find out how large the output array must be, call this function with out=NULL.

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.

Parameters

filename	filename of file containing passwords.	
username	username string.	
key	output character array.	
keylen	input maximum size of output character array, on output contains actual length of output array.	

Returns

Return GSASL_OK if output buffer contains the password, GSASL_AUTHENTICATION_ERROR if the user could not be found, or other error code.

Gsasl_client_callback_anonymous ()

```
int
(*Gsasl_client_callback_anonymous) (Gsasl_session *sctx,
                                    char *out,
                                    size_t *outlen);
```

Gsasl_client_callback_authentication_id ()

```
int
(*Gsasl_client_callback_authentication_id)
                                (Gsasl_session *sctx,
                                 char *out,
                                 size_t *outlen);
```

Gsasl_client_callback_authorization_id ()

```
int
(*Gsasl_client_callback_authorization_id)
                                (Gsasl_session *sctx,
                                 char *out,
                                 size_t *outlen);
```

Gsasl_client_callback_password ()

```
int
(*Gsasl_client_callback_password) (Gsasl_session *sctx,
                                   char *out,
                                   size_t *outlen);
```

Gsasl_client_callback_passcode ()

```
int
(*Gsasl_client_callback_passcode) (Gsasl_session *sctx,
                                   char *out,
                                   size_t *outlen);
```

Gsasl_client_callback_pin ()

```
int
(*Gsasl_client_callback_pin) (Gsasl_session *sctx,
                              char *suggestion,
                              char *out,
                              size_t *outlen);
```

Gsasl_client_callback_service ()

```
int
(*Gsasl_client_callback_service) (Gsasl_session *sctx,
                                  char *service,
                                  size_t *servicelen,
                                  char *hostname,
                                  size_t *hostnamelen,
                                  char *servicename,
                                  size_t *servicenamelen);
```

Gsasl_client_callback_qop ()

```
Gsasl_qop
(*Gsasl_client_callback_qop) (Gsasl_session *sctx,
                              Gsasl_qop serverqops);
```

Gsasl_client_callback_maxbuf ()

```
size_t
(*Gsasl_client_callback_maxbuf) (Gsasl_session *sctx,
                                 size_t servermaxbuf);
```

Gsasl_client_callback_realm ()

```
int
(*Gsasl_client_callback_realm) (Gsasl_session *sctx,
                                char *out,
                                size_t *outlen);
```

Gsasl_server_callback_retrieve ()

```
int
(*Gsasl_server_callback_retrieve) (Gsasl_session *sctx,
                                   const char *authentication_id,
                                   const char *authorization_id,
                                   const char *realm,
                                   char *key,
                                   size_t *keylen);
```

Gsasl_server_callback_validate ()

```
int
(*Gsasl_server_callback_validate) (Gsasl_session *sctx,
                                   const char *authorization_id,
                                   const char *authentication_id,
                                   const char *password);
```

Gsasl_server_callback_gssapi ()

```
int
(*Gsasl_server_callback_gssapi) (Gsasl_session *sctx,
                                 const char *clientname,
                                 const char *authentication_id);
```

Gsasl_server_callback_securid ()

```
int
(*Gsasl_server_callback_securid) (Gsasl_session *sctx,
                                  const char *authentication_id,
                                  const char *authorization_id,
                                  const char *passcode,
                                  char *pin,
                                  char *suggestpin,
                                  size_t *suggestpinlen);
```

Gsasl_server_callback_cram_md5 ()

```
int
(*Gsasl_server_callback_cram_md5) (Gsasl_session *sctx,
                                   char *username,
                                   char *challenge,
                                   char *response);
```

Gsasl_server_callback_digest_md5 ()

```
int
(*Gsasl_server_callback_digest_md5) (Gsasl_session *sctx,
                                     char *username,
                                     char *realm,
                                     char *secrethash);
```

Gsasl_server_callback_service ()

```
int
(*Gsasl_server_callback_service) (Gsasl_session *sctx,
                                  char *service,
                                  size_t *servicelen,
                                  char *hostname,
                                  size_t *hostnamelen);
```

Gsasl_server_callback_external ()

```
int
(*Gsasl_server_callback_external) (Gsasl_session *sctx);
```

Gsasl_server_callback_anonymous ()

```
int
(*Gsasl_server_callback_anonymous) (Gsasl_session *sctx,
                                    const char *token);
```

Gsasl_server_callback_realm ()

```
int
(*Gsasl_server_callback_realm) (Gsasl_session *sctx,
                                char *out,
                                size_t *outlen,
                                size_t nth);
```

Gsasl_server_callback_qop ()

```
Gsasl_qop
(*Gsasl_server_callback_qop) (Gsasl_session *sctx);
```

Gsasl_server_callback_maxbuf ()

```
size_t
(*Gsasl_server_callback_maxbuf) (Gsasl_session *sctx);
```

Gsasl_server_callback_cipher ()

```
Gsasl_cipher
(*Gsasl_server_callback_cipher) (Gsasl_session *sctx);
```

gsasl_client_callback_authorization_id_set ()

```
void
gsasl_client_callback_authorization_id_set
                                (Gsasl *ctx,
                                 Gsasl_client_callback_authorization_id cb);
```

Warning

`gsasl_client_callback_authorization_id_set` is deprecated and should not be used in newly-written code.

This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the client to set the authorization identity. The function can be later retrieved using gsasl_client_callback_authorization_id_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_client_callback_authorization_id_get ()

```
Gsasl_client_callback_authorization_id
gsasl_client_callback_authorization_id_get
                                (Gsasl *ctx);
```

Warning

`gsasl_client_callback_authorization_id_get` is deprecated and should not be used in newly-written code.

This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_client_callback_authorization_id_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_client_callback_authorization_id_set().

gsasl_client_callback_authentication_id_set ()

```
void
gsasl_client_callback_authentication_id_set
```

```
                                (Gsasl *ctx,
                                 Gsasl_client_callback_authentication_id cb);
```

Warning

`gsasl_client_callback_authentication_id_set` is deprecated and should not be used in newly-written code.

This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the client to set the authentication identity. The function can be later retrieved using gsasl_client_callback_authentication_id_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_client_callback_authentication_id_get ()

```
Gsasl_client_callback_authentication_id
gsasl_client_callback_authentication_id_get
                                (Gsasl *ctx);
```

Warning

`gsasl_client_callback_authentication_id_get` is deprecated and should not be used in newly-written code.

This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_client_callback_authentication_id_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_client_callback_authentication_id_set().

gsasl_client_callback_anonymous_set ()

```
void
gsasl_client_callback_anonymous_set (Gsasl *ctx,
                                     Gsasl_client_callback_anonymous cb);
```

Warning
`gsasl_client_callback_anonymous_set` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the client to set the anonymous token, which usually is the users email address. The function can be later retrieved using gsasl_client_callback_anonymous_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_client_callback_anonymous_get ()

```
Gsasl_client_callback_anonymous
gsasl_client_callback_anonymous_get (Gsasl *ctx);
```

Warning
`gsasl_client_callback_anonymous_get` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_client_callback_anonymous_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_client_callback_anonymous_set().

gsasl_client_callback_password_set ()

```
void
gsasl_client_callback_password_set (Gsasl *ctx,
                                    Gsasl_client_callback_password cb);
```

Warning
`gsasl_client_callback_password_set` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the client to set the password. The function can be later retrieved using gsasl_client_callback_passv

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_client_callback_password_get ()

```
Gsasl_client_callback_password
gsasl_client_callback_password_get (Gsasl *ctx);
```

Warning

`gsasl_client_callback_password_get` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_client_callback_password_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_client_callback_password_set().

gsasl_client_callback_passcode_set ()

```
void
gsasl_client_callback_passcode_set (Gsasl *ctx,
                                    Gsasl_client_callback_passcode cb);
```

Warning

`gsasl_client_callback_passcode_set` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the client to set the passcode. The function can be later retrieved using gsasl_client_callback_passc

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_client_callback_passcode_get ()

```
Gsasl_client_callback_passcode
gsasl_client_callback_passcode_get (Gsasl *ctx);
```

Warning
`gsasl_client_callback_passcode_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_client_callback_passcode_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_client_callback_passcode_set().

gsasl_client_callback_pin_set ()

```
void
gsasl_client_callback_pin_set (Gsasl *ctx,
                               Gsasl_client_callback_pin cb);
```

Warning
`gsasl_client_callback_pin_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the client to chose a new pin, possibly suggested by the server, for the SECURID mechanism. This is not normally invoked, but only when the server requests it. The function can be later retrieved using gsasl_client_callback_pin_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_client_callback_pin_get ()

```
Gsasl_client_callback_pin
gsasl_client_callback_pin_get (Gsasl *ctx);
```

Warning
`gsasl_client_callback_pin_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_client_callback_pin_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_client_callback_pin_set().

gsasl_client_callback_service_set ()

```
void
gsasl_client_callback_service_set (Gsasl *ctx,
                                   Gsasl_client_callback_service cb);
```

Warning
`gsasl_client_callback_service_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the client to set the name of the service. The service buffer should be a registered GSSAPI host-based service name, hostname the name of the server. Servicename is used by DIGEST-MD5 and should be the name of generic server in case of a replicated service. The function can be later retrieved using gsasl_client_callback_service_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_client_callback_service_get ()

```
Gsasl_client_callback_service
gsasl_client_callback_service_get (Gsasl *ctx);
```

Warning
`gsasl_client_callback_service_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_client_callback_service_set().

Parameters

ctx	libgsasl handle.

Returns

Returns the callback earlier set by calling gsasl_client_callback_service_set().

gsasl_client_callback_qop_set ()

```
void
gsasl_client_callback_qop_set (Gsasl *ctx,
                               Gsasl_client_callback_qop cb);
```

Warning

`gsasl_client_callback_qop_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the client to determine the qop to use after looking at what the server offered. The function can be later retrieved using gsasl_client_callback_qop_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_client_callback_qop_get ()

```
Gsasl_client_callback_qop
gsasl_client_callback_qop_get (Gsasl *ctx);
```

Warning

`gsasl_client_callback_qop_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_client_callback_qop_set().

Parameters

ctx	libgsasl handle.

Returns

Returns the callback earlier set by calling gsasl_client_callback_qop_set().

gsasl_client_callback_maxbuf_set ()

```
void
gsasl_client_callback_maxbuf_set (Gsasl *ctx,
                                  Gsasl_client_callback_maxbuf cb);
```

Warning

`gsasl_client_callback_maxbuf_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the client to inform the server of the largest buffer the client is able to receive when using the DIGEST-MD5 "auth-int" or "auth-conf" Quality of Protection (qop). If this directive is missing, the default value 65536 will be assumed. The function can be later retrieved using gsasl_client_callback_maxbuf_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_client_callback_maxbuf_get ()

```
Gsasl_client_callback_maxbuf
gsasl_client_callback_maxbuf_get (Gsasl *ctx);
```

Warning

`gsasl_client_callback_maxbuf_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_client_callback_maxbuf_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_client_callback_maxbuf_set().

gsasl_client_callback_realm_set ()

```
void
gsasl_client_callback_realm_set (Gsasl *ctx,
                                 Gsasl_client_callback_realm cb);
```

Warning
`gsasl_client_callback_realm_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the client to know which realm it belongs to. The realm is used by the server to determine which username and password to use. The function can be later retrieved using gsasl_client_callback_realm_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_client_callback_realm_get ()

```
Gsasl_client_callback_realm
gsasl_client_callback_realm_get (Gsasl *ctx);
```

Warning
`gsasl_client_callback_realm_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_client_callback_realm_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_client_callback_realm_set().

gsasl_server_callback_validate_set ()

```
void
gsasl_server_callback_validate_set (Gsasl *ctx,
                                    Gsasl_server_callback_validate cb);
```

Warning
`gsasl_server_callback_validate_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server for deciding if user is authenticated using authentication identity, authorization identity and password. The function can be later retrieved using gsasl_server_callback_validate_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_validate_get ()

```
Gsasl_server_callback_validate
gsasl_server_callback_validate_get (Gsasl *ctx);
```

Warning

`gsasl_server_callback_validate_get` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_validate_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_server_callback_validate_set().

gsasl_server_callback_retrieve_set ()

```
void
gsasl_server_callback_retrieve_set (Gsasl *ctx,
                                    Gsasl_server_callback_retrieve cb);
```

Warning

`gsasl_server_callback_retrieve_set` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server for deciding if user is authenticated using authentication identity, authorization identity and password. The function can be later retrieved using gsasl_server_callback_retrieve_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_retrieve_get ()

```
Gsasl_server_callback_retrieve
gsasl_server_callback_retrieve_get (Gsasl *ctx);
```

Warning

`gsasl_server_callback_retrieve_get` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_retrieve_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_server_callback_retrieve_set().

gsasl_server_callback_cram_md5_set ()

```
void
gsasl_server_callback_cram_md5_set (Gsasl *ctx,
                                    Gsasl_server_callback_cram_md5 cb);
```

Warning

`gsasl_server_callback_cram_md5_set` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server for deciding if user is authenticated using CRAM-MD5 challenge and response. The function can be later retrieved using gsasl_server_callback_cram_md5_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_cram_md5_get ()

```
Gsasl_server_callback_cram_md5
gsasl_server_callback_cram_md5_get (Gsasl *ctx);
```

Warning
`gsasl_server_callback_cram_md5_get` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_cram_md5_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_server_callback_cram_md5_set().

gsasl_server_callback_digest_md5_set ()

```
void
gsasl_server_callback_digest_md5_set (Gsasl *ctx,
                                      Gsasl_server_callback_digest_md5 cb);
```

Warning
`gsasl_server_callback_digest_md5_set` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server for retrieving the secret hash of the username, realm and password for use in the DIGEST-MD5 mechanism. The function can be later retrieved using gsasl_server_callback_digest_md5_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_digest_md5_get ()

```
Gsasl_server_callback_digest_md5
gsasl_server_callback_digest_md5_get (Gsasl *ctx);
```

Warning
`gsasl_server_callback_digest_md5_get` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_digest_md5_set().

Parameters

ctx	libgsasl handle.

Returns

Return the callback earlier set by calling gsasl_server_callback_digest_md5_set().

gsasl_server_callback_external_set ()

```
void
gsasl_server_callback_external_set (Gsasl *ctx,
                                    Gsasl_server_callback_external cb);
```

Warning

`gsasl_server_callback_external_set` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server for deciding if user is authenticated out of band. The function can be later retrieved using gsasl_server_callback_external_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_external_get ()

```
Gsasl_server_callback_external
gsasl_server_callback_external_get (Gsasl *ctx);
```

Warning

`gsasl_server_callback_external_get` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_external_set().

Parameters

ctx	libgsasl handle.

Returns

Returns the callback earlier set by calling gsasl_server_callback_external_set().

gsasl_server_callback_anonymous_set ()

```
void
gsasl_server_callback_anonymous_set (Gsasl *ctx,
                                     Gsasl_server_callback_anonymous cb);
```

Warning

`gsasl_server_callback_anonymous_set` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server for deciding if user is permitted anonymous access. The function can be later retrieved using gsasl_server_callback_anonymous_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_anonymous_get ()

```
Gsasl_server_callback_anonymous
gsasl_server_callback_anonymous_get (Gsasl *ctx);
```

Warning

`gsasl_server_callback_anonymous_get` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_anonymous_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_server_callback_anonymous_set().

gsasl_server_callback_realm_set ()

```
void
gsasl_server_callback_realm_set (Gsasl *ctx,
                                 Gsasl_server_callback_realm cb);
```

Warning

`gsasl_server_callback_realm_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server to know which realm it serves. The realm is used by the user to determine which username and password to use. The function can be later retrieved using gsasl_server_callback_realm_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_realm_get ()

```
Gsasl_server_callback_realm
gsasl_server_callback_realm_get (Gsasl *ctx);
```

Warning

`gsasl_server_callback_realm_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_realm_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_server_callback_realm_set().

gsasl_server_callback_qop_set ()

```
void
gsasl_server_callback_qop_set (Gsasl *ctx,
                               Gsasl_server_callback_qop cb);
```

Warning

`gsasl_server_callback_qop_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server to know which quality of protection it accepts. The quality of protection eventually used is selected by the client though. It is currently used by the DIGEST-MD5 mechanism. The function can be later retrieved using gsasl_server_callback_qop_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_qop_get ()

```
Gsasl_server_callback_qop
gsasl_server_callback_qop_get (Gsasl *ctx);
```

Warning
`gsasl_server_callback_qop_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_qop_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_server_callback_qop_set().

gsasl_server_callback_maxbuf_set ()

```
void
gsasl_server_callback_maxbuf_set (Gsasl *ctx,
                                  Gsasl_server_callback_maxbuf cb);
```

Warning
`gsasl_server_callback_maxbuf_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server to inform the client of the largest buffer the server is able to receive when using the DIGEST-MD5 "auth-int" or "auth-conf" Quality of Protection (qop). If this directive is missing, the default value 65536 will be assumed. The function can be later retrieved using gsasl_server_callback_maxbuf_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_maxbuf_get ()

```
Gsasl_server_callback_maxbuf
gsasl_server_callback_maxbuf_get (Gsasl *ctx);
```

Warning
`gsasl_server_callback_maxbuf_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_maxbuf_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_server_callback_maxbuf_set().

gsasl_server_callback_cipher_set ()

```
void
gsasl_server_callback_cipher_set (Gsasl *ctx,
                                  Gsasl_server_callback_cipher cb);
```

Warning
`gsasl_server_callback_cipher_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server to inform the client of the cipher suites supported. The DES and 3DES ciphers must be supported for interoperability. It is currently used by the DIGEST-MD5 mechanism. The function can be later retrieved using gsasl_server_callback_cipher_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_cipher_get ()

```
Gsasl_server_callback_cipher
gsasl_server_callback_cipher_get (Gsasl *ctx);
```

Warning
`gsasl_server_callback_cipher_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_cipher_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_server_callback_cipher_set().

gsasl_server_callback_securid_set ()

```
void
gsasl_server_callback_securid_set (Gsasl *ctx,
                                   Gsasl_server_callback_securid cb);
```

Warning
`gsasl_server_callback_securid_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server for validating a user via the SECURID mechanism. The function should return GSASL_OK if user authenticated successfully, GSASL_SECURID_SERVER_NEED_ADDITIONAL_PASSCODE if it wants another passcode, GSASL_SECURID_SERVER_NEED_NEW_PIN if it wants a PIN change, or an error. When (and only when) GSASL_SECURID_SERVER_NEED_NEW_PIN is returned, suggestpin can be populated with a PIN code the server suggests, and suggestpinlen set to the length of the PIN. The function can be later retrieved using gsasl_server_callback_securid_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_securid_get ()

```
Gsasl_server_callback_securid
gsasl_server_callback_securid_get (Gsasl *ctx);
```

Warning
`gsasl_server_callback_securid_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_securid_set().

Parameters

ctx	libgsasl handle.	

Returns

Returns the callback earlier set by calling gsasl_server_callback_securid_set().

gsasl_server_callback_gssapi_set ()

```
void
gsasl_server_callback_gssapi_set (Gsasl *ctx,
                                  Gsasl_server_callback_gssapi cb);
```

Warning
`gsasl_server_callback_gssapi_set` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server for checking if a GSSAPI user is authorized for username (by, e.g., calling krb5_kuserok). The function should return GSASL_OK if the user should be permitted access, or an error code such as GSASL_AUTHENTICATION_ERROR on failure. The function can be later retrieved using gsasl_server_callback_gssapi_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_gssapi_get ()

```
Gsasl_server_callback_gssapi
gsasl_server_callback_gssapi_get (Gsasl *ctx);
```

Warning
`gsasl_server_callback_gssapi_get` is deprecated and should not be used in newly-written code.
This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_gssapi_set().

Parameters

ctx	libgsasl handle.

Returns

Returns the callback earlier set by calling gsasl_server_callback_gssapi_set().

gsasl_server_callback_service_set ()

```
void
gsasl_server_callback_service_set (Gsasl *ctx,
                                   Gsasl_server_callback_service cb);
```

Warning

`gsasl_server_callback_service_set` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Specify the callback function to use in the server to set the name of the service. The service buffer should be a registered GSSAPI host-based service name, hostname the name of the server. The function can be later retrieved using gsasl_server_callback_service_get().

Parameters

ctx	libgsasl handle.	
cb	callback function	

gsasl_server_callback_service_get ()

```
Gsasl_server_callback_service
gsasl_server_callback_service_get (Gsasl *ctx);
```

Warning

`gsasl_server_callback_service_get` is deprecated and should not be used in newly-written code. This function is part of the old callback interface. The new interface uses gsasl_callback_set() to set the application callback, and uses gsasl_callback() or gsasl_property_get() to invoke the callback for certain properties.

Get the callback earlier set by calling gsasl_server_callback_service_set().

Parameters

ctx	libgsasl handle.

Returns

Returns the callback earlier set by calling gsasl_server_callback_service_set().

gsasl_md5 ()

```
int
gsasl_md5 (const char *in,
           size_t inlen,
           char *out[]);
```

Warning
`gsasl_md5` is deprecated and should not be used in newly-written code.
Use a crypto library.

Compute hash of data using MD5. The *out* buffer must be deallocated by the caller.

Parameters

in	input character array of data to hash.	
inlen	length of input character array of data to hash.	
out	newly allocated 16-byte character array with hash of data.	

Returns

Returns GSASL_OK iff successful.

gsasl_hmac_md5 ()

```
int
gsasl_hmac_md5 (const char *key,
                size_t keylen,
                const char *in,
                size_t inlen,
                char *outhash[]);
```

Warning
`gsasl_hmac_md5` is deprecated and should not be used in newly-written code.
Use a crypto library.

Compute keyed checksum of data using HMAC-MD5. The *outhash* buffer must be deallocated by the caller.

Parameters

key	input character array with key to use.	

keylen	length of input character array with key to use.	
in	input character array of data to hash.	
inlen	length of input character array of data to hash.	
outhash	newly allocated 16-byte character array with keyed hash of data.	

Returns

Returns GSASL_OK iff successful.

gsasl_sha1 ()

```
int
gsasl_sha1 (const char *in,
            size_t inlen,
            char *out[]);
```

Warning

`gsasl_sha1` is deprecated and should not be used in newly-written code.
Use a crypto library.

Compute hash of data using SHA1. The *out* buffer must be deallocated by the caller.

Parameters

in	input character array of data to hash.	
inlen	length of input character array of data to hash.	
out	newly allocated 20-byte character array with hash of data.	

Returns

Returns GSASL_OK iff successful.

Since: 1.3

gsasl_hmac_sha1 ()

```
int
gsasl_hmac_sha1 (const char *key,
                 size_t keylen,
                 const char *in,
                 size_t inlen,
                 char *outhash[]);
```

Warning
`gsasl_hmac_sha1` is deprecated and should not be used in newly-written code.
Use a crypto library.

Compute keyed checksum of data using HMAC-SHA1. The *`outhash`* buffer must be deallocated by the caller.

Parameters

key	input character array with key to use.	
keylen	length of input character array with key to use.	
in	input character array of data to hash.	
inlen	length of input character array of data to hash.	
outhash	newly allocated 20-byte character array with keyed hash of data.	

Returns

Returns GSASL_OK iff successful.

Since: 1.3

Types and Values

Chapter 2

Index