GNU SASL Reference Manual

COLLABORATORS			
	TITLE : GNU SASL Reference Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 17, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

List of Figures

Chapter 1

GNU SASL Reference Manual

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

GNU SASL consists of a C library (libgsasl), a command-line application (gsasl), and a manual. The library supports the ANONYMOUS, CRAM-MD5, DIGEST-MD5, EXTERNAL, GS2-KRB5, GSSAPI, LOGIN, NTLM, OPENID20, PLAIN, SCRAM-SHA-1, SCRAM-SHA-1-PLUS, SCRAM-SHA-256, SCRAM-SHA-256-PLUS, SAML20, and SECURID mechanisms.

The design of the library and the intended interaction between applications and the library through the official API is shown in Figure 1.1.

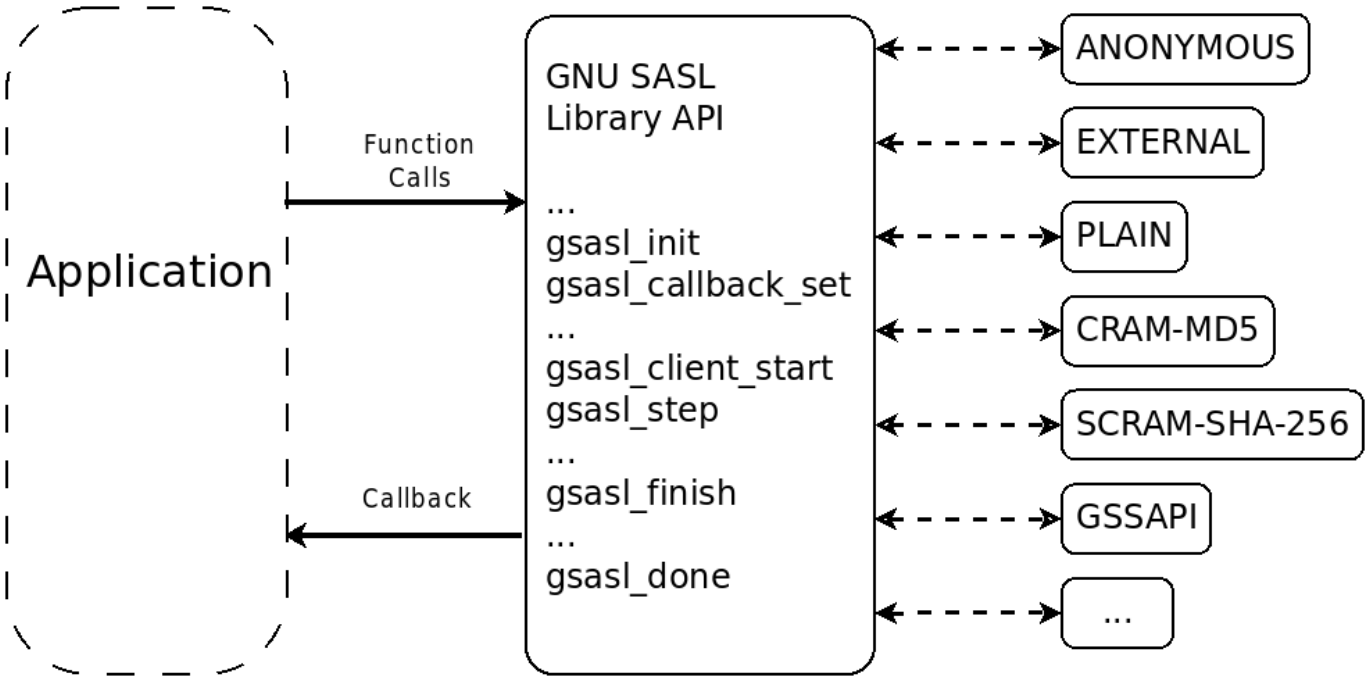

Figure 1.1: Illustration of separation between application and individual mechanism

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.

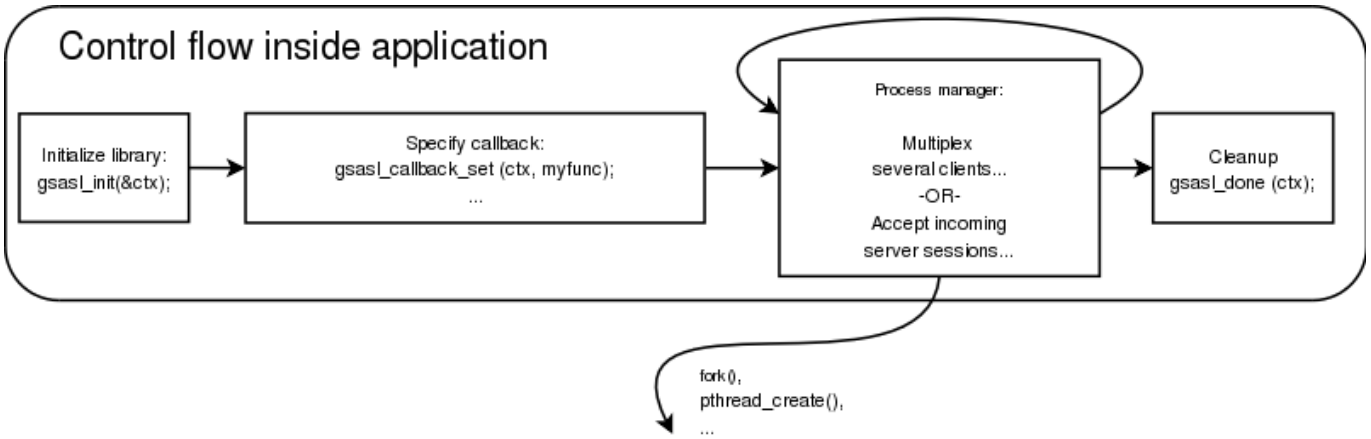

Figure 1.2: High-level control flow of SASL application

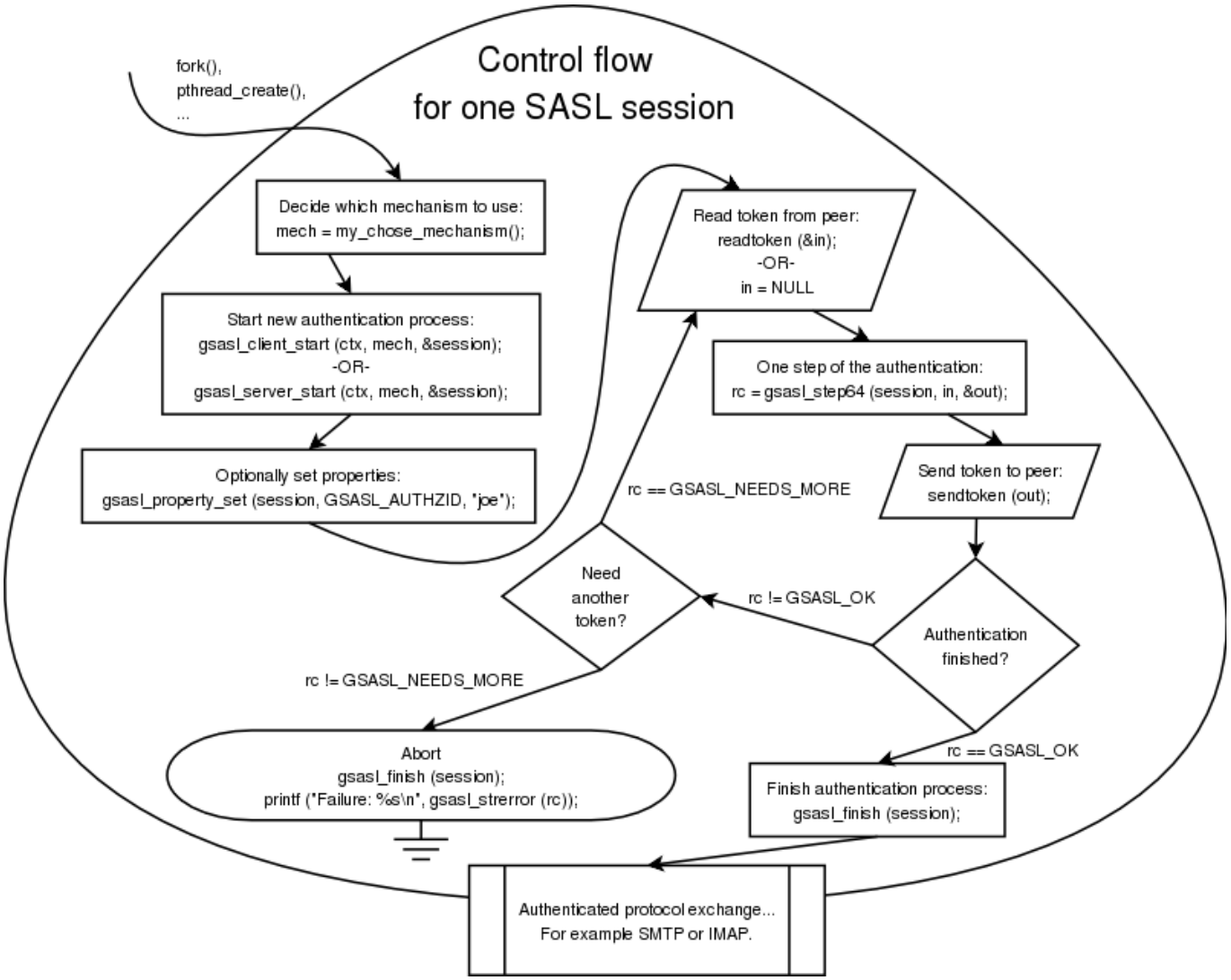

Figure 1.3: Low-level control flow of SASL application

1.1 gsasl-version.h

gsasl-version.h — version symbols

Types and Values

#define	GSASL_VERSION
#define	GSASL_VERSION_MAJOR
#define	GSASL_VERSION_MINOR
#define	GSASL_VERSION_PATCH
#define	GSASL_VERSION_NUMBER

Description

The gsasl-version.h file contains version symbols. It should not be included directly, only via gsasl.h.

Functions

Types and Values

GSASL_VERSION

```
# define GSASL_VERSION "2.2.0"
```

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 2
```

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 2
```

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 0
```

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 0x020200
```

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

1.2 gsasl.h

gsasl.h — main library interfaces

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 ()
int	gsasl_property_set ()
int	gsasl_property_set_raw ()
void	gsasl_property_free ()
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_mechanism_name_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

typedef	Gsasl
typedef	Gsasl_session
enum	Gsasl_rc
enum	Gsasl_property
enum	Gsasl_mechname_limits
enum	Gsasl_qop
enum	Gsasl_saslprep_flags
enum	Gsasl_hash
enum	Gsasl_hash_length

Description

The main library interfaces are declared in gsasl.h.

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.

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 ()

```
int
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.	

Returns

GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Since: 0.2.0

gsasl_property_set_raw ()

```
int
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.	

Returns

GSASL_OK iff successful, otherwise GSASL_MALLOC_ERROR.

Since: 0.2.0

gsasl_property_free ()

```
void
gsasl_property_free (Gsasl_session *sctx,
                     Gsasl_property prop);
```

Deallocate associated data with property *prop* in session handle. After this call, gsasl_property_fast(*sctx* , *prop*) will always return NULL.

Parameters

sctx	session handle.	
prop	enumerated value of Gsasl_property type to clear	

Since: 2.0.0

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.

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_mechanism_name_p ()

```
int
gsasl_mechanism_name_p (const char *mech);
```

Check if the mechanism name string *mech* follows syntactical rules. It does not check that the name is registered with IANA. It does not check that the mechanism name is actually implemented and supported.

SASL mechanisms are named by strings, from 1 to 20 characters in length, consisting of upper-case letters, digits, hyphens, and/or underscores.

Parameters

mech	input variable with mechanism name string.	

Returns

non-zero when mechanism name string *mech* conforms to rules, zero when it does not meet the requirements.

Since: 2.0.0

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 gsasl_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 gsasl_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 gsasl_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 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 gsasl_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 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

```
typedef struct Gsasl Gsasl;
```

Handle to global library context.

Gsasl_session

```
typedef struct Gsasl_session Gsasl_session;
```

Handle to SASL session context.

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_NO_CB_TLS_EXPORTER	Could not get required tls-exporter CB.
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_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	GSS-API library call error.
GSASL_GSSAPI_DECAPSULATE_TOKEN_ERROR	GSS-API library call error.
GSASL_GSSAPI_INQUIRE_MECH_FOR_SASLNAME_ERROR	GSS-API library call error.
GSASL_GSSAPI_TEST_OID_SET_MEMBER_ERROR	GSS-API library call error.
GSASL_GSSAPI_RELEASE_OID_SET_ERROR	GSS-API library call error.

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	Precomputed hashed DIGEST-MD5 password, to avoid storing passwords in the clear.

GSASL_QOPS	Set of quality-of-protection values.
GSASL_QOP	Quality-of-protection value.
GSASL_SCRAM_ITER	Number of iterations in password-to-key hashing.
GSASL_SCRAM_SALT	Salt for password-to-key hashing.
GSASL_SCRAM_SALTED_PASSWORD	Hex-encoded hashed/salted password.
GSASL_SCRAM_SERVERKEY	Hex-encoded SCRAM ServerKey derived from users' pasowrd.
GSASL_SCRAM_STOREDKEY	Hex-encoded SCRAM StoredKey derived from users' pasowrd.

GSASL_CB_TLS_UNIQUE	Base64 encoded tls-unique channel binding.
GSASL_SAML20_IDP_IDENTIFIER	SAML20 user IdP URL.
GSASL_SAML20_REDIRECT_URL	SAML 2.0 URL to access in browser.
GSASL_OPENID20_REDIRECT_URL	OpenID 2.0 URL to access in browser.
GSASL_OPENID20_OUTCOME_DATA	OpenID 2.0 authentication outcome data.
GSASL_CB_TLS_EXPORTER	Base64 encoded tls-exporter channel binding.

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_mechname_limits

SASL mechanisms are named by strings, from 1 to 20 characters in length, consisting of upper-case letters, digits, hyphens, and/or underscores. See also gsasl_mechanism_name_p().

Members

GSASL_MIN_MECHANISM_SIZE	Minimum size of mechanism name strings.

GSASL_MAX_MECHANISM_SIZE	Maximum size of mechanism name strings.

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_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.

enum Gsasl_hash

Hash functions. You may use gsasl_hash_length() to get the output size of a hash function.

Currently only used as parameter to gsasl_scram_secrets_from_salted_password() and gsasl_scram_secrets_from_password() to specify for which SCRAM mechanism to prepare secrets for.

Members

GSASL_HASH_SHA1	Hash function SHA-1.
GSASL_HASH_SHA256	Hash function SHA-256.

Since: 1.10

enum Gsasl_hash_length

Identifiers specifying the output size of hash functions.

These can be used when statically allocating the buffers needed for, e.g., gsasl_scram_secrets_from_password().

Members

GSASL_HASH_SHA1_SIZE	Output size of hash function SHA-1.
GSASL_HASH_SHA256_SIZE	Output size of hash function SHA-256.
GSASL_HASH_MAX_SIZE	Maximum output size of any Gsasl_hash_length.

Since: 1.10

1.3 gsasl-mech.h

gsasl-mech.h — register new application-defined mechanism

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

The builtin mechanisms should suffice for most applications. Applications can register a new mechanism in the library using application-supplied functions. The mechanism will operate as the builtin mechanisms, and the supplied functions will be invoked when necessary. The application uses the normal logic, e.g., calls gsasl_client_start() followed by a sequence of calls to gsasl_step() and finally gsasl_finish().

Functions

Gsasl_init_function ()

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

The implementation of this function pointer should fail if the mechanism for some reason is not available for further use.

Parameters

ctx	a Gsasl libgsasl handle.	

Returns

Returns GSASL_OK iff successful.

Gsasl_done_function ()

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

The implementation of this function pointer deallocate all resources associated with the mechanism.

Parameters

ctx	a Gsasl libgsasl handle.	

Gsasl_start_function ()

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

The implementation of this function should start a new authentication process.

Parameters

sctx	a Gsasl_session session handle.	
mech_data	pointer to void* with mechanism-specific data.	

Returns

Returns GSASL_OK iff successful.

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);
```

The implementation of this function should perform one step of the authentication process.

This reads data from the other end (from *input* and *input_len*), processes it (potentially invoking callbacks to the application), and writes data to server (into newly allocated variable *output* and *output_len* that indicate the length of *output*).

The contents of the *output* buffer is unspecified if this functions returns anything other than GSASL_OK or GSASL_NEEDS_MORE. If this function return GSASL_OK or GSASL_NEEDS_MORE, however, the *output* buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl_free(*output*).

Parameters

sctx	a Gsasl_session session handle.	
mech_data	pointer to void* with mechanism-specific data.	
input	input byte array.	
input_len	size of input byte array.	
output	newly allocated output byte array.	
output_len	pointer to output variable with size of output byte array.	

Returns

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

Gsasl_finish_function ()

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

The implementation of this function should release all resources associated with the particular authentication process.

Parameters

sctx	a Gsasl_session session handle.	
mech_data	pointer to void* with mechanism-specific 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);
```

The implementation of this function should perform data encoding or decoding for the mechanism, after authentication has completed. This might mean that data is integrity or privacy protected.

The *output* buffer is allocated by this function, and it is the responsibility of caller to deallocate it by calling gsasl_free(*output*).

Parameters

sctx	a Gsasl_session session handle.	
mech_data	pointer to void* with mechanism-specific data.	
input	input byte array.	
input_len	size of input byte array.	
output	newly allocated output byte array.	
output_len	pointer to output variable with size of output byte array.	

Returns

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

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;
};
```

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

Members

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

struct Gsasl_mechanism

```
struct Gsasl_mechanism {
  const char *name;

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

Holds all implementation details about a mechanism.

Members

const char *`name`;	string holding name of mechanism, e.g., "PLAIN".
struct Gsasl_mechanism_functions `client`;	client-side Gsasl_mechanism_functions structure.
struct Gsasl_mechanism_functions `server`;	server-side Gsasl_mechanism_functions structure.

Chapter 2

Index