

GNU Generic Security Service Library

GSS-API Library for the GNU system
for version 1.0.4, 6 August 2022

Simon Josefsson

This manual is last updated 6 August 2022 for version 1.0.4 of GNU GSS.

Copyright © 2003–2022 Simon Josefsson.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

Table of Contents

1	Introduction	1
1.1	Getting Started	1
1.2	Features	1
1.3	GSS-API Overview	1
1.4	Supported Platforms	2
1.5	Commercial Support	4
1.6	Downloading and Installing	4
1.7	Bug Reports	5
1.8	Contributing	5
1.9	Planned Features	6
2	Preparation	7
2.1	Header	7
2.2	Initialization	7
2.3	Version Check	7
2.4	Building the source	8
2.5	Out of Memory handling	8
3	Standard GSS API	9
3.1	Simple Data Types	9
3.1.1	Integer types	9
3.1.2	String and similar data	9
3.1.2.1	Opaque data types	9
3.1.2.2	Character strings	9
3.1.3	Object Identifiers	10
3.1.4	Object Identifier Sets	10
3.2	Complex Data Types	10
3.2.1	Credentials	11
3.2.2	Contexts	11
3.2.3	Authentication tokens	11
3.2.4	Interprocess tokens	11
3.2.5	Names	12
3.2.6	Channel Bindings	13
3.3	Optional Parameters	15
3.4	Error Handling	15
3.4.1	GSS status codes	15
3.4.2	Mechanism-specific status codes	18
3.5	Credential Management	18
3.6	Context-Level Routines	20
3.7	Per-Message Routines	22
3.8	Name Manipulation	23
3.9	Miscellaneous Routines	25
3.10	SASL GS2 Routines	27

4	Extended GSS API.....	28
5	Invoking gss.....	29
6	Acknowledgements	31
	Appendix A Criticism of GSS.....	32
	Appendix B Copying Information.....	34
	B.1 GNU Free Documentation License.....	34
	Concept Index	42
	API Index	43

1 Introduction

GSS is an implementation of the Generic Security Service Application Program Interface (GSS-API). GSS-API is used by network servers to provide security services, e.g., to authenticate SMTP/IMAP clients against SMTP/IMAP servers. GSS consists of a library and a manual.

GSS is developed for the GNU/Linux system but works on many platforms including most major Unix and Windows systems.

1.1 Getting Started

This manual documents the GSS programming interface. All functions and data types provided by the library are explained.

The reader is assumed to possess basic familiarity with GSS-API and network programming in C or C++. For general GSS-API information, and some programming examples, there is a guide available online at <http://docs.sun.com/db/doc/816-1331>.

This manual can be used in several ways. If read from the beginning to the end, it gives a good introduction into the library and how it can be used in an application. Forward references are included where necessary. Later on, the manual can be used as a reference manual to get just the information needed about any particular interface of the library. Experienced programmers might want to start looking at the examples at the end of the manual, and then only read up those parts of the interface which are unclear.

1.2 Features

GSS might have a couple of advantages over other libraries doing a similar job.

It's Free Software

Anybody can use, modify, and redistribute it under the terms of the LGPLv3|GPLv2+ license.

It's thread-safe

No global variables are used and multiple library handles and session handles may be used in parallel.

It's internationalized

It handles non-ASCII names and user visible strings used in the library (e.g., error messages) can be translated into the users' language.

It's portable

It should work on all Unix like operating systems, including Windows.

1.3 GSS-API Overview

This section describes GSS-API from a protocol point of view.

The Generic Security Service Application Programming Interface provides security services to calling applications. It allows a communicating application to authenticate the user associated with another application, to delegate rights to another application, and to apply security services such as confidentiality and integrity on a per-message basis.

There are four stages to using the GSS-API:

1. The application acquires a set of credentials with which it may prove its identity to other processes. The application's credentials vouch for its global identity, which may or may not be related to any local username under which it may be running.
2. A pair of communicating applications establish a joint security context using their credentials. The security context is a pair of GSS-API data structures that contain shared state information, which is required in order that per-message security services may be provided. Examples of state that might be shared between applications as part of a security context are cryptographic keys, and message sequence numbers. As part of the establishment of a security context, the context initiator is authenticated to the responder, and may require that the responder is authenticated in turn. The initiator may optionally give the responder the right to initiate further security contexts, acting as an agent or delegate of the initiator. This transfer of rights is termed delegation, and is achieved by creating a set of credentials, similar to those used by the initiating application, but which may be used by the responder.

To establish and maintain the shared information that makes up the security context, certain GSS-API calls will return a token data structure, which is an opaque data type that may contain cryptographically protected data. The caller of such a GSS-API routine is responsible for transferring the token to the peer application, encapsulated if necessary in an application- application protocol. On receipt of such a token, the peer application should pass it to a corresponding GSS-API routine which will decode the token and extract the information, updating the security context state information accordingly.

3. Per-message services are invoked to apply either: integrity and data origin authentication, or confidentiality, integrity and data origin authentication to application data, which are treated by GSS-API as arbitrary octet-strings. An application transmitting a message that it wishes to protect will call the appropriate GSS-API routine (`gss_get_mic` or `gss_wrap`) to apply protection, specifying the appropriate security context, and send the resulting token to the receiving application. The receiver will pass the received token (and, in the case of data protected by `gss_get_mic`, the accompanying message-data) to the corresponding decoding routine (`gss_verify_mic` or `gss_unwrap`) to remove the protection and validate the data.
4. At the completion of a communications session (which may extend across several transport connections), each application calls a GSS-API routine to delete the security context. Multiple contexts may also be used (either successively or simultaneously) within a single communications association, at the option of the applications.

1.4 Supported Platforms

GSS has at some point in time been tested on the following platforms.

1. Debian GNU/Linux 3.0 (Woody)

GCC 2.95.4 and GNU Make. This is the main development platform. `alphaev67-unknown-linux-gnu`, `alphaev6-unknown-linux-gnu`, `arm-unknown-linux-gnu`, `hppa-unknown-linux-gnu`, `hppa64-unknown-linux-gnu`, `i686-pc-linux-gnu`, `ia64-unknown-linux-gnu`, `m68k-unknown-linux-gnu`, `mips-unknown-linux-gnu`,

- `mipsel-unknown-linux-gnu`, `powerpc-unknown-linux-gnu`, `s390-ibm-linux-gnu`, `sparc-unknown-linux-gnu`.
2. Debian GNU/Linux 2.1
GCC 2.95.1 and GNU Make. `armv4l-unknown-linux-gnu`.
 3. Tru64 UNIX
Tru64 UNIX C compiler and Tru64 Make. `alphaev67-dec-osf5.1`, `alphaev68-dec-osf5.1`.
 4. SuSE Linux 7.1
GCC 2.96 and GNU Make. `alphaev6-unknown-linux-gnu`, `alphaev67-unknown-linux-gnu`.
 5. SuSE Linux 7.2a
GCC 3.0 and GNU Make. `ia64-unknown-linux-gnu`.
 6. RedHat Linux 7.2
GCC 2.96 and GNU Make. `alphaev6-unknown-linux-gnu`, `alphaev67-unknown-linux-gnu`, `ia64-unknown-linux-gnu`.
 7. RedHat Linux 8.0
GCC 3.2 and GNU Make. `i686-pc-linux-gnu`.
 8. RedHat Advanced Server 2.1
GCC 2.96 and GNU Make. `i686-pc-linux-gnu`.
 9. Slackware Linux 8.0.01
GCC 2.95.3 and GNU Make. `i686-pc-linux-gnu`.
 10. Mandrake Linux 9.0
GCC 3.2 and GNU Make. `i686-pc-linux-gnu`.
 11. IRIX 6.5
MIPS C compiler, IRIX Make. `mips-sgi-irix6.5`.
 12. AIX 4.3.2
IBM C for AIX compiler, AIX Make. `rs6000-ibm-aix4.3.2.0`.
 13. Microsoft Windows 2000 (Cygwin)
GCC 3.2, GNU make. `i686-pc-cygwin`.
 14. HP-UX 11
HP-UX C compiler and HP Make. `ia64-hp-hpux11.22`, `hppa2.0w-hp-hpux11.11`.
 15. SUN Solaris 2.8
Sun WorkShop Compiler C 6.0 and SUN Make. `sparc-sun-solaris2.8`.
 16. NetBSD 1.6
GCC 2.95.3 and GNU Make. `alpha-unknown-netbsd1.6`, `i386-unknown-netbsdelf1.6`.
 17. OpenBSD 3.1 and 3.2
GCC 2.95.3 and GNU Make. `alpha-unknown-openbsd3.1`, `i386-unknown-openbsd3.1`.

18. FreeBSD 4.7

GCC 2.95.4 and GNU Make. `alpha-unknown-freebsd4.7`, `i386-unknown-freebsd4.7`.

19. Cross compiled to uClinux/uClibc on Motorola Coldfire.

GCC 3.4 and GNU Make `m68k-uclinux-elf`.

If you use GSS on, or port GSS to, a new platform please report it to the author.

1.5 Commercial Support

Commercial support is available for users of GNU GSS. The kind of support that can be purchased may include:

- Implement new features. Such as a new GSS-API mechanism.
- Port GSS to new platforms. This could include porting to an embedded platforms that may need memory or size optimization.
- Integrating GSS as a security environment in your existing project.
- System design of components related to GSS-API.

If you are interested, please write to:

Simon Josefsson Datakonsult AB
Hagagatan 24
113 47 Stockholm
Sweden

E-mail: simon@josefsson.org

If your company provides support related to GNU GSS and would like to be mentioned here, contact the author (see Section 1.7 [Bug Reports], page 5).

1.6 Downloading and Installing

The package can be downloaded from several places, including:

`ftp://ftp.gnu.org/gnu/gss/`

The latest version is stored in a file, e.g., `'gss-1.0.4.tar.gz'` where the `'1.0.4'` indicate the highest version number.

The package is then extracted, configured and built like many other packages that use Autoconf. For detailed information on configuring and building it, refer to the `INSTALL` file that is part of the distribution archive.

Here is an example terminal session that downloads, configures, builds and installs the package. You will need a few basic tools, such as `'sh'`, `'make'` and `'cc'`.

```
$ wget -q ftp://ftp.gnu.org/gnu/gss/gss-1.0.4.tar.gz
$ tar xzf gss-1.0.4.tar.gz
$ cd gss-1.0.4/
$ ./configure
...
$ make
...
```

```
$ make install
...
```

After that GSS should be properly installed and ready for use.

1.7 Bug Reports

If you think you have found a bug in GSS, please investigate it and report it.

- Please make sure that the bug is really in GSS, and preferably also check that it hasn't already been fixed in the latest version.
- You have to send us a test case that makes it possible for us to reproduce the bug.
- You also have to explain what is wrong; if you get a crash, or if the results printed are not good and in that case, in what way. Make sure that the bug report includes all information you would need to fix this kind of bug for someone else.

Please make an effort to produce a self-contained report, with something definite that can be tested or debugged. Vague queries or piecemeal messages are difficult to act on and don't help the development effort.

If your bug report is good, we will do our best to help you to get a corrected version of the software; if the bug report is poor, we won't do anything about it (apart from asking you to send better bug reports).

If you think something in this manual is unclear, or downright incorrect, or if the language needs to be improved, please also send a note.

Send your bug report to:

`'bug-gss@gnu.org'`

1.8 Contributing

If you want to submit a patch for inclusion – from solve a typo you discovered, up to adding support for a new feature – you should submit it as a bug report (see Section 1.7 [Bug Reports], page 5). There are some things that you can do to increase the chances for it to be included in the official package.

Unless your patch is very small (say, under 10 lines) we require that you assign the copyright of your work to the Free Software Foundation. This is to protect the freedom of the project. If you have not already signed papers, we will send you the necessary information when you submit your contribution.

For contributions that doesn't consist of actual programming code, the only guidelines are common sense. Use it.

For code contributions, a number of style guides will help you:

- Coding Style. Follow the GNU Standards document (see Section “top” in **standards**). If you normally code using another coding standard, there is no problem, but you should use ‘indent’ to reformat the code (see Section “top” in **indent**) before submitting your work.
- Use the unified diff format ‘diff -u’.
- Return errors. No reason whatsoever should abort the execution of the library. Even memory allocation errors, e.g. when malloc return NULL, should work although result in an error code.

- Design with thread safety in mind. Don't use global variables. Don't even write to per-handle global variables unless the documented behaviour of the function you write is to write to the per-handle global variable.
- Avoid using the C math library. It causes problems for embedded implementations, and in most situations it is very easy to avoid using it.
- Document your functions. Use comments before each function headers, that, if properly formatted, are extracted into Texinfo manuals and GTK-DOC web pages.
- Supply a ChangeLog and NEWS entries, where appropriate.

1.9 Planned Features

This is also known as the “todo list”. If you like to start working on anything, please let me know so work duplication can be avoided.

- Support non-blocking mode. This would be an API extension. It could work by forking a process and interface to it, or by using a user-specific daemon. E.g., `h = START(accept_sec_context(...))`, `FINISHED(h)`, `ret = FINISH(h)`, `ABORT(h)`.
- Support loadable modules via `dlopen`, a'la Solaris GSS.
- Port to Cyclone? CCured?

2 Preparation

To use GSS, you have to perform some changes to your sources and the build system. The necessary changes are small and explained in the following sections. At the end of this chapter, it is described how the library is initialized, and how the requirements of the library are verified.

A faster way to find out how to adapt your application for use with GSS may be to look at the examples at the end of this manual.

2.1 Header

All standard interfaces (data types and functions) of the official GSS API are defined in the header file `gss/api.h`. The file is taken verbatim from the RFC (after correcting a few typos) where it is known as `gssapi.h`. However, to be able to co-exist gracefully with other GSS-API implementation, the name `gssapi.h` was changed.

The header file `gss.h` includes `gss/api.h`, and declares a few non-standard extensions (by including `gss/ext.h`), takes care of including header files related to all supported mechanisms (e.g., `gss/krb5.h`) and finally adds C++ namespace protection of all definitions. Therefore, including `gss.h` in your project is recommended over `gss/api.h`. If using `gss.h` instead of `gss/api.h` causes problems, it should be regarded a bug.

You must include either file in all programs using the library, either directly or through some other header file, like this:

```
#include <gss.h>
```

The name space of GSS is `gss_*` for function names, `gss_*` for data types and `GSS_*` for other symbols. In addition the same name prefixes with one prepended underscore are reserved for internal use and should never be used by an application.

Each supported GSS mechanism may want to expose mechanism specific functionality, and can do so through one or more header files under the `gss/` directory. The Kerberos 5 mechanism uses the file `gss/krb5.h`, but again, it is included (with C++ namespace fixes) from `gss.h`.

2.2 Initialization

GSS does not need to be initialized before it can be used.

In order to take advantage of the internationalisation features in GSS, e.g. translated error messages, the application must set the current locale using `setlocale()` before calling, e.g., `gss_display_status()`. This is typically done in `main()` as in the following example.

```
#include <gss.h>
#include <locale.h>
...
setlocale (LC_ALL, "");
```

2.3 Version Check

It is often desirable to check that the version of GSS used is indeed one which fits all requirements. Even with binary compatibility new features may have been introduced but

due to problem with the dynamic linker an old version is actually used. So you may want to check that the version is okay right after program startup. The function is called `gss_check_version()` and is described formally in See Chapter 4 [Extended GSS API], page 28.

The normal way to use the function is to put something similar to the following early in your `main()`:

```
#include <gss.h>
...
if (!gss_check_version (GSS_VERSION))
{
    printf ("gss_check_version() failed:\n"
           "Header file incompatible with shared library.\n");
    exit(EXIT_FAILURE);
}
```

2.4 Building the source

If you want to compile a source file that includes the `gss.h` header file, you must make sure that the compiler can find it in the directory hierarchy. This is accomplished by adding the path to the directory in which the header file is located to the compilers include file search path (via the `-I` option).

However, the path to the include file is determined at the time the source is configured. To solve this problem, GSS uses the external package `pkg-config` that knows the path to the include file and other configuration options. The options that need to be added to the compiler invocation at compile time are output by the `--cflags` option to `pkg-config gss`. The following example shows how it can be used at the command line:

```
gcc -c foo.c `pkg-config gss --cflags`
```

Adding the output of `'pkg-config gss --cflags'` to the compilers command line will ensure that the compiler can find the `gss.h` header file.

A similar problem occurs when linking the program with the library. Again, the compiler has to find the library files. For this to work, the path to the library files has to be added to the library search path (via the `-L` option). For this, the option `--libs` to `pkg-config gss` can be used. For convenience, this option also outputs all other options that are required to link the program with the GSS library (for instance, the `'-lshishi'` option). The example shows how to link `foo.o` with GSS into a program `foo`.

```
gcc -o foo foo.o `pkg-config gss --libs`
```

Of course you can also combine both examples to a single command by specifying both options to `pkg-config`:

```
gcc -o foo foo.c `pkg-config gss --cflags --libs`
```

2.5 Out of Memory handling

The GSS API does not have a standard error code for the out of memory error condition. This library will return `GSS_S_FAILURE` and set `minor_status` to `ENOMEM`.

3 Standard GSS API

3.1 Simple Data Types

The following conventions are used by the GSS-API C-language bindings:

3.1.1 Integer types

GSS-API uses the following integer data type:

```
OM_uint32    32-bit unsigned integer
```

3.1.2 String and similar data

Many of the GSS-API routines take arguments and return values that describe contiguous octet-strings. All such data is passed between the GSS-API and the caller using the `gss_buffer_t` data type. This data type is a pointer to a buffer descriptor, which consists of a length field that contains the total number of bytes in the datum, and a value field which contains a pointer to the actual datum:

```
typedef struct gss_buffer_desc_struct {
    size_t    length;
    void      *value;
} gss_buffer_desc, *gss_buffer_t;
```

Storage for data returned to the application by a GSS-API routine using the `gss_buffer_t` conventions is allocated by the GSS-API routine. The application may free this storage by invoking the `gss_release_buffer` routine. Allocation of the `gss_buffer_desc` object is always the responsibility of the application; unused `gss_buffer_desc` objects may be initialized to the value `GSS_C_EMPTY_BUFFER`.

3.1.2.1 Opaque data types

Certain multiple-word data items are considered opaque data types at the GSS-API, because their internal structure has no significance either to the GSS-API or to the caller. Examples of such opaque data types are the `input_token` parameter to `gss_init_sec_context` (which is opaque to the caller), and the `input_message` parameter to `gss_wrap` (which is opaque to the GSS-API). Opaque data is passed between the GSS-API and the application using the `gss_buffer_t` datatype.

3.1.2.2 Character strings

Certain multiple-word data items may be regarded as simple ISO Latin-1 character strings. Examples are the printable strings passed to `gss_import_name` via the `input_name_buffer` parameter. Some GSS-API routines also return character strings. All such character strings are passed between the application and the GSS-API implementation using the `gss_buffer_t` datatype, which is a pointer to a `gss_buffer_desc` object.

When a `gss_buffer_desc` object describes a printable string, the length field of the `gss_buffer_desc` should only count printable characters within the string. In particular, a trailing NUL character should NOT be included in the length count, nor should either the GSS-API implementation or the application assume the presence of an uncounted trailing NUL.

3.1.3 Object Identifiers

Certain GSS-API procedures take parameters of the type `gss_OID`, or Object identifier. This is a type containing ISO-defined tree-structured values, and is used by the GSS-API caller to select an underlying security mechanism and to specify namespaces. A value of type `gss_OID` has the following structure:

```
typedef struct gss_OID_desc_struct {
    OM_uint32    length;
    void         *elements;
} gss_OID_desc, *gss_OID;
```

The elements field of this structure points to the first byte of an octet string containing the ASN.1 BER encoding of the value portion of the normal BER TLV encoding of the `gss_OID`. The length field contains the number of bytes in this value. For example, the `gss_OID` value corresponding to `iso(1) identified-organization(3) icd-ecma(12) member-company(2) dec(1011) cryptoAlgorithms(7) DASS(5)`, meaning the DASS X.509 authentication mechanism, has a length field of 7 and an elements field pointing to seven octets containing the following octal values: 53,14,2,207,163,7,5. GSS-API implementations should provide constant `gss_OID` values to allow applications to request any supported mechanism, although applications are encouraged on portability grounds to accept the default mechanism. `gss_OID` values should also be provided to allow applications to specify particular name types (see section 3.10). Applications should treat `gss_OID_desc` values returned by GSS-API routines as read-only. In particular, the application should not attempt to deallocate them with `free()`.

3.1.4 Object Identifier Sets

Certain GSS-API procedures take parameters of the type `gss_OID_set`. This type represents one or more object identifiers (see [Object Identifiers], page 10). A `gss_OID_set` object has the following structure:

```
typedef struct gss_OID_set_desc_struct {
    size_t      count;
    gss_OID     elements;
} gss_OID_set_desc, *gss_OID_set;
```

The count field contains the number of OIDs within the set. The elements field is a pointer to an array of `gss_OID_desc` objects, each of which describes a single OID. `gss_OID_set` values are used to name the available mechanisms supported by the GSS-API, to request the use of specific mechanisms, and to indicate which mechanisms a given credential supports.

All OID sets returned to the application by GSS-API are dynamic objects (the `gss_OID_set_desc`, the "elements" array of the set, and the "elements" array of each member OID are all dynamically allocated), and this storage must be deallocated by the application using the `gss_release_oid_set` routine.

3.2 Complex Data Types

3.2.1 Credentials

A credential handle is a caller-opaque atomic datum that identifies a GSS-API credential data structure. It is represented by the caller- opaque type `gss_cred_id_t`.

GSS-API credentials can contain mechanism-specific principal authentication data for multiple mechanisms. A GSS-API credential is composed of a set of credential-elements, each of which is applicable to a single mechanism. A credential may contain at most one credential-element for each supported mechanism. A credential-element identifies the data needed by a single mechanism to authenticate a single principal, and conceptually contains two credential-references that describe the actual mechanism-specific authentication data, one to be used by GSS-API for initiating contexts, and one to be used for accepting contexts. For mechanisms that do not distinguish between acceptor and initiator credentials, both references would point to the same underlying mechanism-specific authentication data.

Credentials describe a set of mechanism-specific principals, and give their holder the ability to act as any of those principals. All principal identities asserted by a single GSS-API credential should belong to the same entity, although enforcement of this property is an implementation-specific matter. The GSS-API does not make the actual credentials available to applications; instead a credential handle is used to identify a particular credential, held internally by GSS-API. The combination of GSS-API credential handle and mechanism identifies the principal whose identity will be asserted by the credential when used with that mechanism.

The `gss_init_sec_context` and `gss_accept_sec_context` routines allow the value `GSS_C_NO_CREDENTIAL` to be specified as their credential handle parameter. This special credential-handle indicates a desire by the application to act as a default principal.

3.2.2 Contexts

The `gss_ctx_id_t` data type contains a caller-opaque atomic value that identifies one end of a GSS-API security context.

The security context holds state information about each end of a peer communication, including cryptographic state information.

3.2.3 Authentication tokens

A token is a caller-opaque type that GSS-API uses to maintain synchronization between the context data structures at each end of a GSS-API security context. The token is a cryptographically protected octet-string, generated by the underlying mechanism at one end of a GSS-API security context for use by the peer mechanism at the other end. Encapsulation (if required) and transfer of the token are the responsibility of the peer applications. A token is passed between the GSS-API and the application using the `gss_buffer_t` conventions.

3.2.4 Interprocess tokens

Certain GSS-API routines are intended to transfer data between processes in multi-process programs. These routines use a caller-opaque octet-string, generated by the GSS-API in one process for use by the GSS-API in another process. The calling application is responsible for transferring such tokens between processes in an OS-specific manner. Note that, while GSS-API implementors are encouraged to avoid placing sensitive information within interprocess tokens, or to cryptographically protect them, many implementations will be unable to

avoid placing key material or other sensitive data within them. It is the application's responsibility to ensure that interprocess tokens are protected in transit, and transferred only to processes that are trustworthy. An interprocess token is passed between the GSS-API and the application using the `gss_buffer_t` conventions.

3.2.5 Names

A name is used to identify a person or entity. GSS-API authenticates the relationship between a name and the entity claiming the name.

Since different authentication mechanisms may employ different namespaces for identifying their principals, GSSAPI's naming support is necessarily complex in multi-mechanism environments (or even in some single-mechanism environments where the underlying mechanism supports multiple namespaces).

Two distinct representations are defined for names:

- An internal form. This is the GSS-API "native" format for names, represented by the implementation-specific `gss_name_t` type. It is opaque to GSS-API callers. A single `gss_name_t` object may contain multiple names from different namespaces, but all names should refer to the same entity. An example of such an internal name would be the name returned from a call to the `gss_inquire_cred` routine, when applied to a credential containing credential elements for multiple authentication mechanisms employing different namespaces. This `gss_name_t` object will contain a distinct name for the entity for each authentication mechanism.

For GSS-API implementations supporting multiple namespaces, objects of type `gss_name_t` must contain sufficient information to determine the namespace to which each primitive name belongs.

- Mechanism-specific contiguous octet-string forms. A format capable of containing a single name (from a single namespace). Contiguous string names are always accompanied by an object identifier specifying the namespace to which the name belongs, and their format is dependent on the authentication mechanism that employs the name. Many, but not all, contiguous string names will be printable, and may therefore be used by GSS-API applications for communication with their users.

Routines (`gss_import_name` and `gss_display_name`) are provided to convert names between contiguous string representations and the internal `gss_name_t` type. `gss_import_name` may support multiple syntaxes for each supported namespace, allowing users the freedom to choose a preferred name representation. `gss_display_name` should use an implementation-chosen printable syntax for each supported name-type.

If an application calls `gss_display_name`, passing the internal name resulting from a call to `gss_import_name`, there is no guarantee the resulting contiguous string name will be the same as the original imported string name. Nor do name-space identifiers necessarily survive unchanged after a journey through the internal name-form. An example of this might be a mechanism that authenticates X.500 names, but provides an algorithmic mapping of Internet DNS names into X.500. That mechanism's implementation of `gss_import_name` might, when presented with a DNS name, generate an internal name that contained both the original DNS name and the equivalent X.500 name. Alternatively, it might only store the X.500 name. In the latter case, `gss_display_name` would most likely generate a printable X.500 name, rather than the original DNS name.

The process of authentication delivers to the context acceptor an internal name. Since this name has been authenticated by a single mechanism, it contains only a single name (even if the internal name presented by the context initiator to `gss_init_sec_context` had multiple components). Such names are termed internal mechanism names, or "MN"s and the names emitted by `gss_accept_sec_context` are always of this type. Since some applications may require MNs without wanting to incur the overhead of an authentication operation, a second function, `gss_canonicalize_name`, is provided to convert a general internal name into an MN.

Comparison of internal-form names may be accomplished via the `gss_compare_name` routine, which returns true if the two names being compared refer to the same entity. This removes the need for the application program to understand the syntaxes of the various printable names that a given GSS-API implementation may support. Since GSS-API assumes that all primitive names contained within a given internal name refer to the same entity, `gss_compare_name` can return true if the two names have at least one primitive name in common. If the implementation embodies knowledge of equivalence relationships between names taken from different namespaces, this knowledge may also allow successful comparison of internal names containing no overlapping primitive elements.

When used in large access control lists, the overhead of invoking `gss_import_name` and `gss_compare_name` on each name from the ACL may be prohibitive. As an alternative way of supporting this case, GSS-API defines a special form of the contiguous string name which may be compared directly (e.g. with `memcmp()`). Contiguous names suitable for comparison are generated by the `gss_export_name` routine, which requires an MN as input. Exported names may be re-imported by the `gss_import_name` routine, and the resulting internal name will also be an MN. The `gss_OID` constant `GSS_C_NT_EXPORT_NAME` identifies the "export name" type, and the value of this constant is given in Appendix A. Structurally, an exported name object consists of a header containing an OID identifying the mechanism that authenticated the name, and a trailer containing the name itself, where the syntax of the trailer is defined by the individual mechanism specification. The precise format of an export name is defined in the language-independent GSS-API specification [GSSAPI].

Note that the results obtained by using `gss_compare_name` will in general be different from those obtained by invoking `gss_canonicalize_name` and `gss_export_name`, and then comparing the exported names. The first series of operation determines whether two (unauthenticated) names identify the same principal; the second whether a particular mechanism would authenticate them as the same principal. These two operations will in general give the same results only for MNs.

The `gss_name_t` datatype should be implemented as a pointer type. To allow the compiler to aid the application programmer by performing type-checking, the use of `(void *)` is discouraged. A pointer to an implementation-defined type is the preferred choice.

Storage is allocated by routines that return `gss_name_t` values. A procedure, `gss_release_name`, is provided to free storage associated with an internal-form name.

3.2.6 Channel Bindings

GSS-API supports the use of user-specified tags to identify a given context to the peer application. These tags are intended to be used to identify the particular communications channel that carries the context. Channel bindings are communicated to the GSS-API using the following structure:

```
typedef struct gss_channel_bindings_struct {
    OM_uint32      initiator_addrtype;
    gss_buffer_desc initiator_address;
    OM_uint32      acceptor_addrtype;
    gss_buffer_desc acceptor_address;
    gss_buffer_desc application_data;
} *gss_channel_bindings_t;
```

The `initiator_addrtype` and `acceptor_addrtype` fields denote the type of addresses contained in the `initiator_address` and `acceptor_address` buffers. The address type should be one of the following:

<code>GSS_C_AF_UNSPEC</code>	Unspecified address type
<code>GSS_C_AF_LOCAL</code>	Host-local address type
<code>GSS_C_AF_INET</code>	Internet address type (e.g. IP)
<code>GSS_C_AF_IMPLINK</code>	ARPAnet IMP address type
<code>GSS_C_AF_PUP</code>	pup protocols (eg BSP) address type
<code>GSS_C_AF_CHAOS</code>	MIT CHAOS protocol address type
<code>GSS_C_AF_NS</code>	XEROX NS address type
<code>GSS_C_AF_NBS</code>	nbs address type
<code>GSS_C_AF_ECMA</code>	ECMA address type
<code>GSS_C_AF_DATAKIT</code>	datakit protocols address type
<code>GSS_C_AF_CCITT</code>	CCITT protocols
<code>GSS_C_AF_SNA</code>	IBM SNA address type
<code>GSS_C_AF_DECnet</code>	DECnet address type
<code>GSS_C_AF_DLI</code>	Direct data link interface address type
<code>GSS_C_AF_LAT</code>	LAT address type
<code>GSS_C_AF_HYLINK</code>	NSC Hyperchannel address type
<code>GSS_C_AF_APPLETALK</code>	AppleTalk address type
<code>GSS_C_AF_BSC</code>	BISYNC 2780/3780 address type
<code>GSS_C_AF_DSS</code>	Distributed system services address type
<code>GSS_C_AF_OSI</code>	OSI TP4 address type
<code>GSS_C_AF_X25</code>	X.25
<code>GSS_C_AF_NULLADDR</code>	No address specified

Note that these symbols name address families rather than specific addressing formats. For address families that contain several alternative address forms, the `initiator_address` and `acceptor_address` fields must contain sufficient information to determine which address form is used. When not otherwise specified, addresses should be specified in network byte-order (that is, native byte-ordering for the address family).

Conceptually, the GSS-API concatenates the `initiator_addrtype`, `initiator_address`, `acceptor_addrtype`, `acceptor_address` and `application_data` to form an octet string. The mechanism calculates a MIC over this octet string, and binds the MIC to the context establishment token emitted by `gss_init_sec_context`. The same bindings are presented by the context acceptor to `gss_accept_sec_context`, and a MIC is calculated in the same way. The calculated MIC is compared with that found in the token, and if the MICs differ, `gss_accept_sec_context` will return a `GSS_S_BAD_BINDINGS` error, and the context will not be established. Some mechanisms may include the actual channel binding data in the

token (rather than just a MIC); applications should therefore not use confidential data as channel-binding components.

Individual mechanisms may impose additional constraints on addresses and address types that may appear in channel bindings. For example, a mechanism may verify that the `initiator_address` field of the channel bindings presented to `gss_init_sec_context` contains the correct network address of the host system. Portable applications should therefore ensure that they either provide correct information for the address fields, or omit addressing information, specifying `GSS_C_AF_NULLADDR` as the address-types.

3.3 Optional Parameters

Various parameters are described as optional. This means that they follow a convention whereby a default value may be requested. The following conventions are used for omitted parameters. These conventions apply only to those parameters that are explicitly documented as optional.

- gss_buffer_t types. Specify GSS_C_NO_BUFFER as a value. For an input parameter this signifies that default behavior is requested, while for an output parameter it indicates that the information that would be returned via the parameter is not required by the application.
- Integer types (input). Individual parameter documentation lists values to be used to indicate default actions.
- Integer types (output). Specify NULL as the value for the pointer.
- Pointer types. Specify NULL as the value.
- Object IDs. Specify GSS_C_NO_OID as the value.
- Object ID Sets. Specify GSS_C_NO_OID_SET as the value.
- Channel Bindings. Specify GSS_C_NO_CHANNEL_BINDINGS to indicate that channel bindings are not to be used.

3.4 Error Handling

Every GSS-API routine returns two distinct values to report status information to the caller: GSS status codes and Mechanism status codes.

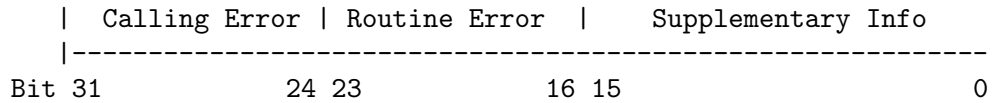
3.4.1 GSS status codes

GSS-API routines return GSS status codes as their `OM_uint32` function value. These codes indicate errors that are independent of the underlying mechanism(s) used to provide the security service. The errors that can be indicated via a GSS status code are either generic API routine errors (errors that are defined in the GSS-API specification) or calling errors (errors that are specific to these language bindings).

A GSS status code can indicate a single fatal generic API error from the routine and a single calling error. In addition, supplementary status information may be indicated via the setting of bits in the supplementary info field of a GSS status code.

These errors are encoded into the 32-bit GSS status code as follows:

MSB LSB



Hence if a GSS-API routine returns a GSS status code whose upper 16 bits contain a non-zero value, the call failed. If the calling error field is non-zero, the invoking application's call of the routine was erroneous. Calling errors are defined in table 3-1. If the routine error field is non-zero, the routine failed for one of the routine-specific reasons listed below in table 3-2. Whether or not the upper 16 bits indicate a failure or a success, the routine may indicate additional information by setting bits in the supplementary info field of the status code. The meaning of individual bits is listed below in table 3-3.

Table 3-1 Calling Errors

Name	Value in field	Meaning
-----	-----	-----
GSS_S_CALL_INACCESSIBLE_READ	1	A required input parameter could not be read
GSS_S_CALL_INACCESSIBLE_WRITE	2	A required output parameter could not be written.
GSS_S_CALL_BAD_STRUCTURE	3	A parameter was malformed

Table 3-2 Routine Errors

Name	Value in field	Meaning
-----	-----	-----
GSS_S_BAD_MECH	1	An unsupported mechanism was requested
GSS_S_BAD_NAME	2	An invalid name was supplied
GSS_S_BAD_NAME_TYPE	3	A supplied name was of an unsupported type
GSS_S_BAD_BINDINGS	4	Incorrect channel bindings were supplied
GSS_S_BAD_STATUS	5	An invalid status code was supplied
GSS_S_BAD_MIC	6	A token had an invalid MIC
GSS_S_NO_CRED	7	No credentials were supplied, or the credentials were unavailable or inaccessible.
GSS_S_NO_CONTEXT	8	No context has been established
GSS_S_DEFECTIVE_TOKEN	9	A token was invalid
GSS_S_DEFECTIVE_CREDENTIAL	10	A credential was invalid
GSS_S_CREDENTIALS_EXPIRED	11	The referenced credentials have expired
GSS_S_CONTEXT_EXPIRED	12	The context has expired

GSS_S_FAILURE	13	Miscellaneous failure (see text)
GSS_S_BAD_QOP	14	The quality-of-protection requested could not be provided
GSS_S_UNAUTHORIZED	15	The operation is forbidden by local security policy
GSS_S_UNAVAILABLE	16	The operation or option is unavailable
GSS_S_DUPLICATE_ELEMENT	17	The requested credential element already exists
GSS_S_NAME_NOT_MN	18	The provided name was not a mechanism name

Table 3-3 Supplementary Status Bits

Name ----	Bit Number -----	Meaning -----
GSS_S_CONTINUE_NEEDED	0 (LSB)	Returned only by gss_init_sec_context or gss_accept_sec_context. The routine must be called again to complete its function. See routine documentation for detailed description
GSS_S_DUPLICATE_TOKEN	1	The token was a duplicate of an earlier token
GSS_S_OLD_TOKEN	2	The token's validity period has expired
GSS_S_UNSEQ_TOKEN	3	A later token has already been processed
GSS_S_GAP_TOKEN	4	An expected per-message token was not received

The routine documentation also uses the name GSS_S_COMPLETE, which is a zero value, to indicate an absence of any API errors or supplementary information bits.

All GSS_S_XXX symbols equate to complete 0M_uint32 status codes, rather than to bitfield values. For example, the actual value of the symbol GSS_S_BAD_NAME_TYPE (value 3 in the routine error field) is $3 \ll 16$. The macros GSS_CALLING_ERROR, GSS_ROUTINE_ERROR and GSS_SUPPLEMENTARY_INFO are provided, each of which takes a GSS status code and removes all but the relevant field. For example, the value obtained by applying GSS_ROUTINE_ERROR to a status code removes the calling errors and supplementary info fields, leaving only the routine errors field. The values delivered by these macros may be directly compared with a GSS_S_XXX symbol of the appropriate type. The macro GSS_ERROR is also provided, which when applied to a GSS status code returns a non-zero value if the status code indicated a calling or routine error, and a zero value otherwise. All macros defined by GSS-API evaluate their argument(s) exactly once.

A GSS-API implementation may choose to signal calling errors in a platform-specific manner instead of, or in addition to the routine value; routine errors and supplementary info should be returned via major status values only.

The GSS major status code `GSS_S_FAILURE` is used to indicate that the underlying mechanism detected an error for which no specific GSS status code is defined. The mechanism-specific status code will provide more details about the error.

In addition to the explicit major status codes for each API function, the code `GSS_S_FAILURE` may be returned by any routine, indicating an implementation-specific or mechanism-specific error condition, further details of which are reported via the `minor_status` parameter.

3.4.2 Mechanism-specific status codes

GSS-API routines return a `minor_status` parameter, which is used to indicate specialized errors from the underlying security mechanism. This parameter may contain a single mechanism-specific error, indicated by a `OM_uint32` value.

The `minor_status` parameter will always be set by a GSS-API routine, even if it returns a calling error or one of the generic API errors indicated above as fatal, although most other output parameters may remain unset in such cases. However, output parameters that are expected to return pointers to storage allocated by a routine must always be set by the routine, even in the event of an error, although in such cases the GSS-API routine may elect to set the returned parameter value to `NULL` to indicate that no storage was actually allocated. Any length field associated with such pointers (as in a `gss_buffer_desc` structure) should also be set to zero in such cases.

3.5 Credential Management

GSS-API Credential-management Routines

Routine	Function
-----	-----
<code>gss_acquire_cred</code>	Assume a global identity; Obtain a GSS-API credential handle for pre-existing credentials.
<code>gss_add_cred</code>	Construct credentials incrementally.
<code>gss_inquire_cred</code>	Obtain information about a credential.
<code>gss_inquire_cred_by_mech</code>	Obtain per-mechanism information about a credential.
<code>gss_release_cred</code>	Discard a credential handle.

gss_acquire_cred

```
OM_uint32 gss_acquire_cred (OM_uint32 * minor_status,           [Function]
                           const gss_name_t desired_name, OM_uint32 time_req, const
                           gss_OID_set desired_mechs, gss_cred_usage_t cred_usage,
                           gss_cred_id_t * output_cred_handle, gss_OID_set *
                           actual_mechs, OM_uint32 * time_rec)
minor_status:  desired_name:  time_req:  desired_mechs:  cred_usage:  out-
put_cred_handle: actual_mechs: time_rec:
```

gss_add_cred

```
OM_uint32 gss_add_cred (OM_uint32 * minor_status, const           [Function]
                        gss_cred_id_t input_cred_handle, const gss_name_t
                        desired_name, const gss_OID desired_mech, gss_cred_usage_t
                        cred_usage, OM_uint32 initiator_time_req, OM_uint32
                        acceptor_time_req, gss_cred_id_t * output_cred_handle,
                        gss_OID_set * actual_mechs, OM_uint32 * initiator_time_rec,
                        OM_uint32 * acceptor_time_rec)
minor_status:  input_cred_handle:  desired_name:  desired_mech:  cred_usage:
initiator_time_req:  acceptor_time_req:  output_cred_handle:  actual_mechs:
initiator_time_rec: acceptor_time_rec:
```

gss_inquire_cred

```
OM_uint32 gss_inquire_cred (OM_uint32 * minor_status,           [Function]
                             const gss_cred_id_t cred_handle, gss_name_t * name,
                             OM_uint32 * lifetime, gss_cred_usage_t * cred_usage,
                             gss_OID_set * mechanisms)
minor_status: cred_handle: name: lifetime: cred_usage: mechanisms:
```

gss_inquire_cred_by_mech

```
OM_uint32 gss_inquire_cred_by_mech (OM_uint32 *                [Function]
                                     minor_status, const gss_cred_id_t cred_handle, const gss_OID
                                     mech_type, gss_name_t * name, OM_uint32 *
                                     initiator_lifetime, OM_uint32 * acceptor_lifetime,
                                     gss_cred_usage_t * cred_usage)
minor_status: cred_handle: mech_type: name: initiator_lifetime: acceptor_lifetime:
cred_usage:
```

gss_release_cred

```
OM_uint32 gss_release_cred (OM_uint32 * minor_status,           [Function]
                             gss_cred_id_t * cred_handle)
minor_status: cred_handle:
```

3.6 Context-Level Routines

GSS-API Context-Level Routines

Routine -----	Function -----
<code>gss_init_sec_context</code>	Initiate a security context with a peer application.
<code>gss_accept_sec_context</code>	Accept a security context initiated by a peer application.
<code>gss_delete_sec_context</code>	Discard a security context.
<code>gss_process_context_token</code>	Process a token on a security context from a peer application.
<code>gss_context_time</code>	Determine for how long a context will remain valid.
<code>gss_inquire_context</code>	Obtain information about a security context.
<code>gss_wrap_size_limit</code>	Determine token-size limit for <code>gss_wrap</code> on a context.
<code>gss_export_sec_context</code>	Transfer a security context to another process.
<code>gss_import_sec_context</code>	Import a transferred context.

`gss_init_sec_context`

```
OM_uint32 gss_init_sec_context (OM_uint32 * minor_status,      [Function]
                               const gss_cred_id_t initiator_cred_handle, gss_ctx_id_t *
                               context_handle, const gss_name_t target_name, const gss_OID
                               mech_type, OM_uint32 req_flags, OM_uint32 time_req, const
                               gss_channel_bindings_t input_chan_bindings, const
                               gss_buffer_t input_token, gss_OID * actual_mech_type,
                               gss_buffer_t output_token, OM_uint32 * ret_flags, OM_uint32
                               * time_rec)
minor_status: initiator_cred_handle: context_handle: target_name: mech_type:
req_flags: time_req: input_chan_bindings: input_token: actual_mech_type:
output_token: ret_flags: time_rec:
```

gss_accept_sec_context

```
OM_uint32 gss_accept_sec_context (OM_uint32 *                               [Function]
    minor_status, gss_ctx_id_t * context_handle, const
    gss_cred_id_t acceptor_cred_handle, const gss_buffer_t
    input_token_buffer, const gss_channel_bindings_t
    input_chan_bindings, gss_name_t * src_name, gss_OID *
    mech_type, gss_buffer_t output_token, OM_uint32 * ret_flags,
    OM_uint32 * time_rec, gss_cred_id_t * delegated_cred_handle)
    minor_status:    context_handle:    acceptor_cred_handle:    input_token_buffer:
    input_chan_bindings: src_name:    mech_type:    output_token:    ret_flags:    time_rec:
    delegated_cred_handle:
```

gss_delete_sec_context

```
OM_uint32 gss_delete_sec_context (OM_uint32 *                               [Function]
    minor_status, gss_ctx_id_t * context_handle, gss_buffer_t
    output_token)
    minor_status: context_handle: output_token:
```

gss_process_context_token

```
OM_uint32 gss_process_context_token (OM_uint32 *                               [Function]
    minor_status, const gss_ctx_id_t context_handle, const
    gss_buffer_t token_buffer)
    minor_status: context_handle: token_buffer:
```

gss_context_time

```
OM_uint32 gss_context_time (OM_uint32 * minor_status,                       [Function]
    const gss_ctx_id_t context_handle, OM_uint32 * time_rec)
    minor_status: context_handle: time_rec:
```

gss_inquire_context

```
OM_uint32 gss_inquire_context (OM_uint32 * minor_status,                   [Function]
    const gss_ctx_id_t context_handle, gss_name_t * src_name,
    gss_name_t * targ_name, OM_uint32 * lifetime_rec, gss_OID *
    mech_type, OM_uint32 * ctx_flags, int * locally_initiated,
    int * open)
    minor_status: context_handle: src_name: targ_name: lifetime_rec: mech_type:
    ctx_flags: locally_initiated: open:
```

gss_wrap_size_limit

```
OM_uint32 gss_wrap_size_limit (OM_uint32 * minor_status,          [Function]
                               const gss_ctx_id_t context_handle, int conf_req_flag,
                               gss_qop_t qop_req, OM_uint32 req_output_size, OM_uint32 *
                               max_input_size)
    minor_status:  context_handle:  conf_req_flag:  qop_req:  req_output_size:
    max_input_size:
```

gss_export_sec_context

```
OM_uint32 gss_export_sec_context (OM_uint32 *                    [Function]
                                   minor_status, gss_ctx_id_t * context_handle, gss_buffer_t
                                   interprocess_token)
    minor_status: context_handle: interprocess_token:
```

gss_import_sec_context

```
OM_uint32 gss_import_sec_context (OM_uint32 *                    [Function]
                                   minor_status, const gss_buffer_t interprocess_token,
                                   gss_ctx_id_t * context_handle)
    minor_status: interprocess_token: context_handle:
```

3.7 Per-Message Routines

GSS-API Per-message Routines

Routine	Function
-----	-----
<code>gss_get_mic</code>	Calculate a cryptographic message integrity code (MIC) for a message; integrity service.
<code>gss_verify_mic</code>	Check a MIC against a message; verify integrity of a received message.
<code>gss_wrap</code>	Attach a MIC to a message, and optionally encrypt the message content.
<code>gss_unwrap</code>	Verify a message with attached MIC, and decrypt message content if necessary.

gss_get_mic

```
OM_uint32 gss_get_mic (OM_uint32 * minor_status, const          [Function]
                       gss_ctx_id_t context_handle, gss_qop_t qop_req, const
                       gss_buffer_t message_buffer, gss_buffer_t message_token)
    minor_status: context_handle: qop_req: message_buffer: message_token:
```

gss_verify_mic

```
OM_uint32 gss_verify_mic (OM_uint32 * minor_status, const      [Function]
                          gss_ctx_id_t context_handle, const gss_buffer_t
                          message_buffer, const gss_buffer_t token_buffer, gss_qop_t *
                          qop_state)
    minor_status: context_handle: message_buffer: token_buffer: qop_state:
```

gss_wrap

```
OM_uint32 gss_wrap (OM_uint32 * minor_status, const          [Function]
                   gss_ctx_id_t context_handle, int conf_req_flag, gss_qop_t
                   qop_req, const gss_buffer_t input_message_buffer, int *
                   conf_state, gss_buffer_t output_message_buffer)
    minor_status: context_handle: conf_req_flag: qop_req: input_message_buffer:
    conf_state: output_message_buffer:
```

gss_unwrap

```
OM_uint32 gss_unwrap (OM_uint32 * minor_status, const      [Function]
                      gss_ctx_id_t context_handle, const gss_buffer_t
                      input_message_buffer, gss_buffer_t output_message_buffer,
                      int * conf_state, gss_qop_t * qop_state)
    minor_status: context_handle: input_message_buffer: output_message_buffer:
    conf_state: qop_state:
```

3.8 Name Manipulation

GSS-API Name manipulation Routines

Routine	Function
-----	-----
<code>gss_import_name</code>	Convert a contiguous string name to internal-form.
<code>gss_display_name</code>	Convert internal-form name to text.
<code>gss_compare_name</code>	Compare two internal-form names.
<code>gss_release_name</code>	Discard an internal-form name.
<code>gss_inquire_names_for_mech</code>	List the name-types supported by the specified mechanism.
<code>gss_inquire_mechs_for_name</code>	List mechanisms that support the specified name-type.
<code>gss_canonicalize_name</code>	Convert an internal name to an MN.
<code>gss_export_name</code>	Convert an MN to export form.
<code>gss_duplicate_name</code>	Create a copy of an internal name.

gss_import_name

```
OM_uint32 gss_import_name (OM_uint32 * minor_status, const      [Function]
                          gss_buffer_t input_name_buffer, const gss_OID
                          input_name_type, gss_name_t * output_name)
                          minor_status: input_name_buffer: input_name_type: output_name:
```

gss_display_name

```
OM_uint32 gss_display_name (OM_uint32 * minor_status,          [Function]
                          const gss_name_t input_name, gss_buffer_t
                          output_name_buffer, gss_OID * output_name_type)
                          minor_status: input_name: output_name_buffer: output_name_type:
```

gss_compare_name

```
OM_uint32 gss_compare_name (OM_uint32 * minor_status,          [Function]
                          const gss_name_t name1, const gss_name_t name2, int *
                          name_equal)
                          minor_status: name1: name2: name_equal:
```

gss_release_name

```
OM_uint32 gss_release_name (OM_uint32 * minor_status,          [Function]
                          gss_name_t * name)
                          minor_status: name:
```

gss_inquire_names_for_mech

```
OM_uint32 gss_inquire_names_for_mech (OM_uint32 *              [Function]
                          minor_status, const gss_OID mechanism, gss_OID_set *
                          name_types)
                          minor_status: mechanism: name_types:
```

gss_inquire_mechs_for_name

```
OM_uint32 gss_inquire_mechs_for_name (OM_uint32 *              [Function]
                          minor_status, const gss_name_t input_name, gss_OID_set *
                          mech_types)
                          minor_status: input_name: mech_types:
```

gss_canonicalize_name

```
OM_uint32 gss_canonicalize_name (OM_uint32 * minor_status,     [Function]
                          const gss_name_t input_name, const gss_OID mech_type,
                          gss_name_t * output_name)
                          minor_status: input_name: mech_type: output_name:
```

gss_export_name

```
OM_uint32 gss_export_name (OM_uint32 * minor_status, const      [Function]
                          gss_name_t input_name, gss_buffer_t exported_name)
                          minor_status: input_name: exported_name:
```

gss_duplicate_name

```
OM_uint32 gss_duplicate_name (OM_uint32 * minor_status,      [Function]
                              const gss_name_t src_name, gss_name_t * dest_name)
                              minor_status: src_name: dest_name:
```

3.9 Miscellaneous Routines

GSS-API Miscellaneous Routines

Routine	Function
-----	-----
<code>gss_add_oid_set_member</code>	Add an object identifier to a set.
<code>gss_display_status</code>	Convert a GSS-API status code to text.
<code>gss_indicate_mechs</code>	Determine available underlying authentication mechanisms.
<code>gss_release_buffer</code>	Discard a buffer.
<code>gss_release_oid_set</code>	Discard a set of object identifiers.
<code>gss_create_empty_oid_set</code>	Create a set containing no object identifiers.
<code>gss_test_oid_set_member</code>	Determines whether an object identifier is a member of a set.
<code>gss_encapsulate_token</code>	Encapsulate a context token.
<code>gss_decapsulate_token</code>	Decapsulate a context token.
<code>gss_oid_equal</code>	Compare two OIDs for equality.

gss_add_oid_set_member

```
OM_uint32 gss_add_oid_set_member (OM_uint32 *      [Function]
                                  minor_status, const gss_OID member_oid, gss_OID_set *
                                  oid_set)
                                  minor_status: member_oid: oid_set:
```

gss_display_status

```
OM_uint32 gss_display_status (OM_uint32 * minor_status,          [Function]
                             OM_uint32 status_value, int status_type, const gss_OID
                             mech_type, OM_uint32 * message_context, gss_buffer_t
                             status_string)
    minor_status:    status_value:    status_type:    mech_type:    message_context:
    status_string:
```

gss_indicate_mechs

```
OM_uint32 gss_indicate_mechs (OM_uint32 * minor_status,          [Function]
                              gss_OID_set * mech_set)
    minor_status: mech_set:
```

gss_release_buffer

```
OM_uint32 gss_release_buffer (OM_uint32 * minor_status,          [Function]
                              gss_buffer_t buffer)
    minor_status: buffer:
```

gss_release_oid_set

```
OM_uint32 gss_release_oid_set (OM_uint32 * minor_status,          [Function]
                               gss_OID_set * set)
    minor_status: set:
```

gss_create_empty_oid_set

```
OM_uint32 gss_create_empty_oid_set (OM_uint32 *                  [Function]
                                     minor_status, gss_OID_set * oid_set)
    minor_status: oid_set:
```

gss_test_oid_set_member

```
OM_uint32 gss_test_oid_set_member (OM_uint32 *                  [Function]
                                    minor_status, const gss_OID member, const gss_OID_set set,
                                    int * present)
    minor_status: member: set: present:
```

gss_encapsulate_token

```
extern OM_uint32 gss_encapsulate_token (gss_const_buffer_t      [Function]
                                         input_token, gss_const_OID token_oid, gss_buffer_t
                                         output_token)
    input_token: token_oid: output_token:
```

gss_decapsulate_token

```
OM_uint32 gss_decapsulate_token (gss_const_buffer_t      [Function]
    input_token, gss_const_OID token_oid, gss_buffer_t
    output_token)
    input_token: token_oid: output_token:
```

gss_oid_equal

```
int gss_oid_equal (gss_const_OID first_oid, gss_const_OID  [Function]
    second_oid)
    first_oid: second_oid:
```

3.10 SASL GS2 Routines**gss_inquire_mech_for_saslname**

```
OM_uint32 gss_inquire_mech_for_saslname (OM_uint32 *      [Function]
    minor_status, const gss_buffer_t sasl_mech_name, gss_OID *
    mech_type)
    minor_status: sasl_mech_name: mech_type:
```

gss_inquire_saslname_for_mech

```
OM_uint32 gss_inquire_saslname_for_mech (OM_uint32 *      [Function]
    minor_status, const gss_OID desired_mech, gss_buffer_t
    sasl_mech_name, gss_buffer_t mech_name, gss_buffer_t
    mech_description)
    minor_status: desired_mech: sasl_mech_name: mech_name: mech_description:
```

4 Extended GSS API

None of the following functions are standard GSS API functions. As such, they are not declared in `gss/api.h`, but rather in `gss/ext.h` (which is included from `gss.h`). See Section 2.1 [Header], page 7.

gss_check_version

```
const char * gss_check_version (const char * req_version)      [Function]
    req_version:
```

gss_userok

```
int gss_userok (const gss_name_t name, const char *          [Function]
    username)
    name: username:
```

5 Invoking gss

Name

GNU GSS (gss) – Command line interface to the GSS Library.

Description

`gss` is the main program of GNU GSS.

Mandatory or optional arguments to long options are also mandatory or optional for any corresponding short options.

Commands

`gss` recognizes these commands:

```
-l, --list-mechanisms
                        List information about supported mechanisms
                        in a human readable format.
-m, --major=LONG      Describe a `major status' error code value.
-a, --accept-sec-context
                        Accept a security context as server.
-i, --init-sec-context=MECH
                        Initialize a security context as client.
                        MECH is the SASL name of mechanism, use -l
                        to list supported mechanisms.
-n, --server-name=SERVICE@HOSTNAME
                        For -i, set the name of the remote host.
                        For example, "imap@mail.example.com".
```

Other Options

These are some standard parameters.

```
-h, --help            Print help and exit
-V, --version          Print version and exit
-q, --quiet            Silent operation (default=off)
```

Examples

To list the supported mechanisms, use `gss -l` like this:

```
$ src/gss -l
Found 1 supported mechanisms.
```

Mechanism 0:

```
    Mechanism name: Kerberos V5
    Mechanism description: Kerberos V5 GSS-API mechanism
    SASL Mechanism name: GS2-KRB5
```

\$

To initialize a Kerberos V5 security context, use the `--init-sec-context` parameter. Kerberos V5 needs to know the name of the remote entity, so you need to supply the `--server-name` parameter as well. That will provide the name of the server. For example, use `imap@mail.example.com` to setup a security context with the `imap` service on the host `mail.example.com`. The Kerberos V5 client will use your ticket-granting ticket (which needs to be available) and acquire a server ticket for the service. The KDC must know about the server for this to work. The tool will print the GSS-API context tokens base64 encoded on standard output.

```
$ gss -i GS2-KRB5 -n host@interop.josefsson.org
Context token (protection is available):
YIICIQYJKoZIhvcSAQICAQBuggIQMIICDKADAgEFoQMCAQ6iBwMFACAAAACjggEYYYIBFDCCARCgAwIBBaEXGxVpbnR.
Input context token:
```

The tool is waiting for the final Kerberos V5 context token from the server. Note the status text informing you that message protection is available.

To accept a Kerberos V5 context, the process is similar. The server needs to know its name, so that it can find the host key from (typically) `/etc/shishi/shishi.keys`. Once started it will wait for a context token from the client. Below we'll paste in the token printed above.

```
$ gss -a -n host@interop.josefsson.org
Importing name "host@interop.josefsson.org"...
Acquiring credentials...
Input context token:
YIICIQYJKoZIhvcSAQICAQBuggIQMIICDKADAgEFoQMCAQ6iBwMFACAAAACjggEYYYIBFDCCARCgAwIBBaEXGxVpbnR.
Context has been accepted. Final context token:
YHEGCSqGSib3EgECAGIAb2IwYKADAgEFoQMCAQ+iVDBSoAMCARKhAwIBAKJBESy1Zoy9DrG+DuV/6aWmAp79s9d+of
$
```

Returning to the client, you may now cut'n'paste the final context token as shown by the server. The client has then authenticated the server as well. The output from the client is shown below.

```
YHEGCSqGSib3EgECAGIAb2IwYKADAgEFoQMCAQ+iVDBSoAMCARKhAwIBAKJBESy1Zoy9DrG+DuV/6aWmAp79s9d+of
Context has been initialized.
$
```

6 Acknowledgements

This manual borrows text from RFC 2743 and RFC 2744 that describe GSS API formally.

Appendix A Criticism of GSS

The author has doubts whether GSS is the best solution for free software projects looking for a implementation agnostic security framework. We express these doubts in this section, so that the reader can judge for herself if any of the potential problems discussed here are relevant for their project, or if the benefit outweigh the problems. We are aware that some of the opinions are highly subjective, but we offer them in the hope they can serve as anecdotal evidence.

GSS can be criticized on several levels. We start with the actual implementation.

GSS does not appear to be designed by experienced C programmers. While generally this may be a good thing (C is not the best language), but since they defined the API in C, it is unfortunate. The primary evidence of this is the `major_status` and `minor_status` error code solution. It is a complicated way to describe error conditions, but what makes matters worse, the error condition is separated; half of the error condition is in the function return value and the other half is in the first argument to the function, which is always a pointer to an integer. (The pointer is not even allowed to be `NULL`, if the application doesn't care about the minor error code.) This makes the API unreadable, and difficult to use. A better solutions would be to return a struct containing the entire error condition, which can be accessed using macros, although we acknowledge that the C language used at the time GSS was designed may not have allowed this (this may in fact be the reason the awkward solution was chosen). Instead, the return value could have been passed back to callers using a pointer to a struct, accessible using various macros, and the function could have a void prototype. The fact that `minor_status` is placed first in the parameter list increases the pain it is to use the API. Important parameters should be placed first. A better place for `minor_status` (if it must be present at all) would have been last in the prototypes.

Another evidence of the C inexperience are the memory management issues; GSS provides functions to deallocate data stored within, e.g., `gss_buffer_t` but the caller is responsible of deallocating the structure pointed at by the `gss_buffer_t` (i.e., the `gss_buffer_desc`) itself. Memory management issues are error prone, and this division easily leads to memory leaks (or worse). Instead, the API should be the sole owner of all `gss_ctx_id_t`, `gss_cred_id_t`, and `gss_buffer_t` structures: they should be allocated by the library, and deallocated (using the utility functions defined for this purpose) by the library.

TBA: specification is unclear how memory for OIDs are managed. For example, who is responsible for deallocate potentially newly allocated OIDs returned as `actual_mechs` in `gss_acquire_cred`? Further, are OIDs deeply copied into OID sets? In other words, if I add an OID into an OID set, and modify the original OID, will the OID in the OID set be modified too?

Another illustrating example is the sample GSS header file given in the RFC, which contains:

```
/*
 * We have included the xom.h header file.  Verify that OM_uint32
 * is defined correctly.
 */
#if sizeof(gss_uint32) != sizeof(OM_uint32)
#error Incompatible definition of OM_uint32 from xom.h
```

`#endif`

The C pre-processor does not know about the `sizeof` function, so it is treated as an identifier, which maps to 0. Thus, the expression does not check that the size of `OM_uint32` is correct. It checks whether the expression `0 != 0` holds.

TBA: thread issues

TBA: multiple mechanisms in a GSS library

TBA: high-level design criticism.

TBA: no credential forwarding.

TBA: internationalization

TBA: dynamically generated OIDs and memory deallocation issue. I.e., should `gss_import_name` or `gss_duplicate_name` allocate memory and copy the OID provided, or simply copy the pointer? If the former, who would deallocate that memory? If the latter, the application may deallocate or modify the OID, which seem unwanted.

TBA: krb5: no way to access authorization-data

TBA: krb5: firewall/pre-IP: iakerb status?

TBA: krb5: single-DES only

TBA: the API may block, unusable in `select()` based servers. Especially if the servers contacted is decided by the, yet unauthenticated, remote client.

TBA: krb5: no support for `GSS_C_PROT_READY_FLAG`. We support it anyway, though.

TBA: krb5: `gssapi-cfx` differ from rfc 1964 in the reply token in that the latter require presence of sequence numbers whereas the former doesn't.

Finally we note that few free security applications uses GSS, perhaps the only major exception to this are Kerberos 5 implementations. While not substantial evidence, this do suggest that the GSS may not be the simplest solution available to solve actual problems, since otherwise more projects would have chosen to take advantage of the work that went into GSS instead of using another framework (or designing their own solution).

Our conclusion is that free software projects that are looking for a security framework should evaluate carefully whether GSS actually is the best solution before using it. In particular it is recommended to compare GSS with the Simple Authentication and Security Layer (SASL) framework, which in several situations provide the same feature as GSS does. The most compelling argument for SASL over GSS is, as its acronym suggest, Simple, whereas GSS is far from it.

However, that said, for free software projects that wants to support Kerberos 5, we do acknowledge that no other framework provides a more portable and interoperable interface into the Kerberos 5 system. If your project needs to use Kerberos 5 specifically, we do recommend you to use GSS instead of the Kerberos 5 implementation specific APIs.

Appendix B Copying Information

B.1 GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
<https://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at

your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <https://www.gnu.org/licenses/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Concept Index

A

AIX 3

C

command line 29
 Compiling your application 8
 Contributing 5

D

Debian 2, 3
 Download 4

F

FDL, GNU Free Documentation License 34
 FreeBSD 4
 Future goals 6

H

Hacking 5
 Header files 7
 HP-UX 3

I

Installation 4
 invoking `gss` 29
 IRIX 3

M

Mandrake 3
 mechanism status codes 15
 Memory allocation failure 8
 Motorola Coldfire 4

N

NetBSD 3

O

OpenBSD 3
 Out of Memory handling 8

R

RedHat 3
 RedHat Advanced Server 3
 Reporting Bugs 5

S

Solaris 3
 status codes 15
 SuSE 3
 SuSE Linux 3

T

Todo list 6
 Tru64 3

U

uClibc 4
 uClinux 4

W

Windows 3

API Index

<code>gss</code>	29	<code>gss_inquire_cred</code>	19
<code>gss_accept_sec_context</code>	21	<code>gss_inquire_cred_by_mech</code>	19
<code>gss_acquire_cred</code>	19	<code>gss_inquire_mech_for_saslname</code>	27
<code>gss_add_cred</code>	19	<code>gss_inquire_mechs_for_name</code>	24
<code>gss_add_oid_set_member</code>	25	<code>gss_inquire_names_for_mech</code>	24
<code>gss_canonicalize_name</code>	24	<code>gss_inquire_saslname_for_mech</code>	27
<code>gss_check_version</code>	28	<code>gss_oid_equal</code>	27
<code>gss_compare_name</code>	24	<code>gss_process_context_token</code>	21
<code>gss_context_time</code>	21	<code>gss_release_buffer</code>	26
<code>gss_create_empty_oid_set</code>	26	<code>gss_release_cred</code>	19
<code>gss_decapsulate_token</code>	27	<code>gss_release_name</code>	24
<code>gss_delete_sec_context</code>	21	<code>gss_release_oid_set</code>	26
<code>gss_display_name</code>	24	<code>gss_test_oid_set_member</code>	26
<code>gss_display_status</code>	26	<code>gss_unwrap</code>	23
<code>gss_duplicate_name</code>	25	<code>gss_userok</code>	28
<code>gss_encapsulate_token</code>	26	<code>gss_verify_mic</code>	23
<code>gss_export_name</code>	25	<code>gss_wrap</code>	23
<code>gss_export_sec_context</code>	22	<code>gss_wrap_size_limit</code>	22
<code>gss_get_mic</code>	22	<code>GSS_CALLING_ERROR</code>	17
<code>gss_import_name</code>	24	<code>GSS_ERROR</code>	17
<code>gss_import_sec_context</code>	22	<code>GSS_ROUTINE_ERROR</code>	17
<code>gss_indicate_mechs</code>	26	<code>GSS_S</code>	16
<code>gss_init_sec_context</code>	20	<code>GSS_SUPPLEMENTARY_INFO</code>	17
<code>gss_inquire_context</code>	21		