The 'KSBA' Reference Manual

The 'KSBA' Reference Manual

Edition 1.0.6

last updated 9 January 2009

for version 1.0.6

by Werner Koch, g10 Code GmbH
wk@gnupg.org

Short Contents

Table of Contents

1 Introduction

KSBA is a library to make the task of working with X.509 certificates, CMS data and related data more easy.

1.1 Getting Started

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

The reader is assumed to possess basic knowledge about the implemented protocols.

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

'KSBA' has a couple of advantages over other libraries doing a similar job, and over open coding the protocols in your application directly.

It's Free Software
: Anybody can use, modify, and redistribute it under the terms of the GNU General Public License (see [Copying], page 20).

It hides the low level stuff
: 'KSBA' a high level interface to the implemented protocols and presents the data in a consistent way. There is no more need to worry about all the nasty details of the protocols. The API gives the C programmer a more usual way of interacting with the data.

It copes with the version details
: X.509 protocols tend to have many different versions and dialects. Applications must usually cope with all of this and it has to be coded over and over again. 'KSBA' hides this by providing just one API which does the Right Thing. Support for new versions and features of the protocols will be added over time.

1.3 Overview

The 'KSBA' library is thread-safe as long as objects described by one context are only used by one thread at a time. No initialization is required.

2 Preparation

To use 'KSBA', 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.

2.1 Header

All interfaces (data types and functions) of the library are defined in the header file '`ksba.h`'. You must include this in all programs using the library, either directly or through some other header file, like this:

```
#include <ksba.h>
```

The name space of 'KSBA' is `ksba_*` for function names, `ksba*` for data types and `KSBA_*` 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.

2.2 Version Check

It is often desirable to check that the version of 'KSBA' used is indeed one which fits all requirements. Even with binary compatibility, new features may have been introduced but through peculiarities of the runtime linker an old version gets actually used. So you better check that the version is as expected right after program startup.

`const char * ksba_check_version (`*const char* *`req_version`*`)` [Function]

Check that the the version of the library is at minimum the one given as a string in *req_version* and return the actual version string of the library; return `NULL` if the condition is not met. If `NULL` is passed to this function, no check is done and only the version string is returned. It is a pretty good idea to run this function as soon as possible, because it may also initializes some subsystems. In a multi-threaded environment if should be called before any more threads are created.

2.3 Building the source

If you want to compile a source file including the '`ksba.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 compiler's 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, 'KSBA' ships with a small helper program `ksba-config` that knows about 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 of `ksba-config`. The following example shows how it can be used at the command line:

```
gcc -c foo.c `ksba-config --cflags`
```

Adding the output of '`ksba-config --cflags`' to the compiler's command line will ensure that the compiler can find the '`ksba.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`' of `ksba-config` can be used. For convenience, this option also outputs all other options that are required to link the program with the 'KSBA' libraries (in particular, the '`-lksba`' option). The example shows how to link '`foo.o`' with the 'KSBA' libraries to a program `foo`.

```
gcc -o foo foo.o `ksba-config --libs`
```

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

```
gcc -o foo foo.c `ksba-config --cflags --libs`
```

3 How to work with X.509 certificates.

One of the most complex data formats are the X.509 certificates. KSBA provides an easy to use interface to handle them.

`ksba_cert_t` [Data type]
: The `ksba_cert_t` type is a handle for an X.509 certificate.

`ksba_sexp_t` [Data type]
: The `ksba_sexp_t` type describes a canonically encoded S-expression stored in a memory buffer. It is alias for `unsigned char *`. Note that a length argument is not required because the length of such an S-expression is intrinsically available.

3.1 How to create a certificate object

This section explains how to create a certificate object, initialize it, copy it and eventually destroy it.

`ksba_cert_t ksba_cert_new` (*void*) [Function]
: The function `ksba_cert_new` creates a new certificate object and returns a handle for it. The certificate object has initially one reference.

  The only reason why this function may fail is an out-of-memory condition in which case `NULL` is returned. You might then get the actual error code using '`gpg_error_from_errno (errno)`'.

`void ksba_cert_ref` (*ksba_cert_t* `cert`) [Function]
: The function `ksba_cert_ref` bumps the reference counter of the certificate object up by one. Thus an extra `ksba_cert_release` is required to actually release the memory used for the object.

`void ksba_cert_release` (*ksba_cert_t* `cert`) [Function]
: The function `ksba_cert_release` reduces the number of references to the certificate object with the handle *cert*. If this was the last reference, it will also destroy the object and releases all associated resources. It is okay to pass `NULL` to the function in which case nothing happens.

`gpg_error_t ksba_cert_read_der` (*ksba_cert_t* `cert`, *ksba_reader_t* `reader`) [Function]
: Read the next certificate from the *reader* object and store it in the certificate object *cert* for future access. The certificate is parsed and rejected if it has any syntactical or semantical error (i.e. does not match the ASN.1 description).

  The function returns `0` if the operation was successfully performed. An error code is returned on failure.

`gpg_error_t ksba_cert_init_from_mem` (*ksba_cert_t* `cert`, *const void* *`buffer`, *size_t* `length`) [Function]
: Parse the *buffer* which should contain a DER encoded certificate of *length* and initialize the certificate object *cert* with it. This function is intended as a convenience function to be used when a certificate is already available in a internal memory buffer.

This avoids the extra code needed to setup the reader object. Note that *cert* must be a valid certificate object.

The function returns `0` if the operation was successfully performed. An error code is returned on failure.

3.2 How to get the attributes of a certificate

The functions in this section allow accessing the attributes of a certificate in a well defined manner. An error will be returned if the certificate object has not yet been initialized by means of `ksba_cert_read_der` or `ksba_cert_init_from_mem`.

`const unsigned char * ksba_cert_get_image (`*ksba_cert_t* **`cert`**`,` *size_t* ***`r_length`**`)` [Function]

This function returns a pointer to the DER encoded buffer with the raw certificate. The length of that buffer gets stored at *r_length*. This function is useful to export or store the raw certificate.

The function returns `NULL` on error or a pointer to a buffer with the raw certificate data. That pointer is only valid as long as the certificate object *cert* is valid and has not been reinitialized.

`gpg_error_t ksba_cert_hash (`*ksba_cert_t* **`cert`**`,` *int* **`what`**`,` *void* `(`***`hasher`**`)(`*void *, const void *, size_t length*`),` *void* ***`hasher_arg`**`)` [Function]

This function feeds the data which is expected to be hashed into the supplied function *hasher*, where the first argument passed is *hasher_arg*, the second the pointer to the data to be hashed and the third the length of this data.

The function returns `0` on success or an error code when something goes wrong. The *hasher* function is not expected to return an error; instead the caller should setup that function in a way to convey encountered errors by means of the *hasher_arg*. Note that a hash function is in general not expected to yield errors anyway.

`const char * ksba_cert_get_digest_algo (`*ksba_cert_t* **`cert`**`)` [Function]

Figure out the the digest algorithm used for the signature and return its OID in dotted decimal format. This function is most likely used to setup the hash context before calling `ksba_cert_hash`.

The function returns `NULL` for an error; on success a constant string with the OID is returned. This string is valid as long the certificate object is valid.

`ksba_sexp_t ksba_cert_get_serial (`*ksba_cert_t* **`cert`**`)` [Function]

The function returns the serial number of the certificate *cert*. The serial number is an integer returned as an canonical encoded S-expression with just one element. The caller must free the returned value. The value `NULL` is returned in case of error.

`char * ksba_cert_get_issuer (`*ksba_cert_t* **`cert`**`,` *int* **`idx`**`)` [Function]

With *idx* given as `0`, this function returns the Distinguished Name (DN) of the certificate issuer; this usually is the name of a certification authority (CA). The format of the returned string is in accordance with RFC-2253. `NULL` is returned if the DN is not available; This is a severe error and actually should have been caught by the certificate reading function.

With *idx* greater than zero, the function may be used to enumerate alternate issuer names. The function returns `NULL` when there are no more alternate names. Only alternate names recognized by `libksba` are returned, others are simply skipped. The format of the returned name is either a RFC-2253 formated string which can be detected by checking whether the first character is a letter or digit. RFC-822 conformant email addresses are returned enclosed in angle brackets; the opening angle bracket should be used to detect this. Other formats are returned as an S-Expression in canonical format, so a opening parenthesis should be used to detect this encoding. The name may include binary null characters, thus strlen may return a length shorter than actually used. The real length is implicitly given by the structure of the S-expression, an extra null is appended for safety reasons.

The caller must free the returned string using `ksba_free` or whatever function has been registered as a replacement.

`char * ksba_cert_get_subject (`*ksba_cert_t* `cert`, *int* `idx``)` [Function]

With *idx* given as `0`, this function returns the Distinguished Name (DN) of the certificate's subject. The format of the returned string is in accordance with RFC-2253. `NULL` is returned if the DN is not available.

With *idx* greater than zero, the function may be used to enumerate alternate subject names. The function returns `NULL` when there are no more alternate names. Only alternate names recognized by `libksba` are returned, others are simply skipped. The format of the returned name is either a RFC-2253 formated string which can be detected by checking whether the first character is a letter or digit. RFC-2822 conform email addresses are returned enclosed in angle brackets; the opening angle bracket should be used to detect this. Other formats are returned as an S-Expression in canonical format, so a opening parenthesis should be used to detect this encoding, the name may include binary null characters, thus strlen may return a length shorter than actually used. The real length is implicitly given by the structure of the S-expression, an extra null is appended for safety reasons.

The caller must free the returned string using `ksba_free` or whatever function has been registered as a replacement.

`ksba_isotime_t` [Data type]

Due to problems with the C data type `time_t`, which will overflow on most 32 bit machines in the year 2038, it was not advisable to use this type for referencing times stored in certificates. Instead, you should use the `ksba_isotime_t` type, which can represent any time since the year 0.

It is implemented as a buffer of 16 bytes and may be handled like a standard string. It should be initialized to zero (i.e. the first byte needs to be 0x00) if it does not hold a valid date. Date values themselves are stored in ISO format and assumed to be referenced from UTC. The string with the date value is always guaranteed to be of length 15 and having a format like: '`"19610711T172059"`'. Note that the 'T' is required by ISO rules.

A simple assignment of these data types is not a good idea. You may use `strcpy` or better a specialized function like:

```
void
```

```
copy_time (ksba_isotime_t d, const ksba_isotime_t s)
{
  if (!*s)
    memset (d, 0, 16);
  else
    strcpy (d, s);
}
```

For reasons of documentation a special function should also be used to compare such times:

```
int
cmp_time (const ksba_isotime_t a, const ksba_isotime_t b)
{
  return strcmp (a, b);
}
```

`gpg_error_t ksba_cert_get_validity` [Function]
(*ksba_cert_t* **cert**, *int* **what**, *ksba_isotime_t* **timebuf**)

Return the validity dates from the certificate. If no value is available an empty date object (i.e. a `strlen` will be stored at *timebuf*, otherwise it will receive the date. On failure an error code is returned.

To return the 'notBefore' date, the value `0` must be supplied for *what*; `1` yields the 'notAfter' value.

`ksba_sexp_t ksba_cert_get_public_key (`*ksba_cert_t* **cert**`)` [Function]

[This needs to get written - for now please see libksba/src/cert.c]

`ksba_sexp_t ksba_cert_get_sig_val (`*ksba_cert_t* **cert**`)` [Function]

[This needs to get written - for now please see libksba/src/cert.c]

`gpg_error_t ksba_cert_get_extension` [Function]
(*ksba_cert_t* **cert**, *int* **idx**, *char const* ***r_oid**, *int* ***r_crit**, *size_t* ***r_deroff**, *size_t* ***r_derle**

[This needs to get written - for now please see libksba/src/cert.c]

`gpg_error_t ksba_cert_is_ca` [Function]
(*ksba_cert_t* **cert**, *int* ***r_ca**, *int* ***r_pathlen**)

Return information on the basicConstraint (2.5.19.19) of CERT. R_CA receives true if this is a CA and only in that case R_PATHLEN is set to the maximum certification path length or -1 if there is no such limitation

`gpg_error_t ksba_cert_get_key_usage` [Function]
(*ksba_cert_t* **cert**, *unsigned int* ***r_flags**)

Get the key usage flags. The function returns `GPG_ERR_NO_DATA` if no key usage is specified. The usage flags are as shown in RFC3280, section 4.2.1.3. The key usage flags are represented by a bitmask, and you can test each bit using symbolic constants, which tells you if that usage is set on the certificate. The constants are

`KSBA_KEYUSAGE_DIGITAL_SIGNATURE`
: Usable for digitalSignature.

`KSBA_KEYUSAGE_NON_REPUDIATION`
: Usable for nonRepudiation.

`KSBA_KEYUSAGE_KEY_ENCIPHERMENT`
: Usable for keyEncipherment.

`KSBA_KEYUSAGE_DATA_ENCIPHERMENT`
: Usable for dataEncipherment.

`KSBA_KEYUSAGE_KEY_AGREEMENT`
: Usable for for keyAgreement.

`KSBA_KEYUSAGE_KEY_CERT_SIGN`
: Usable for keyCertSign.

`KSBA_KEYUSAGE_CRL_SIGN`
: Usable for cRLSign.

`KSBA_KEYUSAGE_ENCIPHER_ONLY`
: Usable for encipherOnly.

`KSBA_KEYUSAGE_DECIPHER_ONLY`
: Usable for decipherOnly.

These are the basic constraints on usage of a certificate. If you need to get additional constraints, see `ksba_cert_get_ext_key_usages`.

`gpg_error_t ksba_cert_get_ext_key_usages` [Function]
(*ksba_cert_t* **cert**, *char* ***result**)

Return a string containing the extended usages for the certificate, delimited by linefeeds.

`gpg_error_t ksba_cert_get_cert_policies` [Function]
(*ksba_cert_t* **cert**, *char* ***r_policies**)

Return a string with the certificatePolicies delimited by linefeeds. The return values may be extended to carry more information per line, so the caller should only use the first white-space delimited token per line. The function returns `GPG_ERR_NO_DATA` when this extension is not used. Caller must free the returned value.

`gpg_error_t ksba_cert_get_crl_dist_point` [Function]
(*ksba_cert_t* **cert**, *int* **idx**, *ksba_name_t* ***r_distpoint**, *ksba_name_t* ***r_issuer**, *unsigned int*

Return the CRLDistPoints given in the certificate extension of certificate *cert*. *idx* should be iterated starting from 0 until the function returns `GPG_ERR_EOF`. *r_distpoint* returns a ksba_name_t object with the distribution point name(s); the return value may be `NULL` to indicate that this name is not available. *r_issuer* returns the CRL issuer; if the returned value is `NULL` the caller should assume that the CRL issuer is the same as the certificate issuer. *r_reason* returns the reason for the CRL. This is a bit encoded value with no bit set if no reason has been specified in the certificate.

The caller may pass `NULL` to any of the pointer arguments if he is not interested in this value. The return values for *r_distpoint* and *r_issuer* must be released by the caller using `ksba_name_release`.

`gpg_error_t ksba_cert_get_subj_key_id` [Function]
(*ksba_cert_t* **cert**, *int* ***r_crit**, *ksba_sexp_t* ***r_keyid**)
Return the subjectKeyIdentifier extension as a simple allocated S-expression at the address of *r_keyid*. 0 is returned on success, `GPG_ERR_NO_DATA` if no such extension is available or any other error code. If *r_crit* is not passed as `NULL`, the critical flag of this is extension is stored at this address.

`gpg_error_t ksba_cert_get_auth_key_id` [Function]
(*ksba_cert_t* **cert**, *ksba_sexp_t* ***r_keyid**, *ksba_name_t* ***r_name**, *ksba_sexp_t* ***r_serial**)
Return the authorityKeyIdentifier in *r_name* and *r_serial* or in *r_keyid*. `GPG_ERR_NO_DATA` is returned if no authorityKeyIdentifier has been found. This error code is also returned if *r_keyid* has been given as NULL and only an authorityKeyIdentifier with the keyIdentifier method is available.

`gpg_error_t ksba_cert_get_authority_info_access` [Function]
(*ksba_cert_t* **cert**, *int* **idx**, *char* ****r_method**, *ksba_name_t* ***r_location**)
Return the authorityInfoAccess attributes. *idx* should be iterated starting from 0 until this function returns `GPG_ERR_EOF`. *r_method* returns an allocated string with the OID of one item and *r_location* returns the GeneralName for that OID. The returned values for *r_method* and *r_location* must be released by the caller unless the function returned an error; the function will however make sure that *r_method* and *r_location* will point to `NULL` if the function returns an error.

See RFC-2459, section 4.2.2.1 for the definition of this attribute.

`gpg_error_t ksba_cert_get_subject_info_access` [Function]
(*ksba_cert_t* **cert**, *int* **idx**, *char* ****r_method**, *ksba_name_t* ***r_location**)
Return the subjectInfoAccess attributes. *idx* should be iterated starting from 0 until this function returns `GPG_ERR_EOF`. *r_method* returns an allocated string with the OID of one item and *r_location* returns the GeneralName for that OID. The returned values for *r_method* and *r_location* must be released by the caller unless the function returned an error; the function will however make sure that *r_method* and *r_location* will point to `NULL` if the function returns an error.

See RFC-2459, section 4.2.2.2 for the definition of this attribute.

3.3 How to set certificate attributes

[This needs to be written. For example code see newpg/sm/sign.c]

3.4 How to associate other data with a certificate.

Certificate objects play a central role in many applications and often it is desirable to associate other data with the certificate to avoid wrapping the certificate object into an own object. 'KSBA' provides a mechanism for this by means of two functions:

`gpg_error_t ksba_cert_set_user_data` [Function]
(*ksba_cert_t* **cert**, *const char* ***key**, *const void* ***data**, *size_t* **datalen**)
Stores arbitrary data along with a certificate. The data is expected in the buffer *data* of length *datalen*. It will be stored under the string *key*. If data is already stored

under this key it will be replaced by the new data. Using `NULL` for *data* will effectively delete the data.

On error (i.e. out of memory) an already existing data object stored under *key* may get deleted.

Caution: This function is definitely not thread safe because we don't employ any locking mechanisms.

`gpg_error_t ksba_cert_get_user_data` (*ksba_cert_t* **`cert`**, *const char* ***`key`**, *void* ***`buffer`**, *size_t* **`bufferlen`**, *size_t* ***`datalen`**) [Function]

Return user data for certificate *cert* stored under the string *key*. The caller needs to provide a suitable large *buffer* and the usable length of this buffer in *bufferlen*. If *datalen* is not `NULL`, the length of the data stored in *buffer* will be stored there.

If *buffer* is given as `NULL`, *bufferlen* will be ignored and the required length of the buffer will be returned at *datalen*.

On success 0 is returned. If no data is stored under the given key, `GPG_ERR_NOT_FOUND` is returned. If the provided buffer is too short and *buffer* is not `NULL`, `GPG_ERR_BUFFER_TOO_SHORT` will be returned.

4 Mastering the Cryptographic Message Syntax

The CMS is also known under the name PKCS#7. Is is a cryptographic framework for securing data transactions and storage, much like OpenPGP. It is heavily based on X.509 semantics and for example used with the email encryption protocol S/MIME.

4.1 CMS Basics

All operations with the CMS framework require the use of a so called CMS object which is internally used to keep track of the current state and to store some meta information.

`ksba_cms_t` [Data type]

The `ksba_cms_t` type is used for this CMS object.

`ksba_stop_reason_t` [Data type]

The `ksba_stop_reason_t` type is an enumeration used for communication between the phases of a parsing or building process.

`ksba_cms_t ksba_cms_new` (*void*) [Function]

This function creates a new CMS object. The only reason the function may fail is an out-of-memory condition in which case `NULL` is returned. It is safe for the caller to translate this to the standard error code `GPG_ERR_ENOMEM`. Any object created with this function should be released after use by using `ksba_cms_release`.

`void ksba_cms_release` (*ksba_cms_t* `cms`) [Function]

Release all resources associated with the *CMS* object. It is perfectly okay to pass `NULL` to this function in which case nothing happens.

`gpg_error_t ksba_cms_set_reader_writer` [Function]
(*ksba_cms_t* `cms`, *ksba_reader_t* `r`, *ksba_writer_t* `w`)

About all usages of the CMS framework require some input and output data (great surprise!). To accomplish this in the most abstract way, no direct output functions are used - instead special reader and writer objects are used instead. Depending on the desired operations either a reader, a writer or both must be given. Associate a reader object with *cms* by passing it as *r* and a writer object by passing it as *w*. Note that no reference counting is done,so make sure that those objects have a lifetime at least as long as *CMS*.

If you forget to set these objects, you will get an appropriate error later when data is actually to be read or written. The function returns zero on success or an error code when invalid objects are passed.

4.2 CMS Parser

KSBA includes a versatile CMS parser for encryption (enveloped data) and digital signing. The parser is capable of handling arbitrary amounts of data without requiring much memory. Well, certain objects are build in memory because it can be assumed that those objects are limited in size; e.g. it does not make sense to use a video clip as the DN despite the fact that the standard does not forbid it.

`gpg_error_t ksba_cms_parse` [Function]
(*ksba_cms_t* **cms**, *ksba_stop_reason_t* ***r_stopreason**)

This is the core function of the parser and commonly used in a loop. The parsing process is divided into several phases to allow the user to get information at the right time and prepare for further processing. The caller has to act on certain stop reasons which are returned by *r_stopreason* and set up things accordingly; KSBA may introduce new stop reasons to let the caller know other details; there is no need for the caller to act on every stop reason; it should only do so for reasons that the caller understands and which are mandatory. The function will return with an error if the caller did not setup things correctly for certain stop reasons.

The use of this function is best explained by an example, leaving out all error checking.

```
do
  {
    ksba_cms_parse (cms, &stopreason);
    if (stopreason == KSBA_SR_BEGIN_DATA)
      {
        get_recipients ();
        decrypt_session_key ();
        setup_bulk_decryption ();
      }
    else if (stopreason == KSBA_SR_END_DATA)
      {
        remove_padding ();
      }
  }
while (stopreason != KSBA_SR_READY);
```

This function assumes that the parsed data is so called 'enveloped data'.

As CMS provides a common framework for a variety of data formats, it is probably very useful to check the type of that data very early. This can be accomplished by hooking into the stop reason `KSBA_SR_GOT_CONTENT` and retrieving the content using the following function.

`ksba_content_t ksba_cms_get_content_type` [Function]
(*ksba_cms_t* **cms**, *int* **what**)

By using a value of `0` for *what* this function returns the content type of the outer container; using `1` does return the content type of the enclosed object.

`ksba_content_t` [Data type]

The `ksba_content_t` type is an enumeration used to describe the content of a CMS message. Here is a list of possible values:

`KSBA_CT_NONE`
No content type known (value `0`)

`KSBA_CT_DATA`
The content is plain data, not further interpreted.

`KSBA_CT_SIGNED_DATA`
: The content is an signed CMS object. This also includes the case of a detached signature where no actual data is included in the message.

`KSBA_CT_ENVELOPED_DATA`
: The content is encrypted using a session key.

`KSBA_CT_DIGESTED_DATA`
: Not yet supported

`KSBA_CT_ENCRYPTED_DATA`
: Not yet supported

`KSBA_CT_AUTH_DATA`
: Not yet supported

`const char * ksba_cms_get_content_oid` [Function]
(*ksba_cms_t* **`cms`**, *int* **`what`**)

Return the object ID of *cms*. This is a constant string valid as long as the context is valid and no new parse is started. This function is similar to `ksba_cms_get_content_type` but returns the OID actually used in the data. Depending on the value of *what* different values are returned: Using a value of `0` yields the OID of the outer container, a value of `1` yields the OID of the inner container if available and the value `2` returns the OID of the algorithm used to encrypt the inner container.

5 Certification Revocation Lists

KSBA also comes with an API to process certification revocation lists. The API is similar to the CMS one but returns the contents entry by entry.

6 Certification Requests

When using decentral generated keys, it is necessary to send out special formated messages so that a CA can generate the certificate.

7 Utilities

A few utility function and objects are available. Some of them must be used to support some of the main functions.

7.1 General Names object

This is an object to handle some of the names used in X.509. We need this object approach because those names may come as a set and there is no other clean way to access them.

`ksba_name_t` [Data type]

The `ksba_name_t` type is an object to represent names sets.

`void ksba_name_release (`*ksba_name_t* ***name***`)` [Function]

This function releases the object *name*. Passing `NULL` is allowed.

`const char * ksba_name_enum (`*ksba_name_t* ***name***`,` *int* ***idx***`)` [Function]

By iterating *idx* up starting with 0, this function returns all General Names stored in *name*. The format of the returned name is either a RFC-2253 formated one which can be detected by checking whether the first character is letter or a digit. RFC 2822 conformant email addresses are returned enclosed in angle brackets, the opening angle bracket should be used to detect this. Other formats are returned as an S-Expression in canonical format, so an opening parenthesis may be used to detect this encoding, in this case the name may include binary null characters, so strlen might return a length shorter than actually used, the real length is implicitly given by the structure of the S-Exp, an extra null is appended for safety reasons. One common format return is a Universal Resource Identifier which has the S-expression: '`(uri <urivalue>)`'.

The returned string has the same lifetime as *name*.

`char * ksba_name_get_uri (`*ksba_name_t* ***name***`,` *int* ***idx***`)` [Function]

Convenience function to return names representing an URI. Caller must free the returned value. Note that this function should not be used to enumerate the names.

Here is an example on how you can use this function to enumerate all URIs:

```
void
print_names (ksba_name_t name)
{
  int idx;
  const char *s;

  for (idx=0; (s = ksba_name_enum (name, idx)); idx++)
    {
      char *p = ksba_name_get_uri (name, idx);
      if (p)
        {
           puts (p);
           ksba_free (p);
        }
    }
}
```

7.2 Object Identifier helpers

[This needs to get written - for now please see libksba/src/oids.c]

7.3 Distinguished Name helpers

These are helper functions for the so called distinguished names. They are used for example as the issuer and subject name.

`gpg_error_t ksba_dn_teststr` (*const char* ***string***, *int* ***seq***, *size_t* ***rerroff***, *size_t* ***rerrlen***) [Function]

Assuming that *string* contains an RFC-2253 encoded string, test whether this string may be passed as a valid DN to libksba. On success the functions returns `0`. On error the function returns an error code and stores the offset of the erroneous part at *rerroff*. *rerrlen* will then receive the length of the erroneous part.

This function is mostly useful to test whether a certain component label is supported. *seq* should be passed as `0` for now. Any of *rerroff* and *rerrlen* may be passed as *NULL* if the caller is not interested at this value.

gpg_error_t ksba_dn_str2der (const char *string, void **rder, size_t *rderlen);

gpg_error_t ksba_dn_der2str (const void *der, size_t derlen, char **r_string);

8 Error Handling

Most functions in 'KSBA' will return an error if they fail. For this reason, the application should always catch the error condition and take appropriate measures, for example by releasing the resources and passing the error up to the caller, or by displaying a descriptive message to the user and canceling the operation.

Some error values do not indicate a system error or an error in the operation, but the reasonable result of an operation. For example, if you try to access optional attributes of a certificate that are not present, you get an appropriate error message. Some error values have specific meanings if returned by a specific function. Such cases are described in the documentation of those functions.

All error codes are defined by the library `libgpg-error`. See there for ways to check the error values and print descriptive strings. Please be aware that you can't check directly against an error code but have to do it like this:

```
err = ksba_foo ();
if (gpg_err_code (err) == GPG_ERR_EOF)
  okay = 1;
```

The only exception is that success (i.e. no error) is defined to be `0`; thus you may directly test for success like:

```
if (!ksba_foo ())
  okay = 1;
```

Appendix A Component Labels

RFC-2253 defines the following table with string representations of name components:

Label	Component	OID
C	countryName	2.5.4.6
CN	commonName	2.5.4.3
DC	domainComponent	0.9.2342.19200300.100.1.25
L	localityName	2.5.4.7
O	organizationName	2.5.4.10
OU	organizationalUnit	2.5.4.11
ST	stateOrProvince	2.5.4.8
STREET	streetAddress	2.5.4.9
UID	userid	0.9.2342.19200300.100.1.1

They are used internally for converting a DN into its string representation; components not listed in this table will be represented by their OID.

For the other direction, i.e. creating a DN from the string representation, KSBA recognizes the following extra labels:

Label	Component	OID
ADDR	postalAddress	2.5.4.16
BC	businessCategory	2.5.4.15
D	description	2.5.4.13
EMAIL	emailAddress	1.2.840.113549.1.9.1
GN	givenName	2.5.4.42
POSTALCODE	postalCode	2.5.4.17
PSEUDO	pseudonym	2.5.4.65
SERIALNUMBER	serialNumber	2.5.4.5
SN	surname	2.5.4.4
T	title	2.5.4.12

GNU General Public License

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

   b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source.

The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

   a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
   b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
   c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   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, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance.

However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so

available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public 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.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> *one line to give the program's name and a brief idea of what it does.*
> Copyright (C) *year name of author*
>
> This program is free software: you can redistribute it and/or modify
> it under the terms of the GNU General Public License as published by
> the Free Software Foundation, either version 3 of the License, or (at
> your option) any later version.
>
> This program is distributed in the hope that it will be useful, but
> WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
> General Public License for more details.
>
> You should have received a copy of the GNU General Public License
> along with this program. If not, see http://www.gnu.org/licenses/.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

> *program* Copyright (C) *year name of author*
> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

Concept Index

(Index is nonexistent)

Function and Data Index