Developing with Assuan

Version 2.5.3
7 December 2017

by Werner Koch and Marcus Brinkmann
{wk,mb}@g10code.com

Copyright © 2001–2013 Free Software Foundation, Inc.
Copyright © 2001–2015 g10 Code GmbH

Permission is granted to copy, distribute and/or modify this document 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. The text of the license can be found in the section entitled "Copying".

Short Contents

Table of Contents

1 Introduction to Assuan

Assuan is an extensible inter-process communication (IPC) protocol and library. It is designed for point-to-point communication and it doesn't provide a naming system. To contact a server, either the client must know how to locate the server, e.g., via a well-known Unix domain socket, or, if the server is transient, how to start it. In the latter case, Assuan provides functionality to start the server process.

In Assuan, communication is typically either via a pipe or a Unix domain socket. This method is neither elegant nor efficient, especially when there is a lot of data spread across several transactions. Not only is there a penalty for an increased number of context switches, but a significant amount of data is *memcpy*ed from the client to a file descriptor and from the file descriptor to the server. Despite these and other disadvantages, this type of client/server communication is useful: the client is separated from the server: they run in different address spaces. This is especially important in situations where the server must have a known degree of reliability and data must be protected: as the Assuan protocol is well defined and clients cannot corrupt the servers' address space, auditing becomes much easier.

Assuan was developed for use by the GNU Privacy Guard (GnuPG) to prevent potentially buggy clients from unwittingly corrupting sensitive transactions or compromising data such as a secret key. Assuan permits the servers, which do the actual work, e.g., encryption and decryption of data using a secret key, to be developed independently of the user interfaces, e.g., mail clients and other encryption front ends. Like a shared library, the interface is well defined and any number of front ends can use it; however, unlike a shared library, the client cannot see or touch the server's data. As with any modular system, Assuan helps keep the components small, understandable and less error prone.

Assuan is not, however, limited to use with GnuPG servers and clients: it was designed to be flexible enough to meet the demands of many transaction-based environments.

2 Description of the Assuan protocol.

The architecture of the modular GnuPG system is based on several highly specialized modules which form a network of clients and servers. A common framework for intermodule communication is therefore needed and implemented as a library.

Goals:

- Common framework for module communication
- Easy debugging
- Easy module testing
- Extensible
- Optional authentication and encryption facility
- Usable to access external hardware

Design criteria:

- Client/Server with back channel
- Use a mainly text based protocol
- Escape certain control characters
- Allow indefinite data length
- Request confidentiality for parts of the communication
- Dummy module to allow direct linking of client and server
- Inline data or descriptor passing for bulk data
- No protection against DoS needed
- Subliminal channels are not an issue

3 Implementation

The implementation is line based with a maximum line size of 1000 octets. The default IPC mechanism is Unix Domain Sockets.

On connect, the server responds either with okay or an error status. To perform an authentication check, the server may send an Inquiry response prior to the first Okay. It may also issue Status messages. The server must check that the client is allowed to connect. This is done by requesting the credentials for the peer and comparing them with the server's credentials. This avoids attacks based on wrong socket permissions.

The server may choose to delay the first response in case of an error. The server, however, never closes the connection, however, the lower protocol may do so after some time of inactivity or when the connection enters an error state.

All textual messages are assumed to be in UTF-8 unless otherwise noted.

3.1 Server responses

`OK [<arbitrary debugging information>]`

Request was successful.

`ERR` *`errorcode`* `[<human readable error description>]`

Request could not be fulfilled. The possible error codes are defined by `libgpg-error`.

`S` *`keyword`* `<status information depending on keyword>`

Informational output by the server, which is still processing the request. A client may not send such lines to the server while processing an Inquiry command. *keyword* shall start with a letter or an underscore.

`# <string>`

Comment line issued only for debugging purposes. Totally ignored.

`D <raw data>`

Raw data returned to client. There must be exactly one space after the 'D'. The values for '%', CR and LF must be percent escaped; these are encoded as %25, %0D and %0A, respectively. Only uppercase letters should be used in the hexadecimal representation. Other characters may be percent escaped for easier debugging. All Data lines are considered one data stream up to the OK or ERR response. Status and Inquiry Responses may be mixed with the Data lines.

`INQUIRE` *`keyword`* `<parameters>`

The server needs further information from the client. The client should respond with data (using the "D" command and terminated by "END"). Alternatively, the client may cancel the current operation by responding with "CAN".

Consider the following examples (lines prefixed with S indicate text that the server sends; lines prefixed with C indicate text that the client sends):

```
S: INQUIRE foo
C: D foo bar
```

```
C: D bar baz
C: END
[Server continues normal work]
```

This implements a callback to the client:

```
S: INQUIRE foo
C: END
[Server continues]
```

and:

```
S: INQUIRE foo
C: CAN
[Server terminates the operaion and in most cases returns an ERR to the client.]
```

But, CAN may also mean "I have no data for you, try to get it from elsewhere."

Note: lines longer than 1000 bytes should be treated as a communication error. (The rationale for having a line length limit is to allow for easier multiplexing of several channels.)

3.2 Client requests

The server waits for client requests after sending an Okay or Error. The client should not issue a request in other cases.

```
command <parameters>
```

command is a one word string without preceding white space. Parameters are command specific, CR, LF and the percent signs should be percent escaped as described above. To send a backslash as the last character it should also be percent escaped. Percent escaping is allowed anywhere in the parameters but not in the command. The line ends with a CR, LF pair or just a LF.

Not yet implemented feature: If there is a need for a parameter list longer than the line length limit (1000 characters including command and CR, LF), the last character of the line (right before the CR/LF or LF) must be a unescaped (i.e., literal) backslash. The following line is then expected to be a continuation of the line with the backslash replaced by a blank and the line ending removed.

```
D <raw data>
```

Sends raw data to the server. There must be exactly one space after the 'D'. The values for '%', CR and LF must be percent escaped. These are encoded as %25, %0D and %0A, respectively. Only uppercase letters should be used in the hexadecimal representation. Other characters may be percent escaped for easier debugging. All Data lines are considered one data stream up to the `OK` or `ERR` response. Status and Inquiry Responses may be mixed with the Data lines.

```
END
```

Lines beginning with a `#` or empty lines are ignored. This is useful to comment test scripts.

Although the commands are application specific, some of them are used by all protocols and partly supported by the Assuan library:

`BYE` Close the connection. The server will respond with `OK`.

`RESET` Reset the connection but not any existing authentication. The server should release all resources associated with the connection.

`END` Used by a client to mark the end of raw data. The server may send `END` to indicate a partial end of data.

`HELP` Lists all commands that the server understands as comment lines on the status channel.

`QUIT` Reserved for future extensions.

`OPTION` Set options for the connection. The syntax of such a line is

OPTION *name* [[=] *value*]

Leading and trailing spaces around *name* and *value* are allowed but should be ignored. For compatibility reasons, *name* may be prefixed with two dashes. The use of the equal sign is optional but suggested if *value* is given.

`CANCEL` This command is reserved for future extensions.

`AUTH` This command is reserved for future extensions. Not yet specified as we don't implement it in the first phase. See Werner's mail to gpa-dev on 2001-10-25 about the rationale for measurements against local attacks.

`NOP` No operation. Returns OK without any action.

3.3 Error codes

Libassuan is used with gpg-error style error codes. It is recommended to set the error source to a different value from the default `GPG_ERR_SOURCE_UNKNOWN` by calling [function assuan_set_gpg_err_source], page 12 early.

4 Preparation

To use Assuan, you have to make some changes to your sources and the build system. The necessary changes are small and explained in the following sections.

4.1 Header

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

```
#include <assuan.h>
```

The namespace of `libassuan` is `assuan_*` for function and type names and `ASSUAN*` 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.

Because `libassuan` makes use of the GPG Error library, using `libassuan` will also use the `GPG_ERR_*` namespace directly, and the `gpg_err*` and `gpg_str*` namespaces indirectly.

4.2 Building sources

If you want to compile a source file including the `assuan.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, `libassuan` ships with a small helper program `libassuan-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 `libassuan-config`. The following example shows how it can be used at the command line:

```
gcc -c foo.c $(libassuan-config --cflags)
```

Adding the output of '`libassuan-config --cflags`' to the compiler's command line will ensure that the compiler can find the `assuan.h` header file.

A similar problem occurs when linking the program with the library. Again, the compiler/linker 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 `libassuan-config` can be used. For convenience, this option also outputs all other options that are required to link the program with the `libassuan` libraries (in particular, the `-lassuan` option). The example shows how to link `foo.o` with the `libassuan` library to a program `foo`.

```
gcc -o foo foo.o $(libassuan-config --libs)
```

You can also combine both examples to a single command by specifying both options to `libassuan-config`:

```
gcc -o foo foo.c $(libassuan-config --cflags --libs)
```

4.3 Building sources using Automake

It is much easier if you use GNU Automake instead of writing your own Makefiles. If you do that you do not have to worry about finding and invoking the `libassuan-config` script at all. `libassuan` provides an Automake macro that does all the work for you.

`AM_PATH_LIBASSUAN` ([*minimum-version*], [*action-if-found*], [*action-if-not-found*]) [Macro]

Check whether `libassuan` (at least version *minimum-version*, if given) exists on the host system. If it is found, execute *action-if-found*, otherwise do *action-if-not-found*, if given.

Additionally, the function defines `LIBASSUAN_CFLAGS` to the flags needed for compilation of the program to find the `assuan.h` header file, and `LIBASSUAN_LIBS` to the linker flags needed to link the program to the `libassuan` library.

You can use the defined Autoconf variables like this in your `Makefile.am`:

```
AM_CPPFLAGS = $(LIBASSUAN_CFLAGS)
LDADD = $(LIBASSUAN_LIBS)
```

4.4 Multi Threading

The `libassuan` library is designed so that it can be used in a threaded application, if some rules are followed.

- Run the initialization functions before you actually start to use threads. Specifically, the functions `assuan_set_gpg_err_source`, `assuan_set_malloc_hooks` and `assuan_set_log_cb` should not be called concurrently with `assuan_new`. Use `assuan_new_ext` instead or ensure proper serialization.
- Only one thread at a time may access an `libassuan` context.
- If you use the default log handler, use `assuan_set_assuan_log_stream` to setup a default log stream.
- If you have callback functions shared by multiple functions, the callback function must be reentrant for that purpose. `libassuan` does not serialize invocation of callback functions across contexts.

5 Generalities

5.1 Data Types used by the library

Assuan uses a so-called context to store a connection's state. The following data type is used for that:

`assuan_context_t` [Data type]

The `assuan_context_t` type is a pointer to an object maintained internally by the library. Contexts are allocated with `assuan_new` or `assuan_new_ext` and released with `assuan_release`. Other functions take this data type to access the state created by these functions.

`assuan_fd_t` [Data type]

The `assuan_fd_t` is a file descriptor (in Unix) or a system handle (in Windows). The special value `ASSUAN_INVALID_FD` is used to specify invalid Assuan file descriptors.

`assuan_fd_t assuan_fdopen` (*int* `fd`) [Function]

Create an assuan file descriptor from a POSIX (libc) file descriptor *fd*. On Unix, this is equivalent to `dup`, while on Windows this will retrieve the underlying system handle with `_get_osfhandle` and duplicate that.

5.2 Initializing the library

Libassuan makes use of Libgpg-error and assumes that Libgpg-error has been initialized. In general `gpgrt_check_version` should be called to guarantee this; the Libgpg-error manual for details.

Libassuan itself requires no initialization. There are however some initialization hooks provided which are often useful. These should be called as early as possible and in a multi-threaded application before a second thread is created.

These functions initialize default values that are used at context creation with `assuan_new`. As there can only be one default, all values can also be set directly with `assuan_new_ext` or with context-specific functions after context creation.

If your application uses its own memory allocation functions or wrappers it is good idea to tell `libassuan` about it so it can make use of the same functions or wrappers:

`struct assuan_malloc_hooks` [Data type]

This structure is used to store the memory allocation callback interface functions. It has the following members, whose semantics are identical to the corresponding system functions:

`void *(*malloc) (size_t cnt)`

This is the function called by Assuan to allocate memory for a context.

`void *(*realloc) (void *ptr, size_t cnt)`

This is the function called by Assuan to reallocate memory for a context.

`void (*free) (void *ptr)`

This is the function called by Assuan to release memory for a context.

`assuan_malloc_hooks_t` [Data type]

This is a pointer to a `struct assuan_malloc_hooks`.

`void assuan_set_malloc_hooks` (*assuan_malloc_hooks_t* `malloc_hooks`) [Function]

This function sets the default allocation hooks for new contexts allocated with `assuan_new`. You need to provide all three functions. Those functions need to behave exactly as their standard counterparts `malloc`, `realloc` and `free`. If you write your own functions, please take care to set `errno` whenever an error has occurred.

`assuan_malloc_hooks_t assuan_get_malloc_hooks` () [Function]

This function gets the default allocation hooks for new contexts allocated with `assuan_new`. The result structure is statically allocated and should not be modified.

The Assuan library uses `libgpg-error` error values, which consist and error code and an error source. The default source used by contexts allocated with `assuan_new` can be set with the following function.

`void assuan_set_gpg_err_source` (*gpg_err_source_t* `err_source`) [Function]

This function sets the default error source for errors generated by contexts allocated with `assuan_new`.

One way to call this function is

```
assuan_set_gpg_err_source (GPG_ERR_SOURCE_DEFAULT);
```

`gpg_err_source_t assuan_get_gpg_err_source` (*void*) [Function]

This function gets the default error source for errors generated by contexts allocated with `assuan_new`.

To integrate assuan logging and diagnostics into your own logging system, you may use the following two functions:

`int (*assuan_log_cb_t) (assuan_context_t ctx, void *hook_value, unsigned int cat, const char *msg)` [Data type]

The user-provided callback function takes a context *ctx*, for which the message *msg* was generated, and a hook value *hook_value* that was supplied when the log handler was registered for the context with `assuan_set_log_cb`, and a category *cat*. The category is one of:

`ASSUAN_LOG_INIT`
`ASSUAN_LOG_CTX`
`ASSUAN_LOG_ENGINE`
`ASSUAN_LOG_DATA`
: RFU

`ASSUAN_LOG_SYSIO`
: Log lowlevel I/O data.

`ASSUAN_LOG_CONTROL`
: Log the control channel.

The user may then, depending on the category, write the message to a log file or treat it in some other way.

If *msg* is a null pointer, then no message should be logged, but the function should return 1 if it is interested in log messages with the category *cat*. If it is not interested, 0 should be returned. This allows `libassuan` to suppress the generation of expensive debug output.

`void assuan_set_log_cb` (*assuan_log_cb_t* `log_cb`, *void* *`log_cb_data`) [Function]

This function sets the default logging handler for log messages generated by contexts allocated with `assuan_new`.

`void assuan_get_log_cb` (*assuan_log_cb_t* *`log_cb`, *void* **`log_cb_data`) [Function]

This function gets the default logging handler for log messages generated by contexts allocated with `assuan_new`.

You do not need to set a log handler, as Assuan provides a configurable default log handler that should be suitable for most purposes. Logging can be disabled completely by setting the log handler to a null pointer.

5.3 Default Log Handler

The default log handler can be configured by the following functions:

`void assuan_set_assuan_log_prefix` (*const char* *`text`) [Function]

Set the prefix to be used at the start of a line emitted by assuan on the log stream to *text*. The default is the empty string.

`const char * assuan_get_assuan_log_prefix` (*void*) [Function]

Return the prefix to be used at the start of a line emitted by assuan on the log stream. The default implementation returns the empty string.

`void assuan_set_assuan_log_stream` (*FILE* *`fp`) [Function]

This sets the default log stream to which `libassuan` should log messages not associated with a specific context to *fp*. The default is to log to `stderr`. This default value is also changed by using `assuan_set_log_stream` (to set a logging stream for a specific context) unless this function has been used. Obviously this is not thread-safe and thus it is highly recommended to use this function to setup a proper default.

`FILE * assuan_get_assuan_log_stream` (*void*) [Function]

Return the stream which is currently being using for global logging.

The log stream used by the default log handler can also be set on a per context basis.

`void assuan_set_log_stream` (*assuan_context_t* `ctx`, *FILE* *`fp`) [Function]

Enable debugging for the context *ctx* and write all debugging output to the stdio stream *fp*. If the default log stream (used for non-context specific events) has not yet been set, a call to this functions implicitly sets this stream also to *fp*.

5.4 How to work with contexts

Some operations work globally on the library, but most operate in a context, which saves state across operations. To allow the use of `libassuan` in mixed environments, such as in a library using GPGME and an application using GPGME, the context is very extensive and covers utilitary information like memory allocation callbacks as well as specific information associated with client/server operations.

`gpg_error_t assuan_new` (*assuan_context_t* ***ctx_p***) [Function]

The function `assuan_new` creates a new context, using the global default memory allocation, log handler and `libgpg-error` source. It is equivalent to

```
gpg_error_t err;
assuan_log_cb_t log_cb;
void *log_cb_data;

assuan_get_log_cb (&log_cb, &log_cb_data);
err = assuan_new_ext (ctx_p, assuan_get_gpg_err_source (),
                      assuan_get_malloc_hooks (), log_cb, log_cb_data);
```

As you can see, this is not thread-safe. Take care not to modify the memory allocation hooks or log callback handler concurrently with `assuan_new`.

The function returns an error if a memory allocation error occurs, and 0 with the new context in *ctx_p* otherwise.

`gpg_error_t assuan_new_ext` (*assuan_context_t* ***ctx_p***, *gpg_err_source_t* **err_source**, *assuan_malloc_hooks_t* **malloc_hooks**, *assuan_log_cb_t* **log_cb**, *void* ***log_cb_data***) [Function]

The function `assuan_new_ext` creates a new context using the supplied `libgpg-error` error source *err_source*, the memory allocation hooks *malloc_hooks* and the log handler *log_cb* with the user data *log_cb_data*.

After the context has been used, it can be destroyed again.

`void assuan_release` (*assuan_context_t ctx*) [Function]

The function `assuan_release` destroys the context CTX and releases all associated resources.

Other properties of the context beside the memory allocation handler, the log handler, and the `libgpg-error` source can be set after context creation. Here are some of them:

`void assuan_set_pointer` (*assuan_context_t* **ctx**, *void* ***pointer***) [Function]

Store the arbitrary pointer value *pointer* into the context *ctx*. This is useful to provide command handlers with additional application context.

`void* assuan_get_pointer` (*assuan_context_t* **ctx**) [Function]

This returns the pointer for context *ctx* which has been set using the above function. A common way to use it is by setting the pointer before starting the processing loop and to retrieve it right at the start of a command handler:

```
static int
cmd_foo (assuan_context_t ctx, char *line)
{
  ctrl_t ctrl = assuan_get_pointer (ctx);
  ...
}
```

`void assuan_set_flag` (*assuan_context_t* **`ctx`**, *assuan_flag_t* **`flag`**, *int* **`value`**) [Function]

Set the the *flag* for context *ctx* to *value*. Values for flags are usually 1 or 0 but certain flags might need other values.

`assuan_flag_t` [Data type]

The flags are all named and collected in an `enum` for better readability. Available flags are:

`ASSUAN_NO_WAITPID`

When using a pipe server, by default Libassuan will wait for the forked process to die in `assuan_release`. In certain cases this is not desirable. By setting this flag, a call to `waitpid` will be suppressed and the caller is responsible to cleanup the child process.

`ASSUAN_CONFIDENTIAL`

Use to return the state of the confidential logging mode.

`ASSUAN_NO_FIXSIGNALS`

Do not modify signal handler for `SIGPIPE`.

`ASSUAN_CONVEY_COMMENTS`

If enabled comment lines are passed to the status callback of the `assuan_transact`.

`ASSUAN_FORCE_CLOSE`

Setting this flag forces the next command to assume that the connection has been closed. This breaks the command processing loop and may be used as an implicit BYE command. *value* is ignored and thus it is not possible to clear this flag.

`int assuan_get_flag` (*assuan_context_t* **`ctx`**, *assuan_flag_t* **`flag`**) [Function]

Return the value of *flag* in context *ctx*.

`void assuan_begin_confidential` (*assuan_context_t* **`ctx`**) [Function]

Put the logging feature into confidential mode. This is to avoid logging of sensitive data.

This is identical to:

```
assuan_set_flag (ctx, ASSUAN_CONFIDENTIAL, 1);
```

`void assuan_end_confidential` (*assuan_context_t* **`ctx`**) [Function]

Get the logging feature out of confidential mode. All data will be logged again (if logging is enabled).

This is identical to:

```
assuan_set_flag (ctx, ASSUAN_CONFIDENTIAL, 0);
```

`struct assuan_system_hooks` [Data type]

This structure is used to store the system callback interface functions. It has the following members, whose semantics are similar to the corresponding system functions, but not exactly equivalent.

`int version`

The user should set this to `ASSUAN_SYSTEM_HOOKS_VERSION`. This indicates to the library which members of this structure are present in case of future extensions. The user should initialize the whole structure with zero bytes.

`void (*usleep) (assuan_context_t ctx, unsigned int usec)`

This is the function called by Assuan to sleep for `USEC` microseconds.

`int (*pipe) (assuan_context_t ctx, assuan_fd_t fd[2], int inherit_idx)`

This is the function called by Assuan to create a pipe. The returned file descriptor `fd[inherit_idx]` must be inheritable by the child process (under Windows, this requires some extra work).

`int (*close) (assuan_context_t ctx, assuan_fd_t fd)`

This is the function called by Assuan to close a file descriptor created through the system functions.

`ssize_t (*read) (assuan_context_t ctx, assuan_fd_t fd, void *buffer, size_t size)`

This is the function called by Assuan to read data from a file descriptor. It is functionally equivalent to the system `read` function.

`ssize_t (*write) (assuan_context_t ctx, assuan_fd_t fd, const void *buffer, size_t size)`

This is the function called by Assuan to write data to a file descriptor. It is functionally equivalent to the system `write` function.

`int (*recvmsg) (assuan_context_t ctx, assuan_fd_t fd, assuan_msghdr_t msg, int flags)`

This is the function called by Assuan to receive a message from a file descriptor. It is functionally equivalent to the system `recvmsg` function.

`int (*sendmsg) (assuan_context_t ctx, assuan_fd_t fd, const assuan_msghdr_t msg, int flags);`

This is the function called by Assuan to send a message to a file descriptor. It is functionally equivalent to the system `sendmsg` function.

`int (*spawn) (assuan_context_t ctx, pid_t *r_pid, const char *name, const char **argv, assuan_fd_t fd_in, assuan_fd_t fd_out, assuan_fd_t *fd_child_list, void (*atfork) (void *opaque, int reserved), void *atforkvalue, unsigned int flags)`

This is the function called by Assuan to spawn a child process. The `stdin` and `stdout` file descriptors are provided in `fd_in` and `fd_out` respectively, but can be set to `ASSUAN_INVALID_FD`, in which case they are set to `/dev/null`. On systems which use `fork` and `exec`, the `atfork` function should be called with `atforkvalue` and `0` for flags in the child process right after `fork` returns. `fd_child_list` is a `ASSUAN_INVALID_FD` terminated array (or `NULL`) and specifies file descriptors to be inherited by the child process.

A special situation occurs if `name` is a null pointer, in which case the process should just fork but not call `exec`. In this case, `*argv` should be set to `"client"` in the parent process and `"server"` in the child process.

Flags is the bit-wise OR of some (or none) of the following flags:

`ASSUAN_SPAWN_DETACHED`
: If set and there is a need to start the server it will be started as a background process. This flag is useful under W32 systems, so that no new console is created and pops up a console window when starting the server. On W32CE systems this flag is ignored.

`pid_t (*waitpid) (assuan_context_t ctx, pid_t pid, int action, int *status, int options)`
: This is the function called by Assuan to wait for the spawned child process *pid* to exit, or, if *action* is 1, to just release all resources associated with *pid* (required on Windows platforms). If *action* is 0, this is equivalent to `waitpid`.

`int (*socketpair) (assuan_context_t ctx, int namespace, int style, int protocol, assuan_fd_t filedes[2])`
: This is the function called by Assuan to create a socketpair. It is equivalent to `socketpair`.

`void assuan_set_system_hooks` (*assuan_system_hooks_t* `system_hooks`) [Function]

Set the default system hooks to use. There is currently no way to reset to the default system hooks.

`void assuan_sock_set_system_hooks` (*assuan_system_hooks_t* `system_hooks`) [Function]

The socket subsystem uses an internal context which uses the default system hooks. This function allows to change these system hooks. The function is not thread-safe and only useful if a certain order of assuan and assuan socket initializations are required.

`void assuan_ctx_set_system_hooks` (*assuan_context_t* `ctx`, *assuan_system_hooks_t* `system_hooks`) [Function]

Set the system hooks for context *ctx*. There is currently no way to reset to the default system hooks, create a new context for that.

The following system hook collections are defined by the library for your convenience:

`ASSUAN_SYSTEM_NPTH`
: System hooks suitable for use with the nPth library.

`ASSUAN_SYSTEM_NPTH_IMPL`
: The implementation of system hooks for use with the nPth library. This must be invoked once somewhere in the application, and defines the structure that is referenced by `ASSUAN_SYSTEM_NPTH`.

`ASSUAN_SYSTEM_PTH`

System hooks suitable for use with the GNU Pth library.

`ASSUAN_SYSTEM_PTH_IMPL`

The implementation of system hooks for use with the GNU Pth library. This must be invoked once somewhere in the application, and defines the structure that is referenced by `ASSUAN_SYSTEM_PTH`.

5.5 How to communicate with the peer

What would be an IPC library without the ability to read and write data? Not very useful. Libassuan has high level functions to take care of of the more boring stuff, but eventually data needs to be written and read.

The basic read and write functions are:

`gpg_error_t` **`assuan_read_line`** (*assuan_context_t* `ctx`, *char ***`line`, *size_t **`linelen`) [Function]

Read the next line written by the peer to the control channel and store a pointer to the buffer holding that line at the address *line*. The valid length of the lines is stored at the address of *linelen*. This buffer is valid until the next read operation on the same context *ctx*. You may modify the context of this buffer. The buffer is invalid (i.e. must not be used) if an error is returned. This function returns `0` on success or an error value.

`gpg_error_t` **`assuan_write_line`** (*assuan_context_t* `ctx`, *const char **`line`) [Function]

Write the string *line* to the other end on the control channel. This string needs to be a proper formatted Assuan protocol line and should not include a linefeed. Sending linefeed or `Nul` characters is not possible and not allowed by the assuan protocol. This function shall not be used for sending data (`D`) lines. This function returns `0` on success or an error value.

To actually send bulk data lines a specialized function is available:

`gpg_error_t` **`assuan_send_data`** (*assuan_context_t* `ctx`, *const void **`buffer`, *size_t* `length`) [Function]

This function is used by a server or a client to send *length* bytes of bulk data in *buffer* to the other end on the control channel. The data will be escaped as required by the Assuan protocol and may get buffered until a line is full. To flush any pending data, *buffer* may be passed as `NULL` and *length* be `0`.

When used by a client, this flush operation does also send the `END` command to terminate the response on an `INQUIRE` request. Note that the function `assuan_transact` takes care of sending this `END` itself.

This function returns `0` on success or an error value.

The input and output of data can be controlled at a higher level using an I/O monitor.

`unsigned int (*assuan_io_monitor_t) (assuan_context_t ctx, void *hook_value, int inout, const char *line, size_t linelen)` [Data type]

The monitor function is called right after a line has been received, if *inout* is `ASSUAN_IO_FROM_PEER`, or just before it is send, if *inout* is `ASSUAN_IO_TO_PEER`. The *hook_value* is provided by the user when registering the I/O monitor function with a context using `assuan_set_io_monitor`. The callback function should return the bitwise OR of some (or none) of the following flags:

`ASSUAN_IO_MONITOR_NOLOG`
: Active logging of this line is suppressed. This can reduce debug output in the case of a frequent message.

`ASSUAN_IO_MONITOR_IGNORE`
: The whole output line is discarded.

`void assuan_set_io_monitor` (*assuan_context_t* `ctx`, *assuan_io_monitor_t* `io_monitor`, *void* `*hook_data`) [Function]

This function registers an I/O monitor *io_monitor* for the context *ctx* with the hook value *hook_data*.

6 How to develop an Assuan client

Depending on the type of the server you want to connect you need to use different functions.

If the peer is not a simple pipe server but one using full-duplex sockets, the full-fledged variant of the above function should be used:

`gpg_error_t assuan_pipe_connect` [Function]
(*assuan_context_t* **`ctx`**,*const char ****`name`**, *const char ****`argv[]`**, *assuan_fd_t ****`fd_child_list`**, *void* (*****`atfork`**) (*void **, *int*), *void ****`atforkvalue`**, *unsigned int* **`flags`**)

A call to this functions forks the current process and executes the program *name*, passing the arguments given in the NULL-terminated list *argv*. A list of file descriptors not to be closed may be given using the `ASSUAN_INVALID_FD` terminated array *fd_child_list*.

If *name* is a null pointer, only a fork but no exec is done. Thus the child continues to run. However all file descriptors are closed and some special environment variables are set. To let the caller detect whether the child or the parent continues, the parent returns with `"client"` returned in *argv* and the child returns with `"server"` in *argv*. This feature is only available on POSIX platforms.

If *atfork* is not NULL, this function is called in the child right after the fork and the value *atforkvalue* is passed as the first argument. That function should only act if the second argument it received is `0`. Such a fork callback is useful to release additional resources not to be used by the child.

flags is a bit vector and controls how the function acts:

`ASSUAN_PIPE_CONNECT_FDPASSING`
: If cleared a simple pipe based server is expected. If set a server based on full-duplex pipes is expected. Such pipes are usually created using the `socketpair` function. It also enables features only available with such servers.

`ASSUAN_PIPE_CONNECT_DETACHED`
: If set and there is a need to start the server it will be started as a background process. This flag is useful under W32 systems, so that no new console is created and pops up a console window when starting the server. On W32CE systems this flag is ignored.

If you are using a long running server listening either on a TCP or a Unix domain socket, the following function is used to connect to the server:

`gpg_error_t assuan_socket_connect` (*assuan_context_t* **`ctx`**, [Function]
*const char ****`name`**, *pid_t* **`server_pid`**, *unsigned int* **`flags`**)

Make a connection to the Unix domain socket *name* using the already-initialized Assuan context at *ctx*. *server_pid* is currently not used but may become handy in the future; if you don't know the server's process ID (PID), pass `ASSUAN_INVALID_PID`. With *flags* set to `ASSUAN_SOCKET_CONNECT_FDPASSING`, `sendmsg` and `recvmesg` are used for input and output and thereby enable the use of descriptor passing.

Connecting to a TCP server is not yet implemented. Standard URL schemes are reserved for *name* specifying a TCP server.

Now that we have a connection to the server, all work may be conveniently done using a couple of callbacks and the transact function:

`gpg_error_t` **`assuan_transact`** (*assuan_context_t* ***ctx***, *const char **`command`, *gpg_error_t* (***`data_cb`)(*void **, *const void **, *size_t*), *void **`data_cb_arg`, *gpg_error_t* (***`inquire_cb`)(*void**, *const char **), *void **`inquire_cb_arg`, *gpg_error_t* (***`status_cb`)(*void**, *const char **), *void **`status_cb_arg`) [Function]

Here *ctx* is the Assuan context opened by one of the connect calls. *command* is the actual Assuan command string. It shall not end with a line feed and its length is limited to `ASSUAN_LINELENGTH` (~1000 bytes)

data_cb is called by Libassuan for data lines; *data_cb_arg* is passed to it along with the data and the length. [FIXME: needs more documentation].

inquire_cb is called by Libassuan when the server requests additional information from the client while processing the command. This callback shall check the provided inquiry name and send the data as requested back using the `assuan_send_data`. The server passed *inquiry_cb_arg* along with the inquiry name to the callback.

status_cb is called by Libassuan for each status line it receives from the server. *status_cb_arg* is passed along with the status line to the callback.

The function returns `0` success or an error value. The error value may be the one one returned by the server in error lines or one generated by the callback functions.

Libassuan supports descriptor passing on some platforms. The next two functions are used with this feature:

`gpg_error_t` **`assuan_sendfd`** (*assuan_context_t* `ctx`, *assuan_fd_t* `fd`) [Function]

Send the descriptor *fd* to the peer using the context *ctx*. The descriptor must be sent before the command is issued that makes use of the descriptor.

Note that calling this function with a *ctx* of `NULL` and *fd* of `ASSUAN_INVALID_FD` can be used as a runtime test to check whether descriptor passing is available on the platform: `0` is returned if descriptor passing is available, otherwise an error with the error code `GPG_ERR_NOT_IMPLEMENTED` is returned.

`gpg_error_t` **`assuan_receivefd`** (*assuan_context_t* `ctx`, *assuan_fd_t **`fd`) [Function]

Receive a descriptor pending for the context *ctx* from the peer. The descriptor must be pending before this function is called. To accomplish this, the peer needs to use `assuan_sendfd` before the trigger is sent (e.g. using `assuan_write_line ("INPUT FD")`.

7 How to develop an Assuan server

Implementing a server for Assuan is a bit more complex than a client. However, it is a straightforward task we are going to explain using a commented example.

The list of the implemented server commands is defined by a table like:

```
static struct {
  const char *name;
  int (*handler) (assuan_context_t, char *line);
} command_table[] = {
  { "FOO", cmd_foo },
  { "BAR", cmd_bar },
  { "INPUT", NULL },
  { "OUTPUT", NULL },
  { NULL }};
```

For convenience this table is usually put after the actual command handlers (`cmd_foo`, `cmd_bar`) or even put inside `command_handler` (see below). Note that the commands `INPUT` and `OUTPUT` do not require a handler because Libassuan provides a default handler for them. It is however possible to assign a custom handler.

A prerequisite for this example code is that a client has already connected to the server. Often there are two modes combined in one program: A pipe-based server, where a client has forked the server process, or a Unix domain socket based server that is listening on the socket.

```
void
command_handler (int fd)
{
  gpg_error_t rc;
  int i;
  assuan_context_t ctx;

  rc = assuan_new (&ctx);
  if (rc)
    {
      fprintf (stderr, "server context creation failed: %s\n",
               gpg_strerror(rc));
      return;
    }

  if (fd == -1)
    {
      assuan_fd_t filedes[2];

      filedes[0] = assuan_fd_from_posix_fd (0);
      filedes[1] = assuan_fd_from_posix_fd (1);
      rc = assuan_init_pipe_server (ctx, filedes);
    }
  else
    rc = assuan_init_socket_server (ctx, fd, ASSUAN_SOCKET_SERVER_ACCEPTED);
```

```
      if (rc)
        {
          fprintf (stderr, "server init failed: %s\n", gpg_strerror (rc));
          return;
        }
```

This is the first part of the command handler. We start off by allocating a new Assuan context with `assuan_new`. See [function assuan_new], page 14.

In case this is called as a pipe based server, *fd* will be based as *fd* and the code assumes that the server's `stdin` and `stdout` file handles are connected to a pipe. The initialization is thus done using the function:

`gpg_error_t assuan_init_pipe_server` (*assuan_context_t* ***ctx***, *assuan_fd_t* ***filedes***[*2*]) [Function]

> This function takes the two file descriptors from *filedes* and returns a new Assuan context at *r_ctx*. As usual, a return value of `0` indicates success and a failure is indicated by returning an error value. In case of error, `NULL` will be stored at *r_ctx*.
>
> In case the server has been called using a bi-directional pipe (socketpair), *filedes* is ignored and the file descriptor is taken from the environment variable `_assuan_connection_fd`. You generally don't need to know this, because `assuan_pipe_connect`, which is called by the client to connect to such a server, automagically sets this variable.

`gpg_error_t assuan_init_socket_server` (*assuan_context_t* ***ctx***, *assuan_fd_t* ***fd***, *unsigned int* ***flags***) [Function]

> This function takes the file descriptor *fd*, which is expected to be associated with a socket, and an Assuan context *ctx*. The following bits are currently defined for *flags*:
>
> `ASSUAN_SOCKET_SERVER_FDPASSING`
> If set, `sendmsg` and `recvmesg` are used for input and output, which enables the use of descriptor passing.
>
> `ASSUAN_SOCKET_SERVER_ACCEPTED`
> If set, *fd* refers to an already accepted socket. That is, Libassuan won't call *accept* for it. It is suggested to set this bit as it allows better control of the connection state.
>
> As usual, a return value of `0` indicates success and a failure is indicated by returning an error value.

On the Windows platform the following function needs to be called after `assuan_init_socket_server`:

`void assuan_set_sock_nonce` (*assuan_context_t* ***ctx***, *assuan_sock_nonce_t* ****nonce***) [Function]

> Save a copy of *nonce* in context *ctx*. This should be used to register the server's nonce with a context established by `assuan_init_socket_server`. It is technically only needed for Windows, but it does no harm to use it on other systems.

After error checking, the implemented assuan commands are registered with the server.

```
  for (i = 0; command_table[i].name; i++)
    {
      rc = assuan_register_command (ctx,
                                    command_table[i].name,
                                    command_table[i].handler, NULL);
      if (rc)
        {
          fprintf (stderr, "register failed: %s\n", gpg_strerror (rc));
          assuan_release (ctx);
          return;
        }
    }
```

`gpg_error_t (*assuan_handler_t) (assuan_context_t` ***ctx***`,` **`char *`*****line***`)` [Data type]

This is the function invoked by Assuan for various command related callback functions. Some of these callback functions have a different type, but most use `assuan_handler_t`.

`gpg_error_t assuan_register_command` (*assuan_context_t* ***ctx***, *const char **`cmd_string`, *assuan_handler_t* `handler`, *const char **`help_string`) [Function]

This registers the command named *cmd_string* with the Assuan context *ctx*. *handler* is the function called by Libassuan if this command is received from the client. *NULL* may be used for *handler* to use a default handler (this only works with a few predefined commands). Note that several default handlers have already been registered when the context has been created: `NOP`, `CANCEL`, `OPTION`, `BYE`, `AUTH`, `RESET` and `END`. It is possible, but not recommended, to override these commands.

help_string is a help string that is used for automatic documentation. It should contain a usage line followed by an empty line and a complete description.

`gpg_error_t assuan_register_post_cmd_notify` (*assuan_context_t* `ctx`, *void* (***`fnc`)(*assuan_context_t*), *gpg_error_t* `err`) [Function]

Register a function to be called right after a command has been processed. *err* is the result code from the last internal assuan operation and not the one returned by the handler. It may be used for command-related cleanup.

`gpg_error_t assuan_register_bye_notify` (*assuan_context_t* `ctx`, *assuan_handler_t* `handler`) [Function]

Register function *fnc* with context *ctx* to be called right before the standard handler for the `BYE` command is being called.

`gpg_error_t assuan_register_reset_notify` (*assuan_context_t* `ctx`, *assuan_handler_t* `handler`) [Function]

Register function *fnc* with context *ctx* to be called right before the standard handler for the `RESET` command is being called.

`gpg_error_t assuan_register_cancel_notify` [Function]
(*assuan_context_t* `ctx`, *assuan_handler_t* `handler`)

Register function *fnc* with context *ctx* to be called right before the standard handler for the `RESET` command is being called.

`gpg_error_t assuan_register_option_handler` [Function]
(*assuan_context_t* `ctx`,
gpg_error_t (*`fnc`)(*assuan_context_t, const char*, const char**))

Register function *fnc* with context *ctx* for processing options. That function is being called with the context, the name and the value of the option. Leading and trailing spaces are removed from the name and the value. The optional leading two dashes of the name are removed as well. If no value has been given, an empty string is passed. The function needs to return `0` on success or an error code.

`gpg_error_t assuan_register_input_notify` [Function]
(*assuan_context_t* `ctx`, *assuan_handler_t* `handler`)

Although the input function may be overridden with a custom handler, it is often more convenient to use the default handler and to know whether an `INPUT` command has been seen and successfully parsed. The second argument passed to that function is the entire line. Because that line has already been parsed when the function gets called, a file descriptor set with the `INPUT` command may already be used. That file descriptor is available by calling `assuan_get_input_fd`. If the notification function returns an error, the input fd does not change.

`gpg_error_t assuan_register_output_notify` [Function]
(*assuan_context_t* `ctx`, *assuan_handler_t* `handler`)

Although the output function may be overridden with a custom handler, it is often more convenient to use the default handler and to know whether an `OUTPUT` command has been seen and successfully parsed. The second argument passed to that function is the entire line. Because that line has already been parsed when the function gets called, a file descriptor set with the `OUTPUT` command may already be used. That file descriptor is available by calling `assuan_get_output_fd`. If the notification function returns an error, the output fd does not change.

`gpg_error_t assuan_set_hello_line` (*assuan_context_t* `ctx`, [Function]
const char *`line`)

This is not actually a register function but may be called also after registering commands. It changes the “Hello” line, sent by the server to the client as a first response, from a default string to the string *line*. For logging purposes, it is often useful to use such a custom hello line which may tell version numbers and such. Linefeeds are allowed in this string, however, each line needs to be shorter than the Assuan line length limit.

Now that everything has been setup, we can start to process our clients requests.

```
for (;;)
  {
    rc = assuan_accept (ctx);
    if (rc == -1)
```

```
        break;
      else if (rc)
        {
          fprintf (stderr, "accept problem: %s\n", gpg_strerror (rc));
          break;
        }

      rc = assuan_process (ctx);
      if (rc)
        {
          fprintf (stderr, "processing failed: %s\n", gpg_strerror (rc));
          continue;
        }
    }
  assuan_release (ctx);
}
```

For future extensibility and to properly detect the end of the connection the core of the server should loop over the accept and process calls.

`gpg_error_t assuan_accept` (*assuan_context_t* `ctx`) [Function]

A call to this function cancel any existing connection and waits for a connection from a client (that might be skipped, depending on the type of the server). The initial handshake is performed which may include an initial authentication or encryption negotiation. On success `0` is returned. An error value will be returned if the connection could for some reason not be established. An error code of `GPG_ERR_EOF` indicates the end of the connection.

`gpg_error_t assuan_process` (*assuan_context_t* `ctx`) [Function]

This function is used to handle the Assuan protocol after a connection has been established using `assuan_accept`. It is the main protocol handler responsible for reading the client commands and calling the appropriate handlers. The function returns `0` on success or an error value if something went seriously wrong. Error values from the individual command handlers, i.e. operational error, are not seen here.

That is all needed for the server code. You only need to come up with the code for the individual command handlers. Take care that the line passed to the command handlers is allocated statically within the context and calls to Assuan functions may modify that line. You are also allowed to modify that line which makes parsing much easier.

8 How to use external I/O event loops

The above implementations of an Assuan client and server are synchronous, insofar as the main routines block until a request or client connection is completely processed. In some programs, for example GUI applications, this is undesirable. Instead, Assuan operations should be non-blocking, and the caller should be able to poll all involved file descriptors to determine when the next Assuan function can be invoked without blocking.

To make this possible, client and server have to adhere to some rules:

- Either partner should always write full lines. If partial lines are written, the remainder of the line should be sent without delay.
- Either partner should eagerly receive status messages. While receiving and sending bulk data may be delayed, the status communication channel is different: Both partners may send status messages in blocking mode at any time the protocol allows them to send such status messages. To ensure that these send operations do not actually block the sender, the recipient must be ready to receive without undue delay.
- If descriptor passing is used over a socket, the descriptor must be sent after the corresponding command without undue delay.

Together, these restrictions allow to limit the need for asynchronous I/O operations to bulk data and the inbound status file descriptor.

In addition to the above rules, client and server should adhere to the following implementation guidelines.

8.1 External I/O event loops in the client.

The reference implementation for using external I/O event loops in the client is the GPGME library, which exports its own external I/O event loop mechanism and utilizes the Assuan library transparently for the user. The following steps document how GPGME achieves this.

1. Before connecting, set up pipes for bulk data transfer (using the INPUT/OUTPUT commands, for example). These are passed to the server either by inheritance (using a pipe server) or by FD passing (using a socket server).
2. Then you need to connect to the server. GPGME uses a pipe server, so it just spawns a server process, which is a non-blocking operation. FIXME: Currently, using a client with external event loop over a socket connection is not supported. It is easy to support (we just need a variation of `assuan_socket_connect` which takes an already connected socket FD and turns it into an Assuan context), so if you need this let us know.
3. After connecting, get the inbound status FD with `assuan_get_active_fds` (the first one returned is the status FD). This FD can be duplicated if it is convenient (GPGME does this to be able to close this FD and associated callback handlers without disrupting Assuan's internals).
4. Then register the Assuan inbound status FD and all bulk data FDs with the I/O event mechanism. In general, this requires setting up callback handlers for these FDs and registering them with the main event loop.

5. When bulk data FDs become ready, you can simply perform the corresponding read or write operations. When the inbound status FD becomes ready, you can receive the next server line with assuan_read_line().
6. You should close and unregister the bulk data FDs when you wrote all data (for outbound FDs) or receive an EOF (for inbound FDs). When you receive an ERR from the server, or an OK for the final operation, you can unregister the inbound status FD and call `assuan_release`.
7. As noted above, all send operations on the outbound status FD are done immediate with blocking. In GPGME, this has never caused any problems.
8. The `INQUIRE` function can be handled in two ways: If the requested data is immediately available, the client can just send the data blockingly. If the requested data needs to be fetched from a blocking source, a callback handler can be registered for the FD with the main event loop. GPGME does not support the `INQUIRE` function, so we do not have any practical experience with this.

Currently, the client can not cancel a pending operation gracefully. It can, however, disconnect from the server at any time. It is the responsibility of the server to periodically send status messages to the client to probe if the connection remains alive.

8.2 External I/O event loops in the server.

Currently, no Assuan server exists which uses external I/O event loops. However, the following guidelines should lead to a usable implementation:

1. For socket servers: You can not use `assuan_accept`, so you should just implement the bind/connect/listen/accept stage yourself. You can register the listen FD with your main event loop, accept the connection when it becomes ready, and finally call `assuan_init_socket_server` with the final argument being `ASSUAN_SOCKET_SERVER_ACCEPTED` to create an Assuan context for this connection. This way you can also handle multiple connections in parallel. The reference implementation for this approach is DirMngr.

   For pipe servers: `assuan_init_pipe_server` creates an Assuan context valid for the pipe FDs.
2. Once you have a context for a single connection, you can get the inbound status FD with `assuan_get_active_fds` (the first one returned is the status FD). This FD can be duplicated if it is convenient. Every time the inbound status FD is readable, you should invoke the function `assuan_process_next` (see below) to process the next incoming message. `assuan_process_next` processes as many status lines as can be received by a single `read` operation. When it returns, the inbound status FD may still be readable, but Assuan does not check this.

   The function `assuan_process_next` returns 0 if it can not make progress reliably, and it returns true in `done` if the client closed the connection. See below for more information on this function.
3. The command will be dispatched by `assuan_process_next` just as with `assuan_process`, however, you will want to implement the command handlers in such a way that they do not block. For example, the command handler may just register the bulk data FDs with the main event loop and return.

When the command is finished, irregardless if this happens directly in the command handler or later, you must call `assuan_process_done` with an appropriate error value (or 0 for success) to return an appropriate status line to the client. You can do this at the end of the command handler, for example by ending it with `return assuan_process_done (error_code);`. Another possibility is to invoke `assuan_process_done` from the place in the code which closes the last active bulk FD registered with the main event loop for this operation.

It is not possible to use `assuan_inquire` in a command handler, as this function blocks on receiving the inquired data from the client. Instead, the asynchronous version `assuan_inquire_ext` needs to be used (see below), which invokes a callback when the client provided the inquired data. A typical usage would be for the command handler to register a continuation with `assuan_inquire_ext` and return 0. Eventually, the continuation would be invoked by `assuan_process_next` when the client data arrived. The continuation could complete the command and eventually call `assuan_process_done`.

Cancellation is supported by returning an appropriate error value to the client with `assuan_process_done`. For long running operations, the server should send progress status messages to the client in regular intervals to notice when the client disconnects.

gpg_error_t assuan_process_next (*assuan_context_t* **ctx**, *int *done*) [Function]

This is the same as `assuan_process` but the caller has to provide the outer loop. He should loop as long as the return code is zero and *done* is false.

gpg_error_t assuan_process_done (*assuan_context_t* **ctx**, *gpg_error_t* **rc**) [Function]

Finish a pending command and return the error code *rc* to the client.

gpg_error_t assuan_inquire_ext (*assuan_context_t* **ctx**, *const char **__keyword__, *size_t* **maxlen**, *gpg_error_t* (***__cb__) (*void *cb_data, gpg_error_t rc, unsigned char *buffer, size_t buffer_len*), *void **__cb_data__) [Function]

This is similar to `assuan_inquire` but the caller has to provide the outer loop (using `assuan_process_next`). The caller should specify a continuation with *cb*, which receives *cb_data* as its first argument, and the error value as well as the inquired data as its remaining arguments.

9 Utility functions

There are a lot of helper functions to make writing Assuan code easier. Some of these functions provide information not available with the general functions.

`gpg_error_t` **`assuan_write_status`** (*assuan_context_t* **`ctx`**, *const char* *__`keyword`__, *const char* *__`text`__) [Function]

> This is a convenience function for a server to send a status line. You need to pass it the *keyword* and the content of the status line in *text*.

`gpg_error_t` **`assuan_inquire`** (*assuan_context_t* **`ctx`**, *const char* *__`keyword`__, *unsigned char* **__`r_buffer`__, *size_t* *__`r_length`__, *size_t* **`maxlen`**) [Function]

> A server may use this function to request specific data from a client. This function sends an 'INQUIRE' command back to the client and returns the client's response in a newly allocated buffer. You need to pass at least the server's context (*ctx*) and a description of the required data (*keyword*). All other parameters may be `NULL` or `0`, but this is rarely useful.
>
> On success the result is stored in a newly allocated buffer stored at *r_buffer*. The length of the data is stored at *r_length*. If *maxlen* has not been given as `0`, it specifies an upper size limit of the expected data. If the client returns too much data the function fails and an error with the error code `GPG_ERR_ASS_TOO_MUCH_DATA` will be returned.

`FILE*` **`assuan_get_data_fp`** (*assuan_context_t* **`ctx`**) [Function]

> Return a stdio stream for the Assuan context *ctx*. This stream may then be used for data output (assuan_write_data). The stream is valid until the end of the current handler. Calling `fclose` for that stream is not required. Assuan does all the buffering needed to insert the status line as well as the required line wrapping and quoting for data lines.
>
> This function is only available on systems supporting either `funopen` or `fopencookie`. If it is not supported `NULL` is returned and `errno` is set to `ENOSYS`.

`gpg_error_t` **`assuan_set_okay_line`** (*assuan_context_t* **`ctx`**, *const char* *__`line`__) [Function]

> Set the text used for the next `OK` response to *line*. This is sometimes useful to send additional human readable information along with the OK line. The string is automatically reset at the end of the current handler.

`gpg_error_t` **`assuan_command_parse_fd`** (*assuan_context_t* **`ctx`**, *char* *__`line`__, *assuan_fd_t* *__`rfd`__) [Function]

> This is the core of the default `INPUT` and `OUTPUT` handler. It may be used in custom commands as well to negotiate a file descriptor. If *line* contains `FD=`*n*, it returns *n* in *rfd* assuming a local file descriptor. If *line* contains just `FD` it returns a file descriptor at *rfd*; this file descriptor needs to have been sent by the client right before using `assuan_sendfd`.
>
> On W32 systems the returned file descriptor is a system handle and not a libc low level I/O file descriptor. Thus applications need to use `_open_osfhandle` before they can pass this descriptor to standard functions like `fdopen` or `dup`.

`const char * assuan_get_command_name` (*assuan_context_t* `ctx`) [Function]

Return the name of the command currently processed by a handler. The returned string is valid until the next call to an Assuan function on the same context. Returns `NULL` if no handler is executed or the command is not known.

`assuan_fd_t assuan_get_input_fd` (*assuan_context_t* `ctx`) [Function]

Return the file descriptor sent by the client using the last `INPUT` command. Returns `ASSUAN_INVALID_FD` if no file descriptor is available.

`assuan_fd_t assuan_get_output_fd` (*assuan_context_t* `ctx`) [Function]

Return the file descriptor sent by the client using the last `OUTPUT` command. Returns `ASSUAN_INVALID_FD` if no file descriptor is available.

`gpg_error_t assuan_close_input_fd` (*assuan_context_t* `ctx`) [Function]

Close the file descriptor set by the last `INPUT` command. This function has the advantage over a simple `close` that it can do some sanity checks and make sure that a following `assuan_get_input_fd` won't return an already closed descriptor.

`gpg_error_t assuan_close_output_fd` (*assuan_context_t* `ctx`) [Function]

Close the file descriptor set by the last `OUTPUT` command. This function has the advantage over a simple `close` that it can do some sanity checks and make sure that a following `assuan_get_input_fd` won't return an already closed descriptor.

`gpg_error_t assuan_set_error` (*assuan_context_t* `ctx`, *gpg_error_t* `err`, *const char* *`text`) [Function]

This is a helper to provide a more descriptive error text with `ERR` lines. For this to work, the text needs to be stored in the context *ctx* while still being in the command handler. This function is commonly called this way

```
return assuan_set_error (ctx, err, "commands needs 5 arguments");
```

The value *err* is passed through and thus the return value of the command handler in the example. The provided text further explains that error to humans.

`pid_t assuan_get_pid` (*assuan_context_t* `ctx`) [Function]

This function returns the pid of the connected connected peer. If that pid is not known `ASSUAN_INVALID_PID` is returned. Note that it is not always possible to learn the pid of the other process. For a pipe based server the client knows it instantly and a mechanism is in place to let the server learn it. For socket based servers the pid is only available on systems providing the `SO_PEERCRED` socket option[1].

`assuan_peercred_t` [Data type]

This structure is used to store the peer credentials. The available members depend on the operating system.

`pid_t pid` The process ID of the peer.

`uid_t uid` The user ID of the peer process.

`gid_t gid` The group ID of the peer process.

[1] to our knowledge only the Linux kernel has this feature

`gpg_error_t` **`assuan_get_peercred`** (*assuan_context_t* `ctx`, *assuan_peercred_t* *`peercred`) [Function]

Return user credentials of the peer. This will work only on certain systems and only when connected over a socket. On success, a pointer to the peer credentials is stored in *peercred*. The information is only valid as long as the state of the connection is unchanged (at least until the next assuan call to the same context).

As of now only the server is able to retrieve this information. Note, that for getting the pid of the peer `assuan_get_pid` is usually better suited.

`int` **`assuan_get_active_fds`** (*assuan_context_t* `ctx`, *int* `what`, *assuan_fd_t* *`fdarray`, *int* `fdarraysize`) [Function]

Return all active file descriptors for the context *ctx*. This function can be used to select on the file descriptors and to call `assuan_process_next` if there is an active one. The first descriptor in the array is the one used for the command connection. Currently *what* needs to be `0` to return descriptors used for reading, `1` will eventually be used to return descriptors used for writing. *fdarray* is an array of integers provided by the caller; *fdarraysize* gives the size of that array.

On success the number of active descriptors are returned. These active descriptors are then stored in *fdarray*. On error `-1` is returned; the most likely reason for this is a too small *fdarray*.

Note that on W32 systems the returned file descriptor is a system handle and not a libc low level I/O file descriptor.

`int` **`assuan_pending_line`** (*assuan_context_t* `ctx`) [Function]

A call to this function return true if a full line has been buffered and thus an entire assuan line may be read without triggering any actual I/O.

10 Socket wrapper functions

It is sometimes useful to support Unix domain sockets on Windows. To do this in a portable way, Assuan provides a set of wrapper functions which may be used on any system but will enhance Windows to support these socket types. The actual implementation is based on local TCP sockets and fully transparent for the client. Server code needs to utilize two extra functions to check the permissions.

`gpg_error_t assuan_sock_init` (*void*) [Function]

Initialize the socket wrappers. Must be called once at startup if any of the socket wrapper functions are used.

`gpg_error_t assuan_sock_deinit` (*void*) [Function]

Deinitialize the socket wrappers.

`int assuan_sock_close` (*assuan_fd_t* ***fd***) [Function]

Wrapper for close which does a closesocket on Windows if needed.

`assuan_fd_t assuan_sock_new` (*int* ***domain***, *int* ***type***, *int* ***proto***)*;* [Function]

Wrapper around socket.

`int assuan_sock_connect` (*assuan_fd_t* ***sockfd***, *struct sockaddr* *****addr***, *int* ***addrlen***) [Function]

Wrapper around connect. For Unix domain sockets under Windows this function also does a write immediately after the the connect to send the nonce as read from the socket's file. Under Unix this function check whether the socket file is a redirection file and connects to the redirected socket instead; see `assuan_sock_set_sockaddr_un` for details on the redirection file format.

`int assuan_sock_connect_byname` (*const char ** ***host***, *unsigned short* ***port***, *int* ***reserved***, *const char ****credentials***, *unsigned int* ***flags***) [Function]

Directly connect to *port* on *host* given as a name. The current implementation requires that *flags* has either `ASSUAN_SOCK_SOCKS` or `ASSUAN_SOCK_TOR` set. On success a new TCP STREAM socket is returned; on error `ASSUAN_INVALID_FD` and ERRNO set. If *credentials* is not `NULL`, it is a string used for password based SOCKS authentication. Username and password are separated by a colon. *reserved* should be 0. To test whether the proxy is available *host* and *port* may be given as NULL/0: If the proxy is available the function returns a valid socket which is in the state after credentials sub-negotiation. The caller now knows that the SOCKS proxy is available and has been authenticated; normally the caller closes the socket then.

`int assuan_sock_bind` (*assuan_fd_t* ***sockfd***, *struct sockaddr ****addr***, *int* ***addrlen***) [Function]

Wrapper around bind. Under Windows this creates a file and writes the port number and a random nonce to this file.

`int assuan_sock_set_sockaddr_un` (*const char ****fname***, *struct sockaddr ****addr***, *int ****r_redirected***) [Function]

This is a helper function to initialize the Unix socket domain address structure *addr* and store the file name *fname* there. If *r_redirected* is not NULL the function checks whether *fname* already exists, is a regular file, and not a socket. In that case *fname* is read to see whether this is a redirection to a socket file. If that is the case 1 is stored at *r_redirected*. If the file does not look like a redirection file 0 will be stored there and *fname* will be used in the regular way.

The format of a redirection file is

```
%Assuan%
socket=name
```

With *name* being is the actual socket to use. No white spaces are allowed, both lines must be terminated by a single linefeed, and extra lines are not allowed. Environment variables are interpreted in *name* if given in `${VAR}` notation. No escape characters are defined; if the string `${` shall be used in file name, an environment variable with that content may be used. The length of the redirection file is limited to 511 bytes which is more than sufficient for any known implementation of Unix domain sockets.

`int assuan_sock_get_nonce` (*struct sockaddr *`addr`, int `addrlen`, assuan_sock_nonce_t *`nonce`*) [Function]

This is used by the server after a bind to return the random nonce. To keep the code readable this may also be used on POSIX system.

`int assuan_sock_check_nonce` (*assuan_fd_t `fd`, assuan_sock_nonce_t *`nonce`*) [Function]

If the option `ASSUAN_SOCKET_SERVER_ACCEPTED` has been used, Libassuan has no way to check the nonce of the server. Thus an explicit check of the saved nonce using this function is required. If this function fails the server should immediately drop the connection. This function may not be used if Libassuan does the accept call itself (i.e. `ASSUAN_SOCKET_SERVER_ACCEPTED` has not been used) because in this case Libassuan calls this function internally. See also `assuan_set_sock_nonce`.

Actually this mechanism is only required on Windows but for cleanness of code it may be used on POSIX systems as well, where this function is a nop.

To control certain properties of the wrapper two additional functions are provided:

`int assuan_sock_set_flag` (*assuan_fd_t `fd`, const char *`name`, int `value`*) [Function]

Set the flags *name* for socket *fd* to *value*. See below for a list of valid names. Returns 0 on success; on failure sets ERRNO and returns -1.

`int assuan_sock_get_flag` (*assuan_fd_t `fd`, const char *`name`, int *`r_value`*) [Function]

Store the current value of the flag *name* for socket *fd* at *r_value*. See below for a list of valid names. Returns 0 on success; on failure sets ERRNO and returns -1.

The supported flags are:

`cygwin` This flag has an effect only on Windows. If the value is 1, the socket is set into Cygwin mode so that Cygwin clients can connect to such a socket. This flag needs to be set before a bind and should not be changed during the lifetime of the socket. There is no need to set this flag for connecting to a Cygwin style socket because no state is required at the client. On non-Windows platforms setting this flag is ignored, reading the flag always returns a value of 0.

`tor-mode`
`socks`

If *value* is 1 globally enable SOCKS5 mode for new connections using IPv6 or IPv4. *fd* must be set to `ASSUAN_INVALID_FD` A future extension may allow to disable SOCKS5 mode for a specified socket but globally disabling SOCKS5 mode is not possible. Using the flag "tor-mode" expects the SOCKS5 proxy to listen on port 9050, the flag "socks" expects the proxy to listen on port 1080.

Connections to the loopback address are not routed though the SOCKS proxy. UDP requests are not supported at all. The proxy will be connected at address 127.0.0.1; an IPv6 connection to the proxy is not yet supported.

GNU Lesser General Public License

Version 2.1, February 1999

Copyright © 1991, 1999 Free Software Foundation, Inc.
59 Temple Place – Suite 330, Boston, MA 02111-1307, USA

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

[This is the first released version of the Lesser GPL. It also counts
as the successor of the GNU Library Public License, version 2, hence the
version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, 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 this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the *Lesser* General Public License because it does *Less* to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

   A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The “Library”, below, refers to any such software library or work which has been distributed under these terms. A “work based on the Library” means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term “modification”.)

“Source code” for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library’s complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. The modified work must itself be a software library.

   b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

   c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

   d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

      (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

   Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

   This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

   If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

   However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

   When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

   You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

   a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
   b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user’s computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
   c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
   d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
   e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

   For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

   b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), 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 distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser 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 Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "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 LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN

WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY 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 LIBRARY (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 LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey 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 library's name and an idea of what it does.
Copyright (C) year  name of author

This library is free software; you can redistribute it and/or modify it
under the terms of the GNU Lesser General Public License as published by
the Free Software Foundation; either version 2.1 of the License, or (at
your option) any later version.

This library 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
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307,
USA.
```

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

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the library
'Frob' (a library for tweaking knobs) written by James Random Hacker.

signature of Ty Coon, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!

GNU General Public License

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. http://fsf.org/

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

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
```

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`.

Index