GNUstep DBusKit and D-Bus Programming Manual

Niels Grewe

Table of Contents

GNUstep D-Bus Programming Manual

1 Introduction

The aim of this manual is to familiarise the reader with the concepts and tools necessary to successfully integrate a GNUstep application into a desktop environment based around message exchange through the D-Bus messaging bus facilities. The manual tries to give succinct explanation of the concepts involved, providing illustrative examples whenever possible.

It will be most useful to a reader who has basic working knowledge of the Objective-C programming language and the OpenStep APIs (either from the GNUstep implementation or from Apple's Cocoa). In depth knowledge of the Distributed Objects system or D-Bus is also beneficial but not required.

1.1 An IPC primer

A typical modern computer system executes multiple units of computation at the same time. Even with a single-core CPU, the operating system will constantly switch between different units of computation by employing different multitasking strategies. This approach has a number of advantages, e.g.:

- It facilitates isolation of processes from one another: A malignant process cannot easily modify the memory of other processes on the system.
- It allows privilege separation: It is not necessary that a web-browser has the same rights as a partitioning utility. Running both in different processes allows the operating system to assign different privileges to both.
- It increases modularity: You can easily change one part of the software on your computer without disturbing the other parts.
- If the computer has more than one CPU, computation can be sped up by running more than one process (or thread) in parallel.

To leverage these advantages effectively, different processes or applications need a mechanism for inter-process communication (IPC) that allows them to exchange information (and ensure synchronisation if needed).

One way to implement an IPC mechanism is by using the message passing paradigm. Entities in a message passing system communicate by exchanging messages with each other, which makes it a natural fit for object oriented languages, where the basic abstraction is the object.

The message passing paradigm is also used in Objective-C (actually Objective-C inherited it from Smalltalk), where you interact with objects by sending messages to them. E.g. the intended meaning of

```
[alice greet];
```

would be sending the `-greet` message to the `alice` object, which is referred to as the *receiver* of the message. This idiom can be quite easily extended beyond the single process case, which the NeXT did by including the *Distributed Objects* system in the OpenStep specification that GNUstep implements. The message passing paradigm is also employed by D-Bus, and we will look at the similarities and differences of both systems in the following sections.

1.2 Distributed Objects

The GNUstep Distributed Objects (DO) System is designed to go out of a programmer's way. Since ordinary (intra-process) usage Objective-C already has message passing semantics, Distributed Objects simply extends these semantics to objects in other processes.

This works by usage of the proxy design pattern. A proxy is a stand-in object that receives messages *in lieu* of another object and forwards them (most likely after processing them as it sees fit). In the case of Distributed Objects, the proxy will take the message that is being sent to the remote object, encode it a `NSInvocation` object and send a serialised version of the invocation to the remote process where it is invoked on the receiver it was initially intended for.

Establishing a connection to a remote object using DO is thus a simple three step process:

1. Look up a process that exposes ('vends', in DO parlance) an object.
2. Establish a communication channel to the process.
3. Create a proxy object to send messages to the remote object.

Afterwards, the generated proxy can be used just like any in-process object.

Task 1. involves the `NSPortNameServer` class which can be used to obtain a communication endpoint (`NSPort`) to a service with a specific name:

```
NSPort *sendPort = [[NSPortNameServer systemDefaultPortNameServer]
  portForName: @"MyService"];
```

Task 2. involves `NSPort` and `NSConnection`. While the former is concerned with the low-level details of encoding messages to a wire format, the latter manages sending messages over ports. A connection to the above `MyService` using the created `sendPort` could be obtained like this:

```
NSConnection *c = [NSConnection connectionWithReceivePort: [NSPort port]
                                                 sendPort: sendPort];
```

Task 3. is done by calling `-rootProxy` on the `NSConnection` object. This will return an instance of `NSDistantObject`: A proxy that will use `NSConnection` and `NSPort` to forward messages to the remote object.

```
id remoteObject = [c rootProxy];
```

The DO mode of operation has a few notable advantages:

- Usual message passing semantics apply.
- The native Objective-C type system is used in both processes. No type conversion is necessary.
- New objects can be vended implicitly by returning them from the root proxy. New proxies will be created automatically for them.
- DO can make intelligent decisions about the remote objects: If process *A* has vended object *O* to process *B* (yielding the proxy *P(O)*), and *B* latter vends *P(O)* to *A*, *A* will not use *P(P(O))*, but its local reference to *O*.

It goes without saying that DO is pretty useful and GNUstep uses it in many places. It drives, for example, the services architecture, the pasteboard server, or the distributed notification system. For further information about DO, please consult the Objective-C GNUstep Base Programming Manual. We will now turn our attention to the D-Bus IPC system.

1.3 D-Bus

Distributed Objects has already been part of NeXT's OpenStep Specification, which appeared in 1994 and thus predates the D-Bus IPC system for quite some time. But while DO is only useful in an Objective-C context, D-Bus was created to suit the needs of desktop environments such as KDE or GNOME, which use (among others) C or C++ as their core programming languages.

1.3.1 Message Busses

One core concept of D-Bus is that of the message bus. A standard desktop system that uses D-Bus usually has two active message buses, dubbed the *well-known buses*. One is the *system bus*, to which system-wide services connect, the other is the *session bus* which is started per user session and allows applications on the user's desktop to communicate.

The purpose of a bus, which is running as a separate process (the *dbus-daemon*), is to provide name-services to the connected applications and route messages between them.

1.3.2 Services

A process that connects to a message bus is considered to be a *service*, even if it will not expose any object to the bus. A unique name, which starts with a colon (e.g. *:1.1*) and is required for message routing, will be assigned to every service by the bus. The service can also request further names from the bus. A text editor might, for example, want to request the name *org.gnustep.TextEditor* from the bus. These names are referred to as *well-known names* and usually utilise reverse-DNS notation.

These names can be subject to different assignment policies. A service can specify that it wants to be queued for a name that has already be assigned. It will then become the owner of the name when the last previous owner exits or releases the name. Alternatively, the service can request to replace an existing name, a feature that can be used to ensure that only one application of a specific type is running (as would be the case for, e.g., a screensaver).

1.3.3 Object Paths

When using DO, the object graph vended by a service is generated implicitly: If a message send to a remote object returns another object, that object will implicitly be vended and wrapped in a proxy for use by the other process. D-Bus operates quite differently in that respect: Every object needs to be assigned a name that can be used by remote processes to interact with the object. These object names are organised in the directory-like structure, where each object is uniquely identified by its *object path*. The UDisks service (*org.freedesktop.UDisks*) on the system bus does, for example, expose different disks of a computer at different paths:

```
/org/freedesktop/UDisks/devices/sda
/org/freedesktop/UDisks/devices/sdb
```

It is worth noting that it is a D-Bus convention to have the root object of the service not reside at the root path ("/") but at one that corresponds to the service name with all dots replaced by the path separator. Thus you don not access the root object of *org.freedesktop.UDisks* at "/" but at "/org/freedesktop/UDisks". The reason for this is to ensure proper name-spacing if different code modules in a single process have registered multiple names on the bus (which will all point to the same unique name).

1.3.4 Interfaces

D-Bus object-path nodes are the receivers and senders of D-Bus messages. They receive calls to methods and emit signals, which are broadcast by the bus and can be watched for by other applications. These methods and signals can be aggregated into *interfaces*, which are a bit, but not quite, like Objective-C protocols. One interface that almost every D-Bus object implements is *org.freedesktop.Introspectable*, which has as its sole member the `Introspect()`-method. This will return XML-encoded information about all methods, signals, and properties the object exposes.

Interfaces are also used as namespaces for their members: Identically named methods with different implementations are allowed to appear in multiple interfaces, something that is not possible with Objective-C protocols.

1.3.5 Type System

For arguments and values of methods, signals, and properties, D-Bus defines its own type system, which is similar to the C type system. It contains integer and floating point types of different sizes as well as array and structure types. The type system represents dictionaries as arrays of ordered pairs. Additionally, there is a type available for references to objects (but these references are only valid within a single service) and a variant type that, just like Objective-C's `id`, allows for values of arbitrary types. This type system has to be adopted by any application that wants to interface with D-Bus.

1.4 Comparison

Feature	**Distributed Objects**	**D-Bus**
IPC paradigm	message passing	message passing
type system	native Objective-C type system	custom D-Bus type system (C-like)
supported programming languages	Objective-C[1]	many languages through bindings
polymorphism	no special provisions	through overloaded method names in different interfaces
object-graph generation	implicit	explicit with named objects
name service	provided by separate nameserver objects	integrated
delivery of broadcast information	distributed notification system implemented on top of DO	integrated as D-Bus signals

[1] Please note that the GNUstep and Apple implementations of Distributed Objects are incompatible.

2 Using D-Bus From Objective-C

In order to access D-Bus services from an Objective-C application, the DBusKit framework is required. It provides infrastructure for managing connections to D-Bus message buses and translating Objective-C message sends to D-Bus method calls. This way, DBusKit can make interacting with D-Bus objects appear quite similar to the way one usually interacts with the DO system.

2.1 Generating Protocol Declarations With dk_make_protocol

If your application wants to invoke methods on D-Bus objects, some preparations are required: As with all other code, you need to provide declarations for the methods you want to invoke. You can either do this by writing them manually or let the **dk_make_protocol** tool generate them for you. This is possible if an .interface-file containing the introspection data for the interface exists. Calling **dk_make_protocol** with the "**-i**" switch and the name of the .interface-file will generate a header file with an Objective-C protocol declaration for that interface. For the hypothetical interface file for *org.freedesktop.Introspectable*, **dk_make_protocol** might generate the following header file:

```
#import <Foundation/Foundation.h>
/*
 * Objective-C protocol declaration for the D-Bus
 * org.freedesktop.Introspectable interface.
 */
@protocol org_freedesktop_Introspectable

- (NSString*)Introspect;

@end
```

The generated header file does only contain method declarations with arguments and return values that are Objective-C classes. The following default mappings between Foundation classes and D-Bus types are defined:

NSNumber	booleans (b), integers (y, n, q, i, u, x, t), floating point values (d)
NSString	strings (s)
DKProxy	object paths (o)
NSArray	arrays (a?), structs ((?*))
NSDictionary	dictionaries (a{??})
id	variants (v)

Here "?" denotes a single complete D-Bus type signature and "*" denotes possible repetition. It is, however, possible to use the plain C types corresponding to the D-Bus types, because DBusKit is capable of determining all necessary conversions. Thus the following declarations all specify valid ways to invoke `NameHasOwner()` method from *org.freedesktop.DBus*:

```
- (NSNumber*)NameHasOwner: (NSString*)name;
- (NSNumber*)NameHasOwner: (char*)name;
- (BOOL)NameHasOwner: (NSString*)name;
- (BOOL)NameHasOwner: (char*)name;
```

2.2 Obtaining a Proxy to a D-Bus Object

With these provisions in place, it is quite easy to obtain a proxy to a D-Bus object. The process is quite similar to creating a proxy to a distant object using DO. First, you create the required ports:

```
DKPort *sPort = [[DKPort alloc] initWithRemote: @"org.freedesktop.DBus"
                                         onBus: DKDBusSessionBus]
DKPort *rPort = [DKPort sessionBusPort];
```

If a service on the system bus was the desired target, one could pass `DKBusSystemBus` as the second argument of the `DKPort` initialiser or use the `+systemBusPort` convenience method to create a port object without remote.

Afterwards, a connection can be obtained to the *org.freedesktop.DBus* service (which is bus itself) as follows:

```
NSConnection *c = [NSConnection connectionWithReceivePort: rPort
                                                 sendPort: sPort];
```

Please note that this is exactly the way one would create a Distributed Objects connection. Consequentially, on can obtain a proxy to an object of this service by using `-rootProxy`:

```
id remoteObject = [c rootProxy];
```

Unfortunately, a proxy to the root object of a D-Bus service is very often not useful because services tend to install their primary object at a path corresponding to the service name. DBusKit thus extends `NSConnection` with a `-proxyAtPath:` method, which can be used to obtain proxies to non-root object. It could be used to obtain a proper proxy to *org.freedesktop.DBus* like this:

```
id remoteObject = [c proxyAtPath: @"/org/freedesktop/DBus"];
```

2.3 Sending Messages to D-Bus Objects

All further interactions with the remote object are indistinguishable from interactions with an object in the local process. E.g. the introspection data of the remote object could be obtained like this:

```
NSString *introspectionData = [remoteObject Introspect];
```

In some cases it is, however, necessary to treat D-Bus objects special: Since D-Bus allows method names to be overloaded per interface, it might be necessary to specify which method to call. DBusKit provides two facilities to cope with this kind of situation. For one, it is possible to embed the information about the required interface in the selector string of the method to call. This is done by replacing all dots in the interface string with underscores, placing it between `_DKIf_` `_DKIfEnd_` marker and appending the method name.

Assuming a D-Bus object implements a `getBass()` method in the interfaces `org.foo.Fish` and `org.bar.Instruments`, one could distinguish between the methods by constructing the following selectors:

- `-_DKIf_org_foo_Fish_DKIfEnd_getBass`
- `-_DKIf_org_bar_Instruments_DKIfEnd_getBass`

Since this is obviously quite clumsy, it will only be feasible for simple cases.

The other facility provided by DBusKit is the `-setPrimaryDBusInterface:` method, which instructs the proxy to prefer the named interface when looking up methods. E.g. the following statements would result in a call to the correct method:

```
[remoteObject setPrimaryDBusInterface: @"org.bar.Instruments"];
id anInstrument = [remoteObject getBass];
```

2.4 Accessing and changing D-Bus properties

DBusKit will automatically generate getters and setters for D-Bus properties. A D-Bus interface might, for example, specifythe following property in its introspection data:

```
<property name="address" type="s" access="readwrite"/>
```

This property can then be accessed by calling `-address` and changed by calling `-setaddress:` on the proxy object. Just like with other methods, both the plain C types and the corresponding Foundation classes are valid as parameters to the getter and setter methods:

```
- (NSString*)address;
- (char*)address;
- (void)setaddress: (NSString*)address;
- (void)setaddress: (char*)address;
```

If other methods with the same names exist within the same interface of the remote object, those will take precedence over the generated getter and setter methods.

2.5 Watching D-Bus Signals

Besides responding to method calls, D-Bus objects can also actively inform remote objects about events or state changes by the use of *signals*. These signals are published to the bus and the bus will re-broadcast them to all connected entities that subscribe to the signals. DBusKit includes support for receiving D-Bus signals through the `DKNotificationCenter` class. `DKNotificationCenter` keeps to OpenStep conventions in that it delivers the signals it receives from D-Bus in the form of `NSNotification`s and is thus similar to the notification center classes provided by the Foundation library (gnustep-base).

To make use of the notification feature, it is sometimes not even necessary to create any explicit proxies. It is enough to just obtain a reference to one of the notification centers:

```
DKNotificationCenter *center = [DKNotificationCenter sessionBusCenter];
```

(Again, a reference to the notification center for the system bus can be obtained similarly by using `+systemBusCenter`.) In a very simple case, one would simply use the center to add an object as an observer of the *NameAcquired* signal from the *org.freedesktop.DBus* interface.

```
[center addObserver: myObject
           selector: @selector(didReceiveNotification:)
               name: @"DKSignal_org.freedesktop.DBus_NameAquired"
             object: nil];
```

This example also illustrates the naming convention for signals: They start with the "`DKSignal`"-identifier and continue with the interface name and the signal name separated by underscores ("`_`"). Additionally, it is possible to register a custom notification name for a signal:

```
[center registerNotificationName: @"DKNameAquired"
                        asSignal: @"NameAquired"
                     inInterface: @"org.freedesktop.DBus"];
```

If this method returns YES, it will be possible to register observers for the `DKNameAquired` notification (it might fail if the signal was already registered under another name).

Since D-Bus provides a fine-grained matching mechanism for signals, Objective-C applications can specify in great detail what kind of signal they want to receive. The full-blown version of the registration method could be called as follows:

```
[center addObserver: myObject
           selector: @selector(didReceiveNotification:)
             signal: @"NameOwnerChanged"
          interface: @"org.freedesktop.DBus"
             sender: theBus
        destination: nil
             filter: @"org.gnustep.TextEditor"
            atIndex: 0];
```

If registered as an observer this way, `myObject` would only receive a notification if a new application took ownership of the name *org.gnustep.TextEditor*.

When delivering annotification to the observer, the notification center will create a `NSNotification` with a userInfo dictionary that follows a specific format to allow the receiver to process the notification:

member The name of the signal being emitted.

interface The name of the interface the signal belongs to.

sender The *unique* name of the service emitting the signal.

path The path to the object of the service that emitted the signal.

destination
The intended receiver of the signal; might be empty if the signal was broadcast, which is usually the case.

arg0, ..., n
If the signal did specify any values to be send alongside the signal, these values will be present in keys called *arg0*, *arg1*, ..., *argn*.

Additionally, calling `-object` on the notification will return a proxy to the object that emitted the signal.

2.6 Recovering from Failure

There are two common reasons for failure when communicating with objects on D-Bus. One is that the service your application is accessing is going away. In that case, DBusKit will notify you in a way similar to Distributed Objects. This means that when the service disappears from the bus, the `DKPort` used will post a `NSPortDidBecomeInvalidNotification` to the default notification center. You can watch for this notification and attempt recovery afterwards.

A more critical reason for failure is a malfunction or restart of the D-Bus daemon. In that case, all affected ports will issue a `NSPortDidBecomeInvalidNotification` and additionally the `DKDBus` object for the bus will post a `DKBusDisconnectedNotification` with the `DKDBusBusType` identifier at the `busType` key of the userInfo dictionary. Afterwards, DBusKit will attempt to recover from the failure in the background and you cannot use D-Bus services until you receive a `DKBusReconnectedNotification`. After receiving the notification, you can perform recovery as your application requires.

Please note that usually, such recovery from bus failures will only be successful for the system bus, for which one connects to a socket address that is persistent across restarts. For the session bus the socket address is not persistent, but stored in the `DBUS_SESSION_BUS_ADDRESS` environment variable. Hence your application should assume that the user session died when it looses connection to the session bus.

2.7 Multi-Threading Considerations

By default, DBusKit runs in single-threaded mode. This means that all interaction with the D-Bus daemon happens on the runloop of the calling thread. If multiple threads try to send messages D-Bus objects, this model of execution cannot guarantee that message delivery from and to the bus daemon are successful. The framework should still be thread-safe in the sense that it will continue functioning after raising an exception due to timeouts, but the desired behaviour can only be acheived by putting DBusKit in multi-threaded mode.

In multi-threaded mode, DBusKit will exchange messages with the D-Bus daemons via a dedicated worker-thread. To enable this behaviour the `+enableWorkerThread` method must be called on `DKPort`. All processing will then be taking place on the worker thread. Developers should note that after doing so, it is no longer safe to call into DBusKit from `+initialize`-methods. The reason for this is that with present Objective-C runtimes, `+initialize` will obtain a global lock and subsequent initialisations of classes on the worker thread might cause a deadlock. This is also the reason multi-threaded operation is not the default. But developers are encouraged to use this feature if it does not interfere with their application design.

3 Exposing Objects On D-Bus

Unfortunately, the present version of DBusKit does not include support for exposing objects in an Objective-C application to other applications via D-Bus.

Appendix A The GNU Free Documentation License

0. PREAMBLE

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

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

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

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

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

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

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

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

4. MODIFICATIONS

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

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

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

8. TRANSLATION

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

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

9. TERMINATION

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

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

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

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

10. FUTURE REVISIONS OF THIS LICENSE

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

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

11. RELICENSING

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

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

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

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

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

ADDENDUM: How to use this License for your documents

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

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

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

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

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

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

Concept Index