

The Birds Project

# The libBirds Library, Software Development Plan

Ronald S. Burkey

Version 0.5

10/31/01

Copyright © 2001 by Ronald S. Burkey

*Licensing information:* Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation with Invariant Sections "GNU Free Documentation License" and "GNU Lesser General Public License", with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

# 1. Purpose of the SDP Document

This is a standard "Software Development Plan" document, corresponding to the guidelines in RTCA DO-178B. It describes the standards, software life cycle, and development environment used in the software-development effort.

This document is available for free use, under the terms of the GNU Free Documentation License (FDL). The text of the FDL is reproduced later in this document.

## 2. Standards

### 2.1. Software Requirements Standards

The software requirements standards are defined by the document entitled *The libBirds Library, Software Requirements Standards*.

### 2.2. Software Design Standards

The software design standards are defined by the document entitled *The libBirds Library, Software Design Standards*.

### 2.3. Software Code Standards

The software code standards are defined by the document entitled *The libBirds Library, Software Coding Standards*.

### 2.4. Standards for Previously Developed Software

Previously developed software, if used, is required to be certified via DO-178B, at the same software level (or higher) than the software under development.

In the case of software previously developed by the Birds Project, this shall be evidenced by the DO-178B documentation (and particularly the Software Conformity Review) of the previously developed software. In the case of software not developed by the Birds Project, this shall be evidenced by a signed 8110 form.

## **2.5. Standards for Commercial Off-the-Shelf Software**

The standards for COTS are identical to that for previously developed software.

## **3. Software-Component Life Cycles**

For this project, there is only one software component, namely the libBirds library, and hence only one life cycle.

### **3.1. Life Cycle of libBirds Library Development**

#### **3.1.1. Life-Cycle**

The life cycle of the libBirds library begins with the Planning Process. Upon the end of the planning processes, three separate chains of development begin, and these chains persist until the end of the development effort.

Two of the of the development chains consist of one process each: namely, the SCM Process and the SQA Process.

The third development chain consists of 4 processes: the Requirements Process, followed by the Design Process, the Coding Process, the Integration Process, and the Software Verification Process. For simplicity, we'll refer to this as the "RDCIV chain." The development effort basically progresses through these processes, in the order given, but can backtrack to an earlier process upon discovery of errors that can only be corrected in the earlier process.

Upon completion of the development effort, which is the release of the software by the SQA Process, life cycle data is available for input to a Certification Liaison Process. However, the Certification Liaison Process is really outside of the scope of the libBirds development effort, since the software produced is merely a reusable library and not a complete system.

Although not possible for the first libBirds release, due to non-availability of personnel, it is hoped that subsequent releases can be reviewed by DER as part of the development effort, allowing very rapid signoff of form 8110 for developers using the libBirds library. When this becomes possible, it will form part of the Certification Liaison Process.

## **Figure 1. Life Cycle Summary**

### **3.1.1.1. Life-Cycle Processes**

#### *3.1.1.1.1. Planning Process*

The Planning Process precedes all other life-cycle processes. It produces or identifies all other Plans or Standards guiding the remainder of the software-development effort. The specific aim of the libBirds Planning Process is to address all of the issues outlined in DO-178B section 4.0.

#### *3.1.1.1.1.1. Details of Implementation*

TBD

#### *3.1.1.1.1.2. Transition Criteria and Satisfaction of Objectives*

The Planning Process is followed by three separate chains of life-cycle processes, with the three development chains running simultaneously in parallel. These chains are the SCM Process, the SQA Process, and the RDCIV chain (see the 'Life-Cycle' section). The transitions from the Planning Process to these development chains do not necessarily occur simultaneously.

The Planning Process transitions to the RDCIV chain when the PSAC, SDP, SVP, SECI, SRS, SDS, and SCS documents have all been successfully reviewed and signed off. Note that the PSAC, unlike the other documents mentioned, is not actually completed at this point since it contains some high-level requirements (notably under the heading of 'System Overview') that are not known until the Requirements Process.

Subsequent changes to these documents may require a return from the Design Process, Coding Process, or Software Verification Process to the Requirements Process, but the Planning Process is never re-entered in any given life cycle.

The Planning Process transitions to the SCM Process upon successful review and signoff of the SCMP.

The Planning Process transitions to the SQA Process upon successful review and signoff of the SQAP.

#### *3.1.1.1.2. Requirements Process*

The purpose of the Requirements Process is to develop the high-level software requirements. In the case of libBirds, this is essentially a development of the functional requirements. However, the intention of the libBirds Requirements Process is to address all of the issues outlined in DO-178B section 5.1.

The Requirements Process also addresses all considerations of DO-178B section 6.3.1, which according to DO-178B may form a part of the Software Verification Process. This can be done because at the proposed software level ('C'), there is no requirement of independence.

##### *3.1.1.1.2.1. Details of Implementation*

TBD

##### *3.1.1.1.2.2. Transition Criteria and Satisfaction of Objectives*

The Requirements Process transitions to the Design Process upon successful review and signoff of the SRD. The completion of the incomplete PSAC produced in the Planning Process is also needed, along with its review and signoff.

#### *3.1.1.1.3. Design Process*

In the case of libBirds, the main purpose of the Design Process is to develop low-level requirements (primarily API definition) from the high-level requirements provided by the Requirements Process. However, all of the issues outlined in DO-178B section 5.2 are addressed.

The Requirements Process also addresses all considerations of DO-178B sections 6.3.2 & 6.3.3, which according to DO-178B may form a part of the Software Verification Process. This can be done because at the proposed software level ('C'), there is no requirement of independence.

#### *3.1.1.1.3.1. Details of Implementation*

TBD

#### *3.1.1.1.3.2. Transition Criteria and Satisfaction of Objectives*

The Design Process transitions to the Coding Process upon successful review and signoff of the SDD.

#### *3.1.1.1.4. Coding Process*

The purpose of the Coding Process is to produce source code and object code conforming to the SDD, SDS, and SCS. It addresses the issues outlined in DO-178B section 5.3.

##### *3.1.1.1.4.1. Details of Implementation*

TBD

##### *3.1.1.1.4.2. Transition Criteria and Satisfaction of Objectives*

The Coding Process Transitions to the Integration Process when the source code and object code are prepared, have been verified conformant with the SCS and SDS, and have been verified conformant with (and traceable to) the SDD.

#### *3.1.1.1.5. Integration Process*

The purpose of the Integration Process is to integrate the software with the target hardware. Since the aim of the libBirds project is to produce a highly portable library, rather than a hardware-specific library or a physical device, there really can be no required Integration Process.

For any given target architecture, the general-purpose libBirds library is combined with a specific 'board-support package' (not a part of libBirds as such) that provides the Hardware Abstraction Layer specific to that target. It is in the Integration Process of the board-support package life cycle or the application code life cycle (both of which are independent of libBirds) that the integration occurs.

#### *3.1.1.1.5.1. Details of Implementation*

No implementation of the Integration Process is technically required.

However, as a practical matter, it may be more efficient to coordinate BSP development with libBirds development, and therefore to carry out the Integration Process of one or more BSPs at this point -- or at least to attain some level of confidence that later problems in BSP certification will not require changes to libBirds.

The extent to which libBirds and BSP development efforts are coordinated cannot be specified in advance. Thus, the libBirds Integration process must be regarded as optional.

#### *3.1.1.1.5.2. Transition Criteria and Satisfaction of Objectives*

At the option of the developers, the Integration Process may either proceed directly to the Software Verification Process without any effort whatever, or else to optionally coordinate with board-support package (BSP) development-effort Integration Processes. There are no pre-defined criteria associated with this decision.

#### *3.1.1.1.6. Software Verification Process*

In the words of DO-178B section 6.1, "The purpose of the software verification process is to detect and report errors that may have been introduced during the software development processes." The libBirds Software Verification Process attempts to address all of the issues outlined in DO-178B chapter 6, except the following:

- 1) The Requirements Process addresses all considerations of DO-178B section 6.3.1. This can be done because at the proposed software level ('C'), there is no requirement of independence.
- 2) The Design Process addresses all considerations of DO-178B sections 6.3.2 & 6.3.3. This can be done because at the proposed software level ('C'), there is no requirement of independence.
- 3) Reviews and analysis of the outputs of the Integration Process (DO-178B section 6.3.5) are not addressed here (or elsewhere) since the entire Integration Process is optional. Refer to the 'Integration Process' section of the PSAC for further explanation.

#### *3.1.1.1.6.1. Details of Implementation*

TBD

#### ***3.1.1.1.6.2. Transition Criteria and Satisfaction of Objectives***

The Software Verification Process has several outputs:

- a) Review and analysis of the source code.
- b) The SVCP.
- c) Review and analysis of the test results.
- d) The software-test results themselves.

The Software Verification Process can transition to various other life cycle processes:

- 1) To the SQA Process upon successfully creating all Software Verification Process outputs.
- 2) To the Coding Process upon detection of errors in software testing.
- 3) To the Requirements Process or Design Process upon detection of errors more appropriately resolved in the SRD or SDD than in the code.

#### ***3.1.1.1.7. Software Configuration Management Process***

The Software Configuration Management Process (or just 'SCM Process') for the most part operates simultaneously with the other life cycle processes. DO-178B chapter 7 sets out the objectives and activities of the SCM Process in some detail. In summary, the SCM Process provides the following activities:

- 1) Identifying configurations.
- 2) Implementing change control.
- 3) Establishing baselines.
- 4) Archiving the software and the life cycle data.

Though for graphical reasons the figure at the top of the 'Life-Cycle' section of this document depicts the SCM Process as spanning merely the Requirements, Design, and Coding Processes, it actually spans all other life cycle processes, and beyond. Once data is archived by means of the SCM Process, it is theoretically intended to remain archived as long as the libBirds software is present within any airborne units.

Of course, since the Birds Project is not a manufacturer of airborne equipment, and the libBirds library is intended to be used by manufacturers of such equipment, it is not really within the capability of the Birds Project to guarantee this essentially indefinite data retention. It is therefore assumed that users of the libBirds library have prudently



archived the version of libBirds they are using -- i.e., all code and life-cycle data -- within the SCM Processes of their own development efforts.

#### *3.1.1.1.7.1. Details of Implementation*

TBD

#### *3.1.1.1.7.2. Transition Criteria and Satisfaction of Objectives*

The SCM Process does not transition to other life cycle processes, since it operates in parallel with such other processes.

Ultimately, the SCM Process exists to archive life cycle data, and to perpetually maintain this archive subject to the limitations described above. The outputs of the SCM Process are the SCI and the SCM Records demonstrating archival activity. The SECI, which may legitimately be viewed as an output of the SCM Process, is actually produced by the Planning Process, but may subsequently be altered by the SCM Process.

#### *3.1.1.1.8. Software Quality Assurance Process*

The objectives and activities of the Software Quality Assurance Process (or just 'SQA Process') are set out in chapter 8 of DO-178B. Basically, the SQA Process examines the outputs of the other life cycle processes and determines their internal consistency. In other words, it acts to assure that what has actually been accomplished is what was required to be accomplished.

Among the important activities of the SQA Process are these:

- 1) Establishment and management of the problem-reporting system.
- 2) Final release of the software.

The SQA Process operates simultaneously with the other life cycle processes, and spans the Planning Process through release of the software. Furthermore, the SQA Process is independent from the other life cycle processes, in the sense that separate personnel are involved and that the authority for the SQA Process is separate from the authority for the the other life cycle processes.

The SQA Process is the only life-cycle process required to have this characteristic of independence at the proposed software level ('C').

#### *3.1.1.1.8.1. Details of Implementation*

TBD

#### *3.1.1.1.8.2. Transition Criteria and Satisfaction of Objectives*

The SQA Process transitions to the Certification Liaison Process upon release of the software. The outputs of the SQA Process are the SQA Records.

The primary output of the SQA Process is the Software Conformity Review, which is the last step in the release of the software. The Software Conformity Review, however, takes into account not only the life cycle data in general, but also various other SQA Records.

#### *3.1.1.1.9. Certification Liaison*

Since the product of the libBirds development effort is a reusable library rather than an actual airborne device, there are really no certification efforts associated with it, and thus no real Certification Liaison Process.

However, since the intention of the libBirds project is to not merely provide software but to make it very conveniently usable by developers, a very useful certification activity would be review and approval by a DER. This "pre-approval" by a DER would allow immediate signoff of 8110 forms for developers using the libBirds library, very simply and relatively cheaply.

It is not known as this is written whether such DER activity will actually occur. Hence, it should be regarded as an optional activity that may be omitted.

#### *3.1.1.1.9.1. Details of Implementation*

TBD

#### *3.1.1.1.9.2. Transition Criteria and Satisfaction of Objectives*

The outputs of the Certification Liaison Process can be these:

- 1) Informal notification that a DER finds the life cycle data acceptable and will be available to sign off (via 8110) upon demand, for a known fee.
- 2) Notification of life-cycle data problems that will require repair before the DER finds the data acceptable.

Upon the former (option #1), the Certification Liaison Process ends, but there are no further processes to which a transition can occur. Of course, the availability of signoff would be published, so that libBirds users could be made aware of it.

Upon the latter (option #2), problem reports are filed concerning the problems which have been found. Also, a transition may (or may not) occur to an earlier life cycle process so that the problems can be fixed. The reason for this uncertainty of action is that while approval by the DER is a desirable result of the development effort, it is not a crucially necessary one from the point of view of the 'Birds Project'. These findings are often a matter of opinion, and a different DER might view the life cycle data differently.

## 4. Development Environment

### 4.1. Recognized Tools

Only free software tools are recognized as acceptable.

#### 4.1.1. Requirements Development Methods and Tools

The software application **Do178Builder** by Ronald Burkey is used in the Requirements Process. It is a documentation tool that helps to provide a structured approach towards developing the high-level requirements.

#### 4.1.2. Design Methods and Tools

The software application **Do178Builder** by Ronald Burkey is used in the Design Process. It is a documentation tool that helps to provide a structured approach towards deriving low-level requirements from high-level requirements.

#### 4.1.3. Programming Languages

The C programming language is used exclusively, for all code which will form a part of the libBirds library. The C++ language may be used for developing auxiliary programs

running on desktop computers but not forming parts of the libBirds library.

#### **4.1.4. Coding Tools**

Except where specified, refer to <http://www.gnu.org> for more information about all of the tools mentioned below.

The Red Hat **Source Navigator** program is used as an Integrated Development Environment (IDE) from which source-code may be edited and managed. Refer to <http://sources.redhat.com>.

The GNU **indent** program is used as a C-language source-code formatter.

The GNU **gcov** program is used for coverage analysis.

The **lclint** program is recognized as an acceptable static-code analyzer. Refer to <http://lclint.cs.virginia.edu/>.

The **cccc** program is recognized as an acceptable software-metric tool. Refer to <http://cccc.sourceforge.net/>. The GNU **wc** program is also acceptable, for getting simple line-counts and byte-counts.

The GNU **gdb** program is the debugger to be used. The GNU **ddd** program is the graphical front-end used with **gdb** (which by itself is a command-line oriented program). For Motorola 68K/Coldfire targets, a very desirable **gdb** patch is available from <ftp://www.cybertec.com.au/pub/bdm/> allowing use of the Motorola Background Debug Mode (BDM).

The GNU **cvs** program is used as a version-control system.

The GNU **make** program is used for managing software builds.

#### **4.1.5. Compilers**

The GNU **gcc** program is the only C compiler recognized. It is also used as a C++ compiler when required. Refer to <http://www.gnu.org> for more information.

The **gcc** compiler is available both as a native compiler for use when the target CPU is the same type as the CPU of the development system, and as a cross-compiler for use when the target CPU type differs from that of the development system. The compiler itself is provided by GNU as source code which can theoretically itself be compiled to support any given target/development CPU pairing. In practice this is somewhat difficult, and it is better to obtain a pre-built compiler in the desired configuration.

Here is a list of some recognized pre-built compiler configurations:

1. Native Intel 'x86 CPU. Usually provided automatically with any Intel-based Linux distribution.
2. Native PowerPC CPU. Usually provided automatically with any PowerPC-based Linux distribution.
3. Cross-compiler for Motorola 68K/Coldfire target from Intel 'x86 development platform. Available from <http://www.fiddes.net/coldfire>.
4. Cross-compiler for Motorola 68K/Coldfire target from PowerPC development platform. Available from <http://www.fiddes.net/coldfire>, but not as an executable -- it requires compiling.
5. Cross-compiler for ARM target from Intel 'x86 development platform. There are several distinct pre-built forms available. Here's one:  
<http://www.lart.tudelft.nl/lartware/compile-tools/>.

#### **4.1.6. Linkers**

The GNU linker **ld** is used. Like the GNU **gcc** compiler, the linker is available in many forms, depending on whether it is used natively or as a cross-linker.

The linker is always provided from the same sources as the compiler.

#### **4.1.7. Loaders**

The libBirds library depends on no loader as such, since it is a reusable library rather than a stand-alone system. For native use, such as debugging/testing on a desktop computer, the program-loader of the desktop operating system suffices. For debugging/testing code in a target environment, the debugging tool (**gdb**) is used to load the program into the target.

For actual use in a target environment, the developer is expected to provide his own loader program, not a part of libBirds.

## **4.2. Hardware Platforms and Operating Systems**

### **4.2.1. Source-Code Development**

Source-code may be developed on any platform providing the 'Recognized Tools'

listed earlier. These platforms include Microsoft Windows, UNIX, BSD, and Linux. However, it is **easier** to get some of these platforms set up than others. A Linux Intel-based platform is easiest to set up, and a Microsoft Windows platform is the hardest.

In practice, the libBirds development effort has used SuSE Linux PPC and Intel platforms interchangeably, and has not used the other platforms mentioned.

### **4.2.2. Documentation Development**

The **Do178Builder** documentation development tool can be used on any version of Linux, or on Microsoft Windows, BSD, UNIX, or (theoretically) Mac OS X. In practice, the libBirds development effort has used only SuSE Linux PPC and Intel, and Microsoft Windows.

## **5. GNU Free Documentation License**

Version 1.1, March 2000

Copyright (C) 2000 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.

### **0. PREAMBLE**

The purpose of this License is to make a manual, textbook, or other written 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. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

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

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to

software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

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. (For example, 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.

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 "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and 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 has been designed to thwart or discourage subsequent modification by readers is not Transparent. 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 designed for human

modification. Opaque formats include PostScript, PDF, 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 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.

## 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 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 publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added



material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. 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 less than five). 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", and 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. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, 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 as "Endorsements" or to conflict in title with any Invariant Section.

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.

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, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they 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 quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around 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 provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

## 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.