

## Parity Volume Set Specification 2.0

Michael Nahas

Peter Clements

Paul Nettle

Ryan Gallagher

May 11th, 2003

Revision History	
Revision 1.0	October 14th, 2001
Related specification, Inspiration	
Revision 2.0	May 11th, 2003
New Specification, Initial publication and formatting.	
Copyright © 2003 The Authors	Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

### Table of Contents

Introduction Conventions Description

Main packet File Description packet Input File Slice Checksum packet Recovery Slice packet Creator packet

Conclusion A. Optional PAR 2.0 Packets

Unicode Filename packet ASCII Comment packet Unicode Comment packet Input File Slice packet Recovery File Slice Checksum packet Packed Main packet Packed Recovery Slice packet B. How to Add an Application-Specific Packet Type C. GNU Free Documentation License

PREAMBLE APPLICABILITY AND DEFINITIONS VERBATIM COPYING COPYING IN QUANTITY MODIFICATIONS COMBINING DOCUMENTS COLLECTIONS OF DOCUMENTS AGGREGATION WITH INDEPENDENT WORKS TRANSLATION TERMINATION FUTURE REVISIONS OF THIS LICENSE ADDENDUM: How to use this License for your documents **Abstract**

Based on *Parity Volume Set Specification 1.0 [2001-10-14]* by Stefan Wehlus and others.

## Introduction

This document describes a file format for storing redundant data for a set of files.

In operation, a user will select a set of files from which the redundant data is to be made. These are known as *input files* and the set of them is known as the *recovery set*. The user will provide these to a program which generates file(s) that match the specification in this document. The program is known as a *PAR 2.0 Client* or *client* for short, and the generated files are known as *PAR 2.0 files* or *PAR files*. If the files in the recovery set ever get damaged (e.g. when they are transmitted or stored on a faulty disk) the client can read the damaged input files, read the (possibly damaged) PAR files, and regenerate the original input files. Of course, not all damages can be repaired, but many can.

A user can also name some input files that are not to be recovered if damaged. These input files are known as the *non-recovery set*. This feature is in the spec to keep the same functionality as PAR 1.0.

The redundant data in the PAR files is computed using Reed-Solomon codes. These codes can take a set of equal-sized blocks of data and produce a number of same-sized recovery blocks. Then, given a subset of original data blocks and some recovery block, it is possible to reproduce the original data blocks. Reed-Solomon codes can do this recovery as long as the number of missing data blocks does not out number the recovery blocks. The design of the Reed-Solomon codes in this spec is based on James S. Plank's tech report at U. of Tennessee entitled *A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems*. The tech report contains an error, so the design is changed slightly to fix the problem. PAR 2.0 uses a 16-bit Reed-Solomon code and can support 32768 blocks.

The equal-sized blocks for the Reed-Solomon codes come from *slices* of the input files in the recovery set. The slices are consecutive equal-sized chunks of each file. If a file does not fill out the chunk, i.e. *it ends mid-slice*, the rest of the slice is treated as if it is padded with zero bytes.

The PAR 2.0 file itself is made of *packets* - self-contained parts with their own checksum. This design prevents damage to one part of the file from making the whole file unusable.

Packets have a type and each type of packet serves a different purpose. One describes a file. Another contains the checksums of the slices in a file. Another states which files are in the recovery set and which files are in the non-recovery set. And yet another contains a *recovery slice* - the recovery data produced by the Reed-Solomon code.

A PAR 2.0 file is only required to contain 1 specific packet - the packet that identifies the type of client that created the file. This way, if clients are creating files that don't match the specification in some way, they can be tracked down.

The packets for a recovery set and non-recovery set can be broken into multiple files. Files can contain duplicate packets - in fact, this is recommended for vital packets, such as the ones that describe the input files and the one that states which files are in the recovery set. Packets can appear in any order in a file.

## Conventions

There are a number of conventions used in the design of this specification.

The data is 4-byte aligned. That is, every field starts on an index in the file which is congruent to zero, modulus 4. (I.e.,  $\text{address} \% 4 == 0$ ) This is because some memory systems function faster if 32-bit quantities are 4-byte aligned. It should be noted that a file could be corrupted (bytes inserted or deleted) to throw off the alignment. The spec is designed so that future versions could be 8-byte aligned.

All integers in this version of the spec are unsigned integers of either 4 or 8 bytes in length.

Strings are not null-terminated. This is to prevent hackers from using stack-overflow attacks. In order to make a string 4-byte aligned, 1 to 3 zero bytes may be appended. (2 bytes in the case of Unicode strings.) If an N-byte field contains an array, a null-terminated string can be created by copying the N-byte field into a character array of length N+1 and then setting the N+1 character to '\0'.

The lengths of arrays and strings are often implicit. For example, if a region is known to be 32 bytes and that region contains an 8-byte integer and a string, then the string is known to take up 24 bytes. The string is then at least 21 bytes in length, since the 24 bytes contains 0 to 3 bytes of null padding at the end.

All strings in the "core" spec are ASCII. This was chosen because Unicode is not sufficiently supported by tools. There exist optional portions of the spec that do support Unicode strings.

The lengths of files and parts of files are determined by 8-byte integers. This is to support OSes that can handle files longer than 4GB.

All integers are Intel-endian. (That is, little endian.)

The recovery set (and non-recovery set, if present) is identified by a 16-byte value known as the Recovery Set ID. Every part of the PAR file that affects a recovery set contains the recovery set ID. In this 2.0 version, the Recovery Set ID is calculated as an MD5 Hash of certain values. The way of calculating this value could change in future versions.

Files are also identified by a 16-byte value. In this 2.0 Version, it is an MD5 Hash of their name, length and the MD5 Hash of their first 16kB. The way of calculating this value could change in future versions.

Every byte of a PAR file is specified. There are no places to throw junk bytes that can be any value. Padding, where needed, is specified to be zero bytes. The order of items in all arrays is specified.

The specification is designed so that if two clients generate a packet with the same parameters, the packets are identical (except for client-identifying or client-specific packets). Thus, client writers can compare the output of their program against the reference implementation by comparing packets byte-for-byte.

## Description

A PAR 2.0 file consists of a sequence of "packets". A packet has a fixed sized header and a variable

length body. The packet header contains a checksum for the packet - if the packet is damaged, the packet is ignored. The packet header also contains a packet-type. If the client does not understand the packet type, the packet is ignored. To be compliant with this specification, a client must understand the "core" set of packets. Client may process the optional packets or create their own application-specific packets.

The packet header is:

**Table 87. Packet Header**

Length (bytes)	Type	Description
8	byte[8]	Magic sequence. Used to quickly identify location of packets. Value = {'P', 'A', 'R', '2', '\0', 'P', 'K', 'T'} (ASCII)
8	8-byte uint	Length of the entire packet. Must be multiple of 4. (NB: Includes length of header.)
16	MD5 Hash	MD5 Hash of packet. Used as a checksum for the packet. Calculation starts at first byte of Recovery Set ID and ends at last byte of body. <i>Does not include the magic sequence, length field or this field. NB: The MD5 Hash, by its definition, includes the length as if it were appended to the packet.</i>
16	MD5 Hash	Recovery Set ID. All packets that belong together have the same recovery set ID. (See "main packet" for how it is calculated.)
16	byte[16]	Type. Can be anything. All beginning "PAR " (ASCII) are reserved for specification-defined packets. Application-specific packets are recommended to begin with the ASCII name of the client.
?*4	?	Body of Packet. Must be a multiple of 4 bytes.

There are various types of packets. The "core" set of packets - the set of packets that all clients must recognize and process - are listed next. For each, the value for the "type" field will be listed along with the contents of the body of the packet.

### *Main packet*

The main packet contains the slice size and the File IDs of all the files in the recovery set. The MD5 hash of the *body* of the main packet is used as the Recovery Set ID, which is included in the packet header of every packet for the set. Clients reading a file should just test that the Recovery Set ID is the same in all packets and not check that it was calculated to the right value; the method for calculating the Recovery Set ID could change in future versions.

The main packet has a type value of "PAR 2.0\0Main\0\0\0" (ASCII). The packet's body contains the following:

**Table 126. Main packet**

Length (bytes)	Type	Description
8	8-byte uint	Slice size. Must be a multiple of 4.
4	4-byte uint	Number of files in the recovery set.
?*16	MD5 Hash array	File IDs of all files in the recovery set. (See File Description packet.) These hashes are <i>sorted</i> by numerical value (treating them as 16-byte unsigned integers).
?*16	MD5 Hash array	File IDs of all files in the non-recovery set. (See File Description packet.) These hashes are <i>sorted</i> by numerical value (treating them as 16-byte unsigned integers).

### *File Description packet*

A file description packet is included for each input file - recoverable or non-recoverable. The first thing in the packet is the File ID - this uniquely identifies a file in the PAR file. The packet then contains the MD5 hash, MD5 hash of the first 16kB of the file, file length, and the name of the file. The MD5 hash of the first 16kB of the file is included so that a client can identify a file if its name has been changed without the client reading the entire file. (Of course, that assumes the first 16kB hasn't been damaged or lost!)

The File ID in this version is calculated as the MD5 Hash of the last 3 fields of the body of

this packet: MD5-16k, length, and ASCII file name. Note: The length and MD5-16k are included because the Recovery Set ID is a hash of the File IDs and the Recovery Set ID should be a function of file contents as well as names.

File names are case sensitive and can be of any length. If a client is doing recovery on an operating system that has case-insensitive filenames or limited-length filenames, it is up to the client to rename files and directories. If the file's directory does not exist, the client must create it.

The file description packer has a type value of "PAR 2.0\0FileDesc" (ASCII). The packet's body contains the following:

**Table 158. File Descriptor Packet Body Contents**

Length (bytes)	Type	Description
16	MD5 Hash	The File ID.
16	MD5 Hash	The MD5 hash of the entire file.
16	MD5 Hash	The MD5-16k. That is, the MD5 hash of the first 16kB of the file.
8	8-byte uint	Length of the file.
?*4	ASCII char array	Name of the file. <i>This array is not guaranteed to be null terminated!</i> Subdirectories are indicated by an HTML-style '/' (a.k.a. the UNIX slash). <i>The filename must be unique.</i>

### *Input File Slice Checksum packet*

This packet type contains checksums for all the slices that are in an input file. It includes a CRC32 checksum to quickly locate the slices and an MD5 checksum to verify they have not been modified. The CRC32 is specified by CCITT and is the same as in Ethernet packets (and PKZIP, FDDI, etc.). If the file would end mid-slice, the remainder of the slice is filled with 0-value bytes.

The input file slice checksum packet has a type value of "PAR 2.0\0IFSC\0\0\0" (ASCII). The packet's body contains the following:

**Table 193. Slice Checksum Packet Body Contents**

Length (bytes)	Type	Description
16	MD5 Hash	The File ID of the file.

?*20	{MD5 Hash, CRC32} array	MD5 Hash and CRC32 pairs for the slices of the file. The Hash/CRC pairs are in the same order as their respective slices in the file. The Hash comes before the CRC in the array elements.
------	----------------------------	--

## *Recovery Slice packet*

The recovery slice packet contains one slice of recovery data. The recovery data is generated using a 16-bit Galois Field (GF) with generator 0x0001100B.

The algorithm for computing recovery slices is based on James S. Plank's tech report at U. of Tennessee entitled *A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems*. The input slices are ordered and assigned 16-bit constants. Recovery slices are assigned 16-bit exponents. Each 2-byte word of the recovery slice is the sum of the contributions from each input slice. The contribution of each input slice is the 2-byte word of the input slice multiplied by the input slice's constant raised to the recovery slice's exponent. All these computations (adds, multiplies, powers) are done using the 16-bit Galois Field operations.

To generate the recovery data, the slices of the input files are assigned constants. This is based on the order the File IDs appear in the main packet and then by the order the slices appear in the file. So the first slice of the first file in the main packet is assigned the first constant. The second slice of the first file is assigned the second constant. And so on. If the last slice of the first file has the Nth constant, the first slice of the second file is assigned the (N+1)th. And so on.

Here, the PAR 2.0 Spec diverges from Plank's paper. In Plank, the first constant is 1, the second 2, the third 3, etc. This is a bad approach because some constants have an order less than 65535. (That is, there exists constants N where N raised to a power less than 65535 is equal to 1 in the Galois Field.) These constants can prevent recovery matrices from being invertible and can, therefore, stop recovery. This spec does not use those constants. So, the first constant is the first power of two that has order 65535. The second constant is the next power of two that has order 65535. And so on. A power of two has order 65535 if the exponent is not equal to 0 modulus 3, 5, 17, or 257. In C code, that would be `(n%3 != 0 && n%5 != 0 && n%17 != 0 && n%257 != 0)`. Note - this is the *exponent* being tested, and not the constant itself. There are 32768 valid constants. The first few are:

- 2
- 4
- 16
- 128
- 256
- 2048
- 8192
- 16384
- 4107
- 32856
- 17132 ...

- The recovery slice packet has a type value of "PAR 2.0\0RecvSlic" (ASCII). The packet's body contains the following:

● **Table 232. Recovery Slice Packet Body Contents**

length (bytes)	● L	type	● T	description	● D
	● 4	-byte unit	● 4	exponent used to generate recovery data	● E
*4	● ?	byte array	● b	recovery data.	● R

● *Creator packet*

- This packet is used to identify the client that created the file. It is *required* to be in every PAR file. If a client is unable to process a recovery set, the contents of the creator packet *must* be shown to the user. The goal of this is that any client incompatibilities can be found and resolved quickly.

- The creator packet has a type value of "PAR 2.0\0Creator\0" (ASCII). The packet's body contains the following:

● **Table 254. Creator Packet Body Contents**

length (bytes)	● L	type	● T	description	● D
*4	● ?	SCII char array	● A	SCII text identifying the client. This should also include a way to contact the client's creator - either through a URL or an email address. NB: <i>This is not a null terminated string!</i>	● A

- Here ends the "core" packets that all clients must recognize and process.

● **Conclusion**

- That is the official spec. To make sure clients work similarly, the following client conventions should be

followed.

- PAR 2.0 files should always end in ".par2". For example, "file.par2". If a file contains recovery slices, the ".par2" should be preceded by ".volXX-YY" where XX to YY is the range of exponents for the recovery slices. For example, "file.vol20-29.par2". More than 2 digits should be used if necessary. Any exponents that contain fewer digits than the largest exponent should be preceded by zeros so that all filenames have the same length. For example, "file.vol075-149.par2". Exponents should start at 0 and go upwards.
- If multiple PAR files are generated, they may either have a constant number of slices per file (e.g. 20, 20, 20, ...) or exponentially increasing number of slices (e.g., 1, 2, 4, 8, ...). Note that to store 1023 slices takes 52 files if each has 20 slices, but takes only 10 files with the exponential pattern.
- When generating multiple PAR files, it is expected that one file be generated without any slices and containing all main, file description, and input file checksum packets. The other files should also include the main, file description and input file checksum packets. This repeats data that cannot be recovered.
- *NOTE: If the files are to be transmitted over usenet, it might be best to place the main, file description and input file checksum packets at the end, so that the equal-sized recovery slice packets are at the beginning. That way it may be possible to put a single recovery slice in each usenet message.*
- If just a single PAR file is generated, it is expected that the main, file description, and input file checksum packets are repeated multiple times and scattered through out the file. (Once again, repeating data that cannot be recovered.)
- Recall that all files *must* contain a creator packet.
- It is recommended that users are warned when they create PAR files with names that are incompatible with Windows, Mac, or Linux systems. That is, file or directory names that are more than 255 characters long, start with a period (.) or a dash (-), or contain one of these characters: < > : " ' ` ? \* & | [ ] \ ; or newline (\n).
- It is *strongly* recommended that clients query a user before writing to a file whose File Description packet contains an absolute pathname. For Windows, that means one starting with "C:\\" or "\\" for example. For UNIX, that means one starting with "/" or "//". For Mac, that means one starting with ":". This is to prevent PAR files of unknown origin from cracking a system by overwriting system files.

## ● A. Optional PAR 2.0 Packets

● Clients do not need to process these packets. They are included in this spec because many clients might want to implement the functionality and, if they did, it would be good if they were compatible with each other.

### ● *Unicode Filename packet*

● This packet provides an alternate name for a file. This packet *overrides* the default "ASCII" name in the file description packet.

- The Unicode filename packet has a type value of "PAR 2.0\0UniFileN" (ASCII, ironically). The packet's body contains the following:

● **Table A.7. Unicode Filename Packet Contents**

length (bytes)	● L	type	● T	description	● D
6	● 1	D5 Hash	● M	the File ID of the file.	● T
*4	● ?	Unicode char array	● U	the name of the file in Unicode. <i>NB: This is not a null terminated array!</i> This name must obey all the restrictions of the ASCII filename in the File Description packet.	● T

● **ASCII Comment packet**

- The ASCII comment packet contains - would you believe it - a comment in ASCII text! This should be shown to the user. If multiple copies of the same comment are found, only one need be shown.

- The ASCII comment packet has a type value of "PAR 2.0\0CommASCII" (ASCII). The packet's body contains the following

● **Table A.28. ASCII Comment Packet Contents**

length (bytes)	● L	type	● T	description	● D
*4	● ?	ASCII char array	● A	the comment. <i>NB: This is not a null terminated string!</i>	● T

● **Unicode Comment packet**

- The Unicode comment packet contains a comment in Unicode text. This should be shown to the user. If multiple copies of the same comment are found, only one need be shown. If an analogous ASCII version of the same comment is included in the file, the ASCII comment should not be shown.

- The Unicode comment packet has a type value of "PAR 2.0\0CommUni" (ASCII,

ironically). The packet's body contains the following:

● **Table A.45. Unicode Comment Packet Contents**

length (bytes)	● L	type	● T	description	● D
6	● 1	MD5 Hash or zeros	● M	If an ASCII comment packet exists in the file and is just a translation of the Unicode in this comment, this is the MD5 Hash of the ASCII comment packet. Otherwise, it is zeros.	● If
*4	● ?	Unicode char array	● U	The comment. NB: <i>This is not a null terminated string!</i>	● T

● ***Input File Slice packet***

● The input file slice packet is used if the user wants to include the input file inside the PAR file. This can be used to combine lots of small files into one file or to break a large file into smaller files (by distributing its slices into many PAR files). The length of the slice is determined by the slice size in the main packet, unless the slice would run off the end of the file. The packet contains an index for the slice and the slice contains bytes from (slice\_size\*index) to (slice\_size\*index + slice\_size - 1), unless the end of the file is reached. *NOTE: The indices are not the same as the input slice constants used in making recovery slices.*

● If files contain input slices, the ".par2" in the filename should be preceded by ".partXX-YY" where XX to YY is the indices of the slices. For example, "file.part00-09.par2" Indices are assigned to slices in the same order that constants were assigned in generating the recovery packets. *"... based on the order the File IDs appear in the main packet and then by the order the slices appear in the file. So the first slice of the first file in the main packet is assigned the first constant. The second slice of the first file is assigned the second constant. And so on. If the last slice of the first file has the Nth constant, the first slice of the second file is assigned the (N+1)th. And so on."*

● The input file slice packet has a type value of "PAR 2.0\0FileSlic". The packet's body contains the following:

● **Table A.69. Input File Slice Packet Contents**

length (bytes)	● L	type	● T	description	● D
----------------	-----	------	-----	-------------	-----

6	● 1	D5 Hash	● M	the File ID of the file.	● T
	● 8	-byte uint	● 8	the index of the slice. (See description above.)	● T
*4	● ?	byte array	● b	the slice. If the file ends mid-slice, the field is zero padded with 0 to 3 bytes to make it end on a 4-byte alignment.	● T

● *Recovery File Slice Checksum packet*

● So far, we've had input and recovery slices in the PAR file and input slices in an external file (i.e., the input file slice checksum packet). This packet covers the last combination - the recovery slices are in an external file (i.e., one where they don't have packet headers). This packet type may never be used, but it is included for completeness.

● There exists a file description packet for the file. The slices are generated the same as for the recovery slice packet.

● The recovery file slice checksum packet has a type value of "PAR 2.0\0RFSC\0\0\0\0". The packet's body contains the following:

● **Table A.94. Recovery File Slice Checksum Packet Contents**

length (bytes)	● L	type	● T	description	● D
6	● 1	D5 Hash	● M	the File ID of the file.	● T
*24	● ?	MD5 Hash, CRC32, 4-byte uint} array	● {	D5 Hash, CRC32, and exponent triplets for the slices in the file. The Hash/CRC/exponent triplets are in the same order as their respective slices in the file. The order in the array element is MD5 Hash, CRC32, followed by exponent.	● M

● *Packed Main packet*

- The packed main packet replaces the main packet when the client generates packed recover slice packets. The packed format allows recovery on units smaller than the slice size, which both increases the chance of recovery and allows more than 32,768 files. (The non-packed format requires, for each file, at least one slice and the number of slices is limited to 32,768.)

- The only difference from the main packet is the addition of the subslice size. (See the description of packed recovery slice packets to see how this is used.)

- The main packet has a type value of "PAR 2.0\0PkdMain\0" (ASCII). The packet's body contains the following:

● **Table A.115. Packed Main Packet Contents**

Length (bytes)	Type	Description
8	-byte uint	Subslice size. Must be a multiple of 4. Must equally divide the slice size.
8	-byte uint	Slice size. Must be a multiple of 4 and a multiple of the subslice size.
4	-byte uint	Number of files in the recovery set.
*16	D5 Hash array	File IDs of all files in the recovery set. (See File Description packet.) These hashes are <i>sorted</i> by numerical value (treating them as 16-byte unsigned integers).
*16	D5 Hash array	File IDs of all files in the non-recovery set. (See File Description packet.) These hashes are <i>sorted</i> by numerical value (treating them as 16-byte unsigned integers).

● *Packed Recovery Slice packet*

- The packed recovery slice packet contains one slice of recovery data. The recovery data is generated in the same manner as the recovery slice packet; the only thing that differs is how the data from the input slices is laid out.

- Files are broken into subslices and the subslices are ordered, just as in the recovery slice packet - sorted first by the order of the File ID in the packed main packet and then by the order of the subslice within the file.

- These subslices are then grouped together to make up the slices of input data used in the calculations. If X is the number of subslices in a slice, the first X subslices make up the first slice (which has the recovery constant 2). The next X subslices make up the second slice (which has the constant 4). Etc.

- In short, the input slices are made by packing the files together, with files starting on subslice boundaries rather than slice boundaries. Note that there are no subslice checksums, but there are file checksums, which can be used to detect bad regions that are smaller than a slice.

- The recovery slice packet has a type value of "PAR 2.0\0PkdRecvS" (ASCII). The packet's body contains the following:

● **Table A.152. Packed Recovery Slice Packet Contents**

length (bytes)	L	type	T	description	D
	● 4	-byte uint	● 4	xponent used to generate recovery data	● E
*4	● ?	byte array	● b	recovery data.	● R

- This ends the optional packets in this specification.

## ● B. How to Add an Application-Specific Packet Type

- Say the author of "NewsPost" wanted to add his own packet type - one that identified the names of the Usenet messages in which the files are posted. That author can create his own packet type. For example, here is the layout for one where the Usenet messages are identified by a newsgroup and a regular expression which all matches the names of the usenet articles.

- The packet has a type value of "NewsPostUsenet\0\0". (NB: Including the name of the client in the type is the recommended way to ensure unique type names.) The packet's body contains the following:

● **Table B.3. Example Application Specific Packet**

length (bytes) ● L	type ● T	description ● D
6 ● 1	MD5 Hash ● M	the File ID of the file. ● T
● 4	4-byte uint ● 4	the length of the string containing the name of the newsgroup. Must be a multiple of 4. ● T
*4 ● ?	SCII char array ● A	the name of the newsgroup. For example, "alt.binaries.multimedia". ● T
*4 ● ?	SCII char array ● A	regular expression matching the name of articles containing the file. For example, "Unaired Pilot - VCD,NTSC - (??/??)". ● A

● **C. GNU Free Documentation License**

● Copyright (C) 2000,2001,2002 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.

● ***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. 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.

## ● *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.

- 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.

## ● *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.

## ● *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.

## ● *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:

- 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.
- 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.
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- 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.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- 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.
- 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.
- 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.
- 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.

- Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- 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.

## ● *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".

## ● *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.

## ● *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.

## ● *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.

## ● *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.

## ● *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.

- *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 1.2 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.

<http://www.parfiles.net>