C# Coding Style Guide and Naming Guidelines.

Revision 1 (14/01/2003) – Carlos Guzmán Álvarez (carlosga@telefonica.net)
Revision 2 (15/01/2003) – Carlos Guzmán Álvarez (carlosga@telefonica.net)
Revision 3 (29/07/2003) – Carlos Guzmán Álvarez (carlosga@telefonica.net)

Note:
The Naming guidelines section is based on Microsoft C# Ecma documentation. Some things of the Coding Style section are based on Java conventions.

A. Coding Style

A.1 Source Files.

A.1.1 Beginning header.

All source files should begin with a c-style comment that lists the class name, version information, date, and copyright notice:

Example of license header using LGPL license:

```
/* One line for describe the project.
 * Copyright (C) Year  name of author
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */
```

A.2 Line Length

Consider avoiding (if possible) lines longer than 80 characters, switch on the ruler in your editor to get that managed, wrap lines if necessary.

A.3 Wrapping lines

When an expression will not fit on a single line, break it according to these general principles:

- Break after a comma.
- Break after an operator.
- Prefer higher-level breaks to low-level breaks.
- Align the new line with the beginning of the expression at the same level on the previous line.

Example of breaking method calls:

```
someMethod(longExpression1, longExpression2, longExpression3,
          longExpression4, longExpression5);

var = someMethod1(longExpression1,
                  someMethod2(longExpression2,
                     longExpression3));
```

Examples of breaking an arithmetic expression. The first is preferred, since the break occurs outside of the paranthesized expression (higher level rule).

```
       var =  a * b / (c - g + f) +
              4 * z; // PREFER

       var =  a * b / (c - g +
              f) + 4 * z;  // AVOID
```

A.4 Comments

A.4.1 Block Comments

Use the following style for block comments:

```
/* Line 1
 * Line 2
 * Line 3
 */
```

A.4.2 Single line comments

Use this style for end of line comments:

```
/* Comment Line */
```

or

```
// Comment Line
```

A.4.3 End of Line Comments

Use this style for end of line comments:

```
System.Int32 intValue; // Integer value
```

A.4.4 Documentation comments

Place documentation comments on separate XML files and use <include ... /> for make reference to the XML documentation file.

Example:

```
/// <include file='Documentation.xml'path='doc/member
[@name="T:Example"]/*'/>
public sealed class Example
{
    ...
}
```

A.5 Declarations

A.5.1 Number per line.

One declaration per line is recommended since it encourages commenting, example:

```
int       value;
string name;  // PREFER

int       value; string name;     // AVOID
```

A.5.2 Initialization

Try to initialize local variables where they're declared.

Example:

```
int       value  = 0;
string name   = String.Empty;
bool   flag   = false;
```

A.5.3 Placement

Put declarations only at the beginning of blocks.

Example:

```
void myMethod()
{
    int int1 = 0;           // beginning of method block
    if (condition)
    {
        int int2 = 0;    // beginning of "if" block
        ...
    }
}
```

A.5.4 Class and Interfaces

Use this style for end of line comments:

- No space between a method name and the parenthesis "(" starting its parameter list.
- Open brace "{"starts a line by itself indented to match its declaration statement.
- Closing brace "}" starts a line by itself indented to match its corresponding opening statement.

Example:

```
public class MyClass : MyBaseClass
{
    public MyClass()
    {
```

```
    }

    public string ReadData()
    {
        /* Access a custom resource. */
    }
}
```

Example of Interface:

```
/* C#
 * Code for the IAccount interface module.
 */
public interface IAccount
{
    void PostInterest();
    void DeductFees(IFeeSchedule feeSchedule);
}
```

A.6 Statements

A.6.1 Simple statements

Each line should contain at most one statement.

Example:

```
var1++;          // Correct
var2++;          // Correct
var1++; var2--;  // AVOID!
```

A.6.2 Return statements

A return statement with a value should not use parentheses unless they make the return value more obvious in some way.

Example:

```
return;

return var1;

return (var1 > var2 ? 0 : 1);
```

A.6.3 if statement

Use always the open and close bracket for if and else blocks, put an space between the if and the open parenthesis.

Examples:

```
if (x > 10)
{
    if (y > 20)
    {
        Console.WriteLine("Statement_1");
    }
    else
```

```
        {
            Console.WriteLine("Statement_2");
        }
    }

    if (Condition_1)
    {
        Console.WriteLine("Statement_1");
    }
    else
    {
        if (Condition_2)
        {
            Console.WriteLine("Statement_2");
        }
        else
        {
            if (Condition_3)
            {
                Console.WriteLine("Statement_1");
            }
        }
    }

    if (condition)   // AVOID! THIS OMITS THE BRACES {}!
        statement;
```

A.6.4 switch statement

A `switch` statement should have the following form:

```
    switch (condition)
    {
        case 0:
        case 1:
        {
            statements;
        }
        break;

        case 2:
        {
            statements;
        }
        break;

        case 3:
        {
            statements;
        }
        break;

        default:
        {
            statements;
        }
        break;
    }
```

A.6.5 for statement

A `for` statement should have the following form:

```
for ([initializers]; [expression]; [iterators])
{
    statements;
}
```

A.6.6 foreach statement

A `for` statement should have the following form:

```
foreach (type identifier  in expression)
{
    statements;
}
```

A.6.7 while and do...while statements

A `while` statement should have the following form:

```
while (expression)
{
    statements;
}
```

A `do` ... `while` statement should have the following form:

```
do
{
    statements;
} while (expression);
```

A.6.8 try...catch statements

A `try ... catch` statement should have the following form:

```
try
{
    statements;
}
catch (ExceptionClass ex)
{
    statements;
}
```

or

```
try
{
    statements;
}
catch (ExceptionClass ex)
{
    statements;
}
finally
{
```

```
        statements;
    }
```

A.7 White space

A.7.1 Blank lines

Blank lines improve readability by setting off sections of code that are logically related.

Two blank lines should always be used in the following circumstances:

- Between sections of a source file
- Between class and interface definitions

One blank line should always be used in the following circumstances:

- Between methods
- Between logical sections inside a method to improve readability

A.7.2 Blank spaces

Blank spaces should be used in the following circumstances:

- A keyword followed by a parenthesis should be separated by a space. Example:

  ```
  while (true)
  {
      ...
  }
  ```

  Note that a blank space should not be used between a method name and its opening parenthesis. This helps to distinguish keywords from method calls.

- A blank space should appear after commas in argument lists.
- All binary operators except **.** should be separated from their operands by spaces. Blank spaces should never separate unary operators such as unary minus, increment ("++"), and decrement ("--") from their operands. Example:

  ```
  a += c + d;
  a = (a + b) / (c * d);
  while (d++ == s++)
  {
      n++;
  }
  ```

- The expressions in a for statement should be separated by blank spaces. Example:

  ```
  for (expr1; expr2; expr3)
  ```

B. Naming guidelines

B.1 Capitalization styles

The following section describes different ways of capitalizing identifiers.

B.1.1 Pascal casing

This convention capitalizes the first character of each word. For example:

```
Color    BitConverter
```

B.1.2 Camel casing

This convention capitalizes the first character of each word except the first word. For example:

```
backgroundColor    totalValueCount
```

B.1.3 All uppercase

Only use all uppercase letters for an identifier if it contains an abbreviation. For example:

```
System.IO
System.WinForms.UI
```

B.1.4 Capitalization summary

The following table summarizes the capitalization style for the different kinds of identifiers:

Type	Case	Notes
Class	PascalCase	
Attribute Class	PascalCase	Has a suffix of Attribute
Exception Class	PascalCase	Has a suffix of Exception
Constant	PascalCase	
Enum type	PascalCase	
Enum values	PascalCase	
Event	PascalCase	
Interface	PascalCase	Has a prefix of **I**
Local variable	camelCase	
Method	PascalCase	
Namespace	PascalCase	
Property	PascalCase	
Public Instance Field	PascalCase	Rarely used (use a property instead)
Protected Instance Field	camelCase	Rarely used (use a property instead)
Parameter	camelCase	

B.2 Word choice

- Do avoid using class names duplicated in heavily used namespaces. For example, don't use the following for a class name.

```
System    Collections    Forms    UI
```

- Do not use abbreviations in identifiers.
- If you must use abbreviations, do use camelCase for any abbreviation containing more than two characters, even if this is not the usual abbreviation.

B.3 Namespaces

The general rule for namespace naming is: `CompanyName.TechnologyName`.

- Do avoid the possibility of two published namespaces having the same name, by prefixing namespace names with a company name or other well-established brand. For example, `Microsoft.Office` for the Office Automation classes provided by `Microsoft`.
- Do use **PascalCase**, and separate logical components with periods (as in `Microsoft.Office.PowerPoint`). If your brand employs non-traditional casing, do follow the casing defined by your brand, even if it deviates from normal namespace casing (for example, `NeXT.WebObjects`, and `ee.cummings`).
- Do use plural namespace names where appropriate. For example, use `System.Collections` rather than `System.Collection.Exceptions` to this rule are brand names and abbreviations. For example, use `System.IO not System.IOs`.
- Do not have namespaces and classes with the same name.

B.4 Classes

- Do name classes with nouns or noun phrases.
- Do use PascalCase.
- Do use sparingly, abbreviations in class names.
- Do not use any prefix (such as **"C"**, for example). Where possible, avoid starting with the letter **"I"**, since that is the recommended prefix for interface names. If you must start with that letter, make sure the second character is lowercase, as in IdentityStore.
- Do not use any underscores.

```
public class FileStream
{
    …
}
public class Button
{
    …
```

```
}
public class String
{
    …
}
```

B.5 Interfaces

- Do name interfaces with nouns or noun phrases, or adjectives describing behavior. For example, `IComponent` (descriptive noun), `ICustomAttributeProvider` (noun phrase), and `IPersistable` (adjective).
- Do use PascalCase.
- Do use sparingly, abbreviations in interface names.
- Do not use any underscores.
- Do prefix interface names with the letter **"I"**, to indicate that the type is an interface.
- Do use similar names when defining a class/interface pair where the class is a standard implementation of the interface. The names should differ only by the "I" prefix in the interface name. This approach is used for the interface IComponent and its standard implementation, Component.

```
public interface    IComponent
{
    …
}
public class Component : IComponent
{
    …
}
public interface    IserviceProvider
{
    …
}
public interface    IFormatable
{
    …
}
```

B.6 Enums

- Do use **PascalCase** for enums.
- Do use **PascalCase** for enum value names.
- Do use sparingly, abbreviations in enum names.
- Do not use a family-name prefix on enum.
- Do not use any "Enum" suffix on enum types.

- Do use a singular name for enums
- Do use a plural name for bit fields
- Do define enumerated values using an enum if they are used in a parameter or property. This gives development tools a chance at knowing the possible values for a property or parameter.

```
public enum FileMode
{
    Create,
    CreateNew,
    Open,
    OpenOrCreate,
    Truncate
}
```

- Do use the `Flags` custom attribute if the numeric values are meant to be bitwise ORed together

```
[Flags]
public enum Bindings
{
    CreateInstance,
    DefaultBinding,
    ExcatBinding,
    GetField,
    GetProperty,
    IgnoreCase,
    InvokeMethod,
    NonPublic,
    OABinding,
    SetField
    SetProperty,
    Static
}
```

- Do use int as the underlying type of an enum. (An exception to this rule is if the enum represents flags and there are more than 32 flags, or the enum may grow to that many flags in the future, or the type needs to be different from int for backward compatibility.)
- Do use enums only if the value can be completely expressed as a set of bit flags. Do not use enums for open sets (such as operating system version).

B.7 Static fields

- Do name static members with nouns, noun phrases, or abbreviations for nouns.
- Do name static members using **PascalCase**.
- Do not use Hungarian-type prefixes on static member names.

B.8 Parameters

- Do use descriptive names such that a parameter's name and type clearly imply its meaning.
- Do name parameters using **camelCase**.

- Do prefer names based on a parameter's meaning, to names based on the parameter's type. It is likely that development tools will provide the information about type in a convenient way, so the parameter name can be put to better use describing semantics rather than type.
- Do not reserve parameters for future use. If more data is need in the next version, a new overload can be added.
- Do not use Hungarian-type prefixes.

```
Type   GetType(string typeName)
String Format(string format, object[] args)
```

B.9 Methods

- Do name methods with verbs or verb phrases.
- Do name methods with **PascalCase**

```
RemoveAll(), GetCharArray(), Invoke()
```

B.10 Properties

- Do name properties using noun or noun phrases
- Do name properties with **PascalCase**.
- Consider having a property with the same as a type. When declaring a property with the same name as a type, also make the type of the property be that type. In other words, the following is okay

```
public enum Color
{
    ...
}
public class Control
{
    public Color Color
    {
       get {...}
       set {...}
    }
}
```

but this is not

```
   public enum Color
{
   //..
}
   public class Control
   {
      public int Color
      {
```

```
            get {...}
            set {...}
        }
    }
```

In the latter case, it will not be possible to refer to the members of the Color enum because `Color.Xxx` will be interpreted as being a member access that first gets the value of the Color property (of type `int`) and then accesses a member of that value (which would have to be an instance member of `System.Int32`).

B.11 Events

- Do name event handlers with the "EventHandler" suffix.

```
public delegate void MouseEventHandler(object sender, MouseEvent
e);
```

- Do use two parameters named `sender` and `e`. The sender parameter represents the object that raised the event, and this parameter is always of type **object**, even if it is possible to employ a more specific type. The state associated with the event is encapsulated in an instance **e** of an event class. Use an appropriate and specific event class for its type.

```
public delegate void MouseEventHandler(object sender, MouseEvent
e);
```

- Do name event argument classes with the "EventArgs" suffix.

```
public class MouseEventArgs : EventArgs
{
    int x;
    int y;
    public MouseEventArgs(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
 public int X
    {
        get { return x; }
    }
 public int Y
 {
     get { return y; }
    }
}
```

- Do name event names that have a concept of pre- and post-operation using the present and past tense (do not use **BeforeXxx/AfterXxx** pattern). For example, a close event that could be canceled would have a Closing and Closed event.

```
public event ControlEventHandler ControlAdded
{
   //..
}
```

- Consider naming events with a verb.

B.12 Case sensitivity

- Don't use names that require case sensitivity. Components might need to be usable from both case-sensitive and case-insensitive languages. Since case-insensitive languages cannot distinguish between two names within the same context that differ only by case, components must avoid this situation.

Examples of what not to do:

- Don't have two namespaces whose names differ only by case.

```
namespace ee.cummings;
namespace Ee.Cummings;
```

- Don't have a method with two parameters whose names differ only by case.

```
void F(string a, string A)
```

- Don't have a namespace with two types whose names differ only by case.

```
System.WinForms.Point p;
System.WinForms.POINT pp;
```

- Don't have a type with two properties whose names differ only by case.

```
int F {get, set};
int F {get, set}
```

- Don't have a type with two methods whose names differ only by case.

```
void f();
void F();
```

B.13 Avoiding type name confusion

Different languages use different names to identify the fundamental managed types, so in a multi-language environment, designers must take care to avoid language-specific terminology. This section describes a set of rules that help avoid type name confusion.

- Do use semantically interesting names rather than type names.
- In the rare case that a parameter has no semantic meaning beyond its type, use a generic name. For example, a class that supports writing a variety of data types into a stream might have:

```
void Write(double value);
void Write(float value);
void Write(long value);
void Write(int value);
void Write(short value);
```

rather than a language-specific alternative such as:

```
void Write(double doubleValue);
void Write(float floatValue);
void Write(long longValue);
```

```
void Write(int intValue);
void Write(short shortValue);
```

- In the extremely rare case that it is necessary to have a uniquely named method for each fundamental data type, do use the following universal type names: *Sbyte, Byte, Int16, UInt16, Int32, UInt32, Int64, UInt64, Single, Double, Boolean, Char, String, and Object*. For example, a class that supports reading a variety of data types from a stream might have:

```
Double ReadDouble();
float  ReadSingle();
long   ReadIn64();
int    ReadInt32();
short  ReadInt16();
```

rather than a language-specific alternative such as:

```
double ReadDouble();
float  ReadFloat();
long   ReadLong();
int    ReadInt();
short  ReadShort();
```

B.14GUI Controls prefixes

Prefix	Object Type
cbo	ComboBox
chk	CheckBox
btn	Button
clst	CheckedListBox
con	Connection
ctl	Control
com	Command
dta	DataAdapter
dtr	DataReader
dts	DataSet
dtv	DataView
dir	Directory List Box
dlg	Common dialog Control
drv	DriveList Box
edt	EditBox **(TextBox with Multiline = true)**
frm	Form
grb	GroupBox
grd	Grid/DataGrid
grc	Column
grh	Header
hsb	Horizontal ScrollBar
hpl	HyperLink Label
iml	ImageList
lbl	Label
lin	Line
lst	ListBox
mnu	Menu
pan	Panel
pic	PictureBox
pro	ProgressBar
prj	ProjectHook

sep	Separator
shp	Shape
spn	Spinner
tab	TabControl
pag	TabPage
txt	TextBox
tmr	Timer
tbl	ToolBar
trb	TrackBar
trv	TreeView
vsb	Vertical ScrollBar

C. Document License

GNU Free Documentation License

Version 1.2, November 2002

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.

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 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 Warrany 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 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.

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.