# Mini−XML Programmers Manual, Version 2.0

Michael Sweet

# Table of Contents

# Table of Contents

# Table of Contents

# Introduction

This programmers manual describes Mini−XML version 2.0, a small XML parsing library that you can use to read and write XML and XML−like data files in your application without requiring large non−standard libraries. Mini−XML only requires an ANSI C compatible compiler (GCC works, as do most vendors' ANSI C compilers) and a "make" program.

Mini−XML provides the following functionality:

- Reading of UTF−8 and UTF−16 encoded XML files and strings.
- Writing of UTF−8 encoded XML files and strings.
- Data is stored in a linked−list tree structure, preserving the XML data hierarchy.
- Supports arbitrary element names, attributes, and attribute values with no preset limits, just available memory.
- Supports integer, real, opaque ("cdata"), and text data types in "leaf" nodes.
- Functions for creating and managing trees of data.
- "Find" and "walk" functions for easily locating and navigating trees of data.

Mini−XML doesn't do validation or other types of processing on the data based upon schema files or other sources of definition information, nor does it support character entities other than those required by the XML specification.

# Legal Stuff



# History

Mini−XML was initially developed for the Gimp−Print project to replace the rather large and unwieldy `libxml2` library with something substantially smaller and easier−to−use. It all began one morning in June of 2003 when Robert posted the following sentence to the developer's list:

> It's bad enough that we require libxml2, but rolling our own XML parser is a bit more than we can handle.

I then replied with:

> Given the limited scope of what you use in XML, it should be trivial to code a mini−XML API in a few hundred lines of code.

I took my own challenge and coded furiously for two days to produced the initial public release of Mini−XML, total lines of code: 696. Robert promptly integrated Mini−XML into Gimp−Print and removed libxml2.

Thanks to lots of feedback and support from various developers, Mini−XML has evolved since then to provide a more complete XML implementation and now stands at a whopping 2,240 lines of code, compared to 96,335 lines of code for libxml2 version 2.6.9. Aside from Gimp−Print, Mini−XML is used for the following projects/software applications:

- Common UNIX Printing System
- CUPS Driver Development Kit
- ESP Print Pro
- ZynAddSubFX

Please email me (mxml @ easysw . com) if you would like your project added or removed from this list, or if you have any comments/quotes you would like me to publish about your experiences with Mini−XML.

## Organization of This Document

This manual is organized into the following chapters and appendices:

- Chapter 1, "Building, Installing, and Packaging Mini−XML", provides compilation, installation, and packaging instructions for Mini−XML.
- Chapter 2, "Getting Started with Mini−XML", shows how to use the Mini−XML library in your programs.
- Chapter 3, "More Mini−XML Programming Techniques", shows additional ways to use the Mini−XML library.
- Chapter 4, "Using the mxmldoc Utility", describes how to use the `mxmldoc(1)` program to generate software documentation.
- Appendix A, "GNU Library General Public License", provides the terms and conditions for using and distributing Mini−XML.
- Appendix B, "Release Notes", lists the changes in each release of Mini−XML.
- Appendix C, "Library Reference", contains a complete reference for Mini−XML, generated by `mxmldoc`.

## Notation Conventions

Various font and syntax conventions are used in this guide. Examples and their meanings and uses are explained below:

| Example | Description |
|---|---|
| `lpstat`<br>`lpstat(1)` | The names of commands; the first mention of a command or function in a chapter is followed by a manual page section number. |
| */var*<br>*/usr/share/cups/data/testprint.ps* | File and directory names. |
| `Request ID is Printer-123` | Screen output. |
| **`lp -d printer filename ENTER`** | Literal user input; special keys like **`ENTER`** are in ALL CAPS. |
| 12.3 | Numbers in the text are written using the period (.) to indicate the decimal point. |

# Abbreviations

The following abbreviations are used throughout this manual:

*Gb*
: Gigabytes, or 1073741824 bytes

*kb*
: Kilobytes, or 1024 bytes

*Mb*
: Megabytes, or 1048576 bytes

*UTF−8, UTF−16*
: Unicode Transformation Format, 8−bit or 16−bit

*W3C*
: World Wide Web Consortium

*XML*
: Extensible Markup Language

# Other References

*The Unicode Standard, Version 4.0, Addison−Wesley, ISBN 0−321−18578−1*
: The definition of the Unicode character set which is used for XML.

*Extensible Markup Language (XML) 1.0 (Third Edition)*
: The XML specification from the World Wide Web Consortium (W3C)

# 1 − Building, Installing, and Packaging Mini−XML

This chapter describes how to build, install, and package Mini−XML on your system.

## Compiling Mini−XML

Mini−XML comes with an autoconf−based configure script; just type the following command to get things going:

```
./configure ENTER
```

The default install prefix is */usr/local*, which can be overridden using the **`--prefix`** option:

```
./configure --prefix=/foo ENTER
```

Other configure options can be found using the **`--help`** option:

```
./configure --help ENTER
```

Once you have configured the software, use the `make(1)` program to do the build and run the test program to verify that things are working, as follows:

```
make ENTER
```

# Installing Mini−XML

Use the `make` command with the **`install`** target to install Mini−XML in the configured directories:

```
make install ENTER
```

# Creating Mini−XML Packages

Mini−XML includes two files that can be used to create binary packages. The first file is *mxml.spec* which is used by the `rpmbuild(8)` software to create Red Hat Package Manager ("RPM") packages which are commonly used on Linux. Since `rpmbuild` wants to compile the software on its own, you can provide it with the Mini−XML tar file to build the package:

```
rpmbuild −ta mxml−version.tar.gz ENTER
```

The second file is *mxml.list* which is used by the `epm(1)` program to create software packages in a variety of formats. The `epm` program is available from the following URL:

```
http://www.easysw.com/epm/
```

Use the `make` command with the **`epm`** target to create portable and native packages for your system:

```
make epm ENTER
```

The packages are stored in a subdirectory named *dist* for your convenience. The portable packages utilize scripts and tar files to install the software on the target system; this is especially useful when installing on systems with different Linux distributions. Use the *mxml.install* script to install the software and *mxml.remove* script to remove the software.

The native packages will be in the local OS's native format: RPM for Red Hat Linux, DPKG for Debian Linux, PKG for Solaris, and so forth. Use the corresponding commands to install the native packages.

# 2 – Getting Started with Mini−XML

This chapter describes how to write programs that use Mini−XML to access data in an XML file.

## The Basics

Mini−XML provides a single header file which you include:

```
#include <mxml.h>
```

The Mini−XML library is included with your program using the **−lmxml** option:

```
gcc −o myprogram myprogram.c −lmxml ENTER
```

If you have the `pkg−config(1)` software installed, you can use it to determine the proper compiler and linker options for your installation:

```
pkg−config −−cflags mxml ENTER
pkg−config −−libs mxml ENTER
```

## Nodes

Every piece of information in an XML file (elements, text, numbers) is stored in memory in "nodes". Nodes are defined by the `mxml_node_t` structure. The `type` member defines the node type (element, integer, opaque, real, or text) which determines which value you want to look at in the `value` union.

New nodes can be created using the `mxmlNewElement()`, `mxmlNewInteger()`, `mxmlNewOpaque()`, `mxmlNewReal()`, and `mxmlNewText()` functions. Only elements can have child nodes, and the top node must be an element, usually "?xml".

Each node has pointers for the node above (`parent`), below (`child`), to the left (`prev`), and to the right (`next`) of the current node. If you have an XML file like the following:

```
<?xml version="1.0"?>
<data>
    <node>val1</node>
    <node>val2</node>
    <node>val3</node>
    <group>
        <node>val4</node>
        <node>val5</node>
        <node>val6</node>
    </group>
    <node>val7</node>
    <node>val8</node>
    <node>val9</node>
</data>
```

the node tree returned by `mxmlLoadFile()` would look like the following in memory:

```
?xml
  |
data
  |
node - node - node - group - node - node - node
  |      |      |      |       |      |      |
val1   val2   val3     |     val7   val8   val9
                       |
                     node - node - node
                       |      |      |
                     val4   val5   val6
```

where "−" is a pointer to the next node and "|" is a pointer to the first child node.

Once you are done with the XML data, use the `mxmlDelete()` function to recursively free the memory that is used for a particular node or the entire tree:

```
mxmlDelete(tree);
```

# Loading and Saving XML Files

You load an XML file using the `mxmlLoadFile()` function:

```
FILE *fp;
mxml_node_t *tree;

fp = fopen("filename.xml", "r");
tree = mxmlLoadFile(NULL, fp, MXML_NO_CALLBACK);
fclose(fp);
```

The third argument specifies a callback function which returns the value type of the immediate children for a new element node: `MXML_INTEGER`, `MXML_OPAQUE`, `MXML_REAL`, or `MXML_TEXT`. This function is called *after* the element and its attributes have been read, so you can look at the element name, attributes, and attribute values to determine the proper value type to return. The default value type is MXML_TEXT if no callback is used.

Similarly, you save an XML file using the `mxmlSaveFile()` function:

```
FILE *fp;
mxml_node_t *tree;

fp = fopen("filename.xml", "w");
mxmlSaveFile(tree, fp, MXML_NO_CALLBACK);
fclose(fp);
```

Callback functions for saving are used to optionally insert whitespace before and after elements in the node tree. Your function will be called up to four times for each element node with a pointer to the node and a "where" value of `MXML_WS_BEFORE_OPEN`, `MXML_WS_AFTER_OPEN`, `MXML_WS_BEFORE_CLOSE`, or `MXML_WS_AFTER_CLOSE`. The callback function should return `NULL` if no whitespace should be added and the string to insert (spaces, tabs, carriage returns, and newlines) otherwise.

The `mxmlLoadString()`, `mxmlSaveAllocString()`, and `mxmlSaveString()` functions load XML node trees from and save XML node trees to strings:

```
char buffer[8192];
char *ptr;
mxml_node_t *tree;

...
tree = mxmlLoadString(NULL, buffer, MXML_NO_CALLBACK);

...
mxmlSaveString(tree, buffer, sizeof(buffer), MXML_NO_CALLBACK);

...
ptr = mxmlSaveAllocString(tree, MXML_NO_CALLBACK);
```

## Finding and Iterating Nodes

The `mxmlWalkPrev()` and `mxmlWalkNext()` functions can be used to iterate through the XML node tree:

```
mxml_node_t *node = mxmlWalkPrev(current, tree, MXML_DESCEND);

mxml_node_t *node = mxmlWalkNext(current, tree, MXML_DESCEND);
```

In addition, you can find a named element/node using the `mxmlFindElement()` function:

```
mxml_node_t *node = mxmlFindElement(tree, tree, "name", "attr",
                                    "value", MXML_DESCEND);
```

The `name`, `attr`, and `value` arguments can be passed as `NULL` to act as wildcards, e.g.:

```
/* Find the first "a" element */
node = mxmlFindElement(tree, tree, "a", NULL, NULL, MXML_DESCEND);

/* Find the first "a" element with "href" attribute */
```

```
node = mxmlFindElement(tree, tree, "a", "href", NULL, MXML_DESCEND);

/* Find the first "a" element with "href" to a URL */
node = mxmlFindElement(tree, tree, "a", "href",
                       "http://www.easysw.com/~mike/mxml/", MXML_DESCEND);

/* Find the first element with a "src" attribute*/
node = mxmlFindElement(tree, tree, NULL, "src", NULL, MXML_DESCEND);

/* Find the first element with a "src" = "foo.jpg" */
node = mxmlFindElement(tree, tree, NULL, "src", "foo.jpg", MXML_DESCEND);
```

You can also iterate with the same function:

```
mxml_node_t *node;

for (node = mxmlFindElement(tree, tree, "name", NULL, NULL, MXML_DESCEND);
     node != NULL;
     node = mxmlFindElement(node, tree, "name", NULL, NULL, MXML_DESCEND))
{
  ... do something ...
}
```

The `MXML_DESCEND` argument can actually be one of three constants:

- `MXML_NO_DESCEND` means to not to look at any child nodes in the element hierarchy, just look at siblings at the same level or parent nodes until the top node or top−of−tree is reached. The previous node from "group" would be the "node" element to the left, while the next node from "group" would be the "node" element to the right.
- `MXML_DESCEND_FIRST` means that it is OK to descend to the first child of a node, but not to descend further when searching. You'll normally use this when iterating through direct children of a parent node, e.g. all of the "node" elements under the "?xml" parent node in the example above. This mode is only applicable to the search function; the walk functions treat this as `MXML_DESCEND` since every call is a first time.
- `MXML_DESCEND` means to keep descending until you hit the bottom of the tree. The previous node from "group" would be the "val3" node and the next node would be the first node element under "group". If you were to walk from the root node "?xml" to the end of the tree with `mxmlWalkNext()`, the order would be:

```
?xml
data
node
val1
node
val2
node
val3
group
node
val4
node
val5
node
val6
node
val7
node
```

```
val8
node
val9
```

If you started at "val9" and walked using `mxmlWalkPrev()`, the order would be reversed, ending at "?xml".

# 3 − More Mini−XML Programming Techniques

This chapter shows additional ways to use the Mini−XML library in your programs.

## Load Callbacks

## Save Callbacks

## Changing Node Values

## Formatted Text

## Indexing

# 4 – Using the mxmldoc Utility

This chapter describes how to use the `mxmldoc(1)` utility that comes with Mini–XML to automatically generate documentation for your programs.

## The Basics

The `mxmldoc` utility scans C and C++ source and header files and produces an XML file describing the library interface and an XHTML file providing a human–readable reference to the code. Each source and header file must conform to some simple code commenting conventions so that `mxmldoc` can extract the necessary descriptive text.

The `mxmldoc` command requires the name of an XML file to store the code information; this file is created and updated as necessary. The XML file is optionally followed by a list of source files to scan. After scanning any source files on the command–line, `mxmldoc` writes XHTML documentation to the standard output, which can be redirected to the file using the **`>filename`** syntax:

```
mxmldoc myfile.xml >myfile.html ENTER
mxmldoc myfile.xml file1.c file2.cxx file3.h >myfile.html ENTER
```

If no source files are provided on the command–line, the current contents of the XML file are converted to XHTML.

# Code Documentation Conventions

As noted previously, source code must be commented properly for `mxmldoc` to generate correct documentation for the code. Single line comments can use the C++ `//` comment sequence, however all multi−line comments must use the C `/* ... */` comment sequence.

## Functions and Methods

All implementations of functions and methods must begin with a comment header describing what the function does, the possible input limits (if any), and the possible output values (if any), and any special information needed, as follows:

```
/*
 * 'do_this()' - Compute y = this(x).
 *
 * Notes: none.
 */

float            /* O - Inverse power value, 0.0 <= y <= 1.1 */
do_this(float x) /* I - Power value (0.0 <= x <= 1.1) */
{
  ...
  return (y);
}
```

Return/output values are indicated using an "O" prefix, input values are indicated using the "I" prefix, and values that are both input and output use the "IO" prefix for the corresponding in−line comment.

## Variables and Class/Structure/Union Members

Each variable or member must be declared on a separate line and must be immediately followed by a comment describing the variable or member, as follows:

```
int this_variable;   /* The current state of this */
int that_variable;   /* The current state of that */
```

## Types

Each type must have a comment block immediately before the typedef, as follows:

```
/*
 * This type is for foobar options.
 */
typedef int this_type_t;
```

## Classes, Structures, and Unions

Each class, structure, and union must have a comment block immediately before the definition, and each member must be documented in accordance with the function and variable documentation requirements, as follows:

```
/*
 * This structure is for foobar options.
 */
struct this_struct_s
{
  int this_member;   /* Current state for this */
  int that_member;   /* Current state for that */
};

/*
 * This class is for barfoo options.
 */
class this_class_c
{
  int this_member;   /* Current state for this */
  int that_member;   /* Current state for that */

  /*
   * 'get_this()' - Get the current state for this.
   */
  int                /* O - Current state for this */
  get_this()
  {
    return (this_member);
  }
};
```

## Enumerations

Each enumeration must have a comment block immediately before the definition describing what the enumeration is for, and each enumeration value must have a comment immediately after the value, as follows:

```
/*
 * Enumeration of media trays.
 */
 enum this_enum_e
 {
   THIS_TRAY,   /* This tray */
   THAT_TRAY    /* That tray */
 };
```

# XML Schema

Listing 4−1 shows the XML schema file *mxmldoc.xsd* which is included with Mini−XML. This schema file can be used to convert the XML files produced by `mxmldoc` into other formats.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Mini-XML 2.0 documentation schema for mxmldoc output.
      Copyright 2003-2004 by Michael Sweet.

      This program is free software; you can redistribute it and/or
      modify it under the terms of the GNU Library General Public
      License as published by the Free Software Foundation; either
      version 2, or (at your option) any later version.

      This program is distributed in the hope that it will be useful,
      but WITHOUT ANY WARRANTY; without even the implied warranty of
      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
      GNU General Public License for more details.
    </xsd:documentation>
  </xsd:annotation>

  <!-- basic element definitions -->
  <xsd:element name="argument" type="argumentType"/>
  <xsd:element name="class" type="classType"/>
  <xsd:element name="constant" type="constantType"/>
  <xsd:element name="description" type="xsd:string"/>
  <xsd:element name="enumeration" type="enumerationType"/>
  <xsd:element name="function" type="functionType"/>
  <xsd:element name="mxmldoc" type="mxmldocType"/>
  <xsd:element name="namespace" type="namespaceType"/>
  <xsd:element name="returnvalue" type="returnvalueType"/>
  <xsd:element name="seealso" type="identifierList"/>
  <xsd:element name="struct" type="structType"/>
  <xsd:element name="typedef" type="typedefType"/>
  <xsd:element name="type" type="xsd:string"/>
  <xsd:element name="union" type="unionType"/>
  <xsd:element name="variable" type="variableType"/>

  <!-- descriptions of complex elements -->
  <xsd:complexType name="argumentType">
    <xsd:sequence>
      <xsd:element ref="type" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="default" type="xsd:string" use="optional"/>
    <xsd:attribute name="name" type="identifier" use="required"/>
    <xsd:attribute name="direction" type="direction" use="optional" default="I"/>
  </xsd:complexType>

  <xsd:complexType name="classType">
    <xsd:sequence>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="class"/>
```

*Listing 4−1, XML Schema File "mxmldoc.xsd"*

```
        <xsd:element ref="enumeration"/>
        <xsd:element ref="function"/>
        <xsd:element ref="struct"/>
        <xsd:element ref="typedef"/>
        <xsd:element ref="union"/>
        <xsd:element ref="variable"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
    <xsd:attribute name="parent" type="xsd:string" use="optional"/>
  </xsd:complexType>

  <xsd:complexType name="constantType">
    <xsd:sequence>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="enumerationType">
    <xsd:sequence>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="constant" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="functionType">
    <xsd:sequence>
      <xsd:element ref="returnvalue" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="argument" minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element ref="seealso" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="identifier" use="required"/>
    <xsd:attribute name="scope" type="scope" use="optional"/>
  </xsd:complexType>

  <xsd:complexType name="mxmldocType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="class"/>
      <xsd:element ref="enumeration"/>
      <xsd:element ref="function"/>
      <xsd:element ref="namespace"/>
      <xsd:element ref="struct"/>
      <xsd:element ref="typedef"/>
      <xsd:element ref="union"/>
      <xsd:element ref="variable"/>
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="namespaceType">
    <xsd:sequence>
      <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="class"/>
        <xsd:element ref="enumeration"/>
        <xsd:element ref="function"/>
```

*Listing 4−1, XML Schema File "mxmldoc.xsd" (con't)*

```
      <xsd:element ref="struct"/>
      <xsd:element ref="typedef"/>
      <xsd:element ref="union"/>
      <xsd:element ref="variable"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="name" type="identifier" use="required"/>
</xsd:complexType>

<xsd:complexType name="returnvalueType">
  <xsd:sequence>
    <xsd:element ref="type" minOccurs="1" maxOccurs="1"/>
    <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="structType">
  <xsd:sequence>
    <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="variable"/>
      <xsd:element ref="function"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="name" type="identifier" use="required"/>
</xsd:complexType>

<xsd:complexType name="typedefType">
  <xsd:sequence>
    <xsd:element ref="type" minOccurs="1" maxOccurs="1"/>
    <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="identifier" use="required"/>
</xsd:complexType>

<xsd:complexType name="unionType">
  <xsd:sequence>
    <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="variable" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="identifier" use="required"/>
</xsd:complexType>

<xsd:complexType name="variableType">
  <xsd:sequence>
    <xsd:element ref="type" minOccurs="1" maxOccurs="1"/>
    <xsd:element ref="description" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="identifier" use="required"/>
</xsd:complexType>

<!-- data types -->
<xsd:simpleType name="direction">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="I"/>
    <xsd:enumeration value="O"/>
    <xsd:enumeration value="IO"/>
  </xsd:restriction>
```

*Listing 4−1, XML Schema File "mxmldoc.xsd" (con't)*

```
  </xsd:simpleType>

  <xsd:simpleType name="identifier">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[a-zA-Z_(.]([a-zA-Z_(.,)* 0-9])*"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="identifierList">
    <xsd:list itemType="identifier"/>
  </xsd:simpleType>

  <xsd:simpleType name="scope">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value=""/>
      <xsd:enumeration value="private"/>
      <xsd:enumeration value="protected"/>
      <xsd:enumeration value="public"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

*Listing 4−1, XML Schema File "mxmldoc.xsd" (con't)*

# A – GNU Library General Public License



## Preamble

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

This license, the Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

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

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

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non−free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non−free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the libary" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

# TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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

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

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

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

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

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

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

> ***a)*** The modified work must itself be a software library.
>
> ***b)*** You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
>
> ***c)*** You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
>
> ***d)*** If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs

whatever part of its purpose remains meaningful.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

> ***a)*** Accompany the work with the complete corresponding machine−readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine−readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
>
> ***b)*** Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
>
> ***c)*** If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
>
> ***d)*** Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

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

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

Library together in an executable that you distribute.

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

> ***a)*** Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
>
> ***b)*** Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

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

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

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

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

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

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

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

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

***13.*** The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

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

## NO WARRANTY

***15.*** BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

***16.*** IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## END OF TERMS AND CONDITIONS

# B − Release Notes

## Changes in Mini−XML 2.0

- New programmers manual.
- Added UTF−16 support (input only; all output is UTF−8)
- Added index functions to build a searchable index of XML nodes.
- Added character entity callback interface to support additional character entities beyond those defined in the XHTML specification.
- Added support for XHTML character entities.
- The mxmldoc utility now produces XML output which conforms to an updated XML schema, described in the file "doc/mxmldoc.xsd".
- Changed the whitespace callback interface to return strings instead of a single character, allowing for greater control over the formatting of XML files written using Mini−XML. THIS CHANGE WILL REQUIRE CHANGES TO YOUR 1.x CODE IF YOU USE WHITESPACE CALLBACKS.
- The mxmldoc utility now produces XML output which conforms to an updated XML schema, described in the file "doc/mxmldoc.xsd".
- Changed the whitespace callback interface to return strings instead of a single character, allowing for greater control over the formatting of XML files written using Mini−XML. THIS CHANGE WILL REQUIRE CHANGES TO YOUR 1.x CODE IF YOU USE WHITESPACE CALLBACKS.
- The mxmldoc utility is now capable of documenting C++ classes, functions, and structures, and correctly handles C++ comments.
- Added new modular tests for mxmldoc.
- Updated the mxmldoc output to be more compatible with embedding in manuals produced with HTMLDOC.

- The makefile incorrectly included a "/" separator between the destination path and install path. This caused problems when building and installing with MingW.

## Changes in Mini−XML 1.3

- Fixes for mxmldoc.
- Added support for reading standard HTML entity names.
- mxmlLoadString/File() did not decode character entities in element names, attribute names, or attribute values.
- mxmlLoadString/File() would crash when loading non− conformant XML data under an existing parent (top) node.
- Fixed several bugs in the mxmldoc utility.
- Added new error callback function to catch a variety of errors and log them to someplace other than stderr.
- The mxmlElementSetAttr() function now allows for NULL attribute values.
- The load and save functions now properly handle quoted element and attribute name strings properly, e.g. for !DOCTYPE declarations.

## Changes in Mini−XML 1.2

- Added new "set" methods to set the value of a node.
- Added new formatted text methods mxmlNewTextf() and mxmlSetTextf() to create/set a text node value using printf−style formats.
- Added new standard callbacks for use with the mxmlLoad functions.
- Updated the HTML documentation to include examples of the walk and load function output.
- Added −−with/without−ansi configure option to control the strdup() function check.
- Added −−with/without−snprintf configure option to control the snprintf() and vsnprintf() function checks.

## Changes in Mini−XML 1.1.2

- The mxml(3) man page wasn't updated for the string functions.
- mxmlSaveString() returned the wrong number of characters.
- mxml_add_char() updated the buffer pointer in the wrong place.

## Changes in Mini−XML 1.1.1

- The private mxml_add_ch() function did not update the start−of−buffer pointer which could cause a crash when using mxmlSaveString().
- The private mxml_write_ws() function called putc() instead of using the proper callback which could cause a crash when using mxmlSaveString().
- Added a mxmlSaveAllocString() convenience function for saving an XML node tree to an allocated string.

## Changes in Mini−XML 1.1

- The mxmlLoadFile() function now uses dynamically allocated string buffers for element names, attribute names, and attribute values. Previously they were capped at 16383, 255, and 255 bytes, respectively.

- Added a new mxmlLoadString() function for loading an XML node tree from a string.
- Added a new mxmlSaveString() function for saving an XML node tree to a string.
- Add emulation of strdup() if the local platform does not provide the function.

# Changes in Mini−XML 1.0

- The mxmldoc program now handles function arguments, structures, unions, enumerations, classes, and typedefs properly.
- Documentation provided via mxmldoc and more in−line comments in the code.
- Added man pages and packaging files.

# Changes in Mini−XML 0.93

- New mxmldoc example program that is also used to create and update code documentation using XML and produce HTML reference pages.
- Added mxmlAdd() and mxmlRemove() functions to add and remove nodes from a tree. This provides more flexibility over where the nodes are inserted and allows nodes to be moved within the tree as needed.
- mxmlLoadFile() now correctly handles comments.
- mxmlLoadFile() now supports the required "gt", "quot", and "nbsp" character entities.
- mxmlSaveFile() now uses newlines as whitespace when valid to do so.
- mxmlFindElement() now also takes attribute name and attribute value string arguments to limit the search to specific elements with attributes and/or values.

  NULL pointers can be used as "wildcards".
- Added uninstall target to makefile, and auto−reconfig if Makefile.in or configure.in are changed.
- mxmlFindElement(), mxmlWalkNext(), and mxmlWalkPrev() now all provide "descend" arguments to control whether they descend into child nodes in the tree.
- Fixed some whitespace issues in mxmlLoadFile().
- Fixed Unicode output and whitespace issues in mxmlSaveFile().
- mxmlSaveFile() now supports a whitespace callback to provide more human−readable XML output under program control.

# Changes in Mini−XML 0.92

- mxmlSaveFile() didn't return a value on success.

# Changes in Mini−XML 0.91

- mxmlWalkNext() would go into an infinite loop.

# Changes in Mini−XML 0.9

- Initial public release.

# C – Library Reference

## Contents

- Enumerations
- Functions
- Structures
- Types
- Unions
- Variables

# Enumerations

- `mxml_type_e`

## mxml_type_e

### Description

The XML node type.

### Values

| Name | Description |
| --- | --- |
| `MXML_ELEMENT` | XML element with attributes |
| `MXML_INTEGER` | Integer value |
| `MXML_OPAQUE` | Opaque string |
| `MXML_REAL` | Real value |
| `MXML_TEXT` | Text fragment |

# Functions

- `mxmlAdd()`
- `mxmlDelete()`
- `mxmlElementGetAttr()`
- `mxmlElementSetAttr()`
- `mxmlEntityAddCallback()`
- `mxmlEntityGetName()`
- `mxmlEntityGetValue()`
- `mxmlEntityRemoveCallback()`
- `mxmlFindElement()`
- `mxmlIndexDelete()`
- `mxmlIndexEnum()`
- `mxmlIndexFind()`
- `mxmlIndexNew()`
- `mxmlIndexReset()`
- `mxmlLoadFile()`
- `mxmlLoadString()`
- `mxmlNewElement()`
- `mxmlNewInteger()`
- `mxmlNewOpaque()`
- `mxmlNewReal()`
- `mxmlNewText()`
- `mxmlNewTextf()`
- `mxmlRemove()`
- `mxmlSaveAllocString()`
- `mxmlSaveFile()`
- `mxmlSaveString()`
- `mxmlSetElement()`
- `mxmlSetErrorCallback()`
- `mxmlSetInteger()`
- `mxmlSetOpaque()`
- `mxmlSetReal()`
- `mxmlSetText()`
- `mxmlSetTextf()`
- `mxmlWalkNext()`
- `mxmlWalkPrev()`

## mxmlAdd()

### Description

Add a node to a tree. Adds the specified node to the parent. If the child argument is not NULL, puts the new node before or after the specified child depending on the value of the where argument. If the child argument is NULL, puts the new node at the beginning of the child list (MXML_ADD_BEFORE) or at the end of the child list (MXML_ADD_AFTER). The constant MXML_ADD_TO_PARENT can be used to specify a NULL child pointer.

### Syntax

```
void
mxmlAdd(
    mxml_node_t * parent,
    int where,
    mxml_node_t * child,
    mxml_node_t * node);
```

### Arguments

| Name | Description |
|---|---|
| parent | Parent node |
| where | Where to add, MXML_ADD_BEFORE or MXML_ADD_AFTER |
| child | Child node for where or MXML_ADD_TO_PARENT |
| node | Node to add |

### Returns

Nothing.

## mxmlDelete()

### Description

Delete a node and all of its children. If the specified node has a parent, this function first removes the node from its parent using the mxmlRemove() function.

### Syntax

```
void
mxmlDelete(
    mxml_node_t * node);
```

### Arguments

| Name | Description |
|---|---|
| node | Node to delete |

### Returns

Nothing.

## mxmlElementGetAttr()

### Description

Get an attribute. This function returns NULL if the node is not an element or the named attribute does not exist.

### Syntax

```
const char *
mxmlElementGetAttr(
    mxml_node_t * node,
    const char * name);
```

### Arguments

| Name | Description |
|---|---|
| node | Element node |
| name | Name of attribute |

### Returns

Attribute value or NULL

## mxmlElementSetAttr()

### Description

Set an attribute. If the named attribute already exists, the value of the attribute is replaced by the new string value. The string value is copied into the element node. This function does nothing if the node is not an element.

### Syntax

```
void
mxmlElementSetAttr(
    mxml_node_t * node,
    const char * name,
    const char * value);
```

### Arguments

| Name | Description |
|---|---|
| node | Element node |
| name | Name of attribute |
| value | Attribute value |

### Returns

Nothing.

## mxmlEntityAddCallback()

### Description

Add a callback to convert entities to Unicode.

### Syntax

```
void
mxmlEntityAddCallback(
    int (*cb)(const char *name));
```

### Arguments

| Name | Description |
| --- | --- |
| `(*cb)(const char *name)` | Callback function to add |

### Returns

Nothing.

## mxmlEntityGetName()

### Description

Get the name that corresponds to the character value. If val does not need to be represented by a named entity, NULL is returned.

### Syntax

```
const char *
mxmlEntityGetName(
    int val);
```

### Arguments

| Name | Description |
|---|---|
| `val` | Character value |

### Returns

Entity name or NULL

## mxmlEntityGetValue()

### Description

Get the character corresponding to a named entity. The entity name can also be a numeric constant. −1 is returned if the name is not known.

### Syntax

```
int
mxmlEntityGetValue(
    const char * name);
```

### Arguments

| Name | Description |
|---|---|
| name | Entity name |

### Returns

Character value or −1 on error

## mxmlEntityRemoveCallback()

### Description

Remove a callback.

### Syntax

```
void
mxmlEntityRemoveCallback(
    int (*cb)(const char *name));
```

### Arguments

| Name | Description |
| --- | --- |
| (*cb)(const char *name) | Callback function to remove |

### Returns

Nothing.

## mxmlFindElement()

### Description

Find the named element. The search is constrained by the name, attribute name, and value; any NULL names or values are treated as wildcards, so different kinds of searches can be implemented by looking for all elements of a given name or all elements with a specific attribute. The descend argument determines whether the search descends into child nodes; normally you will use MXML_DESCEND_FIRST for the initial search and MXML_NO_DESCEND to find additional direct descendents of the node. The top node argument constrains the search to a particular node's children.

### Syntax

```
mxml_node_t *
mxmlFindElement(
    mxml_node_t * node,
    mxml_node_t * top,
    const char * name,
    const char * attr,
    const char * value,
    int descend);
```

### Arguments

| Name | Description |
|---|---|
| `node` | Current node |
| `top` | Top node |
| `name` | Element name or NULL for any |
| `attr` | Attribute name, or NULL for none |
| `value` | Attribute value, or NULL for any |
| `descend` | Descend into tree − MXML_DESCEND, MXML_NO_DESCEND, or MXML_DESCEND_FIRST |

### Returns

Element node or NULL

## mxmlIndexDelete()

### Description

Delete an index.

### Syntax

```
void
mxmlIndexDelete(
    mxml_index_t * ind);
```

### Arguments

| Name | Description |
| --- | --- |
| ind | Index to delete |

### Returns

Nothing.

## mxmlIndexEnum()

### Description

Return the next node in the index. Nodes are returned in the sorted order of the index.

### Syntax

```
mxml_node_t *
mxmlIndexEnum(
    mxml_index_t * ind);
```

### Arguments

| Name | Description |
|---|---|
| ind | Index to enumerate |

### Returns

Next node or NULL if there is none

## mxmlIndexFind()

### Description

Find the next matching node. You should call mxmlIndexReset() prior to using this function for the first time with a particular set of "element" and "value" strings. Passing NULL for both "element" and "value" is equivalent to calling mxmlIndexEnum().

### Syntax

```
mxml_node_t *
mxmlIndexFind(
    mxml_index_t * ind,
    const char * element,
    const char * value);
```

### Arguments

| Name | Description |
| --- | --- |
| `ind` | Index to search |
| `element` | Element name to find, if any |
| `value` | Attribute value, if any |

### Returns

Node or NULL if none found

# mxmlIndexNew()

## Description

Create a new index. The index will contain all nodes that contain the named element and/or attribute. If both "element" and "attr" are NULL, then the index will contain a sorted list of the elements in the node tree. Nodes are sorted by element name and optionally by attribute value if the "attr" argument is not NULL.

## Syntax

```
mxml_index_t *
mxmlIndexNew(
    mxml_node_t * node,
    const char * element,
    const char * attr);
```

## Arguments

| Name | Description |
|---|---|
| `node` | XML node tree |
| `element` | Element to index or NULL for all |
| `attr` | Attribute to index or NULL for none |

## Returns

New index

## mxmlIndexReset()

### Description

Reset the enumeration/find pointer in the index and return the first node in the index. This function should be called prior to using mxmlIndexEnum() or mxmlIndexFind() for the first time.

### Syntax

```
mxml_node_t *
mxmlIndexReset(
    mxml_index_t * ind);
```

### Arguments

| Name | Description |
|---|---|
| `ind` | Index to reset |

### Returns

First node or NULL if there is none

## mxmlLoadFile()

### Description

Load a file into an XML node tree. The nodes in the specified file are added to the specified top node. If no top node is provided, the XML file MUST be well−formed with a single parent node like <?xml> for the entire file. The callback function returns the value type that should be used for child nodes. If MXML_NO_CALLBACK is specified then all child nodes will be either MXML_ELEMENT or MXML_TEXT nodes. The constants MXML_INTEGER_CALLBACK, MXML_OPAQUE_CALLBACK, MXML_REAL_CALLBACK, and MXML_TEXT_CALLBACK are defined for loading child nodes of the specified type.

### Syntax

```
mxml_node_t *
mxmlLoadFile(
    mxml_node_t * top,
    FILE * fp,
    mxml_type_t (*cb)(mxml_node_t *node));
```

### Arguments

| Name | Description |
|---|---|
| `top` | Top node |
| `fp` | File to read from |
| `(*cb)(mxml_node_t *node)` | Callback function or MXML_NO_CALLBACK |

### Returns

First node or NULL if the file could not be read.

## mxmlLoadString()

### Description

Load a string into an XML node tree. The nodes in the specified string are added to the specified top node. If no top node is provided, the XML string MUST be well−formed with a single parent node like <?xml> for the entire string. The callback function returns the value type that should be used for child nodes. If MXML_NO_CALLBACK is specified then all child nodes will be either MXML_ELEMENT or MXML_TEXT nodes. The constants MXML_INTEGER_CALLBACK, MXML_OPAQUE_CALLBACK, MXML_REAL_CALLBACK, and MXML_TEXT_CALLBACK are defined for loading child nodes of the specified type.

### Syntax

```
mxml_node_t *
mxmlLoadString(
    mxml_node_t * top,
    const char * s,
    mxml_type_t (*cb)(mxml_node_t *node));
```

### Arguments

| Name | Description |
| --- | --- |
| `top` | Top node |
| `s` | String to load |
| `(*cb)(mxml_node_t *node)` | Callback function or MXML_NO_CALLBACK |

### Returns

First node or NULL if the string has errors.

## mxmlNewElement()

### Description

Create a new element node. The new element node is added to the end of the specified parent's child list. The constant MXML_NO_PARENT can be used to specify that the new element node has no parent.

### Syntax

```
mxml_node_t *
mxmlNewElement(
    mxml_node_t * parent,
    const char * name);
```

### Arguments

| Name | Description |
|---|---|
| parent | Parent node or MXML_NO_PARENT |
| name | Name of element |

### Returns

New node

## mxmlNewInteger()

### Description

Create a new integer node. The new integer node is added to the end of the specified parent's child list. The constant MXML_NO_PARENT can be used to specify that the new integer node has no parent.

### Syntax

```
mxml_node_t *
mxmlNewInteger(
    mxml_node_t * parent,
    int integer);
```

### Arguments

| Name | Description |
|---|---|
| `parent` | Parent node or MXML_NO_PARENT |
| `integer` | Integer value |

### Returns

New node

## mxmlNewOpaque()

### Description

Create a new opaque string. The new opaque node is added to the end of the specified parent's child list. The constant MXML_NO_PARENT can be used to specify that the new opaque node has no parent. The opaque string must be nul−terminated and is copied into the new node.

### Syntax

```
mxml_node_t *
mxmlNewOpaque(
    mxml_node_t * parent,
    const char * opaque);
```

### Arguments

| Name | Description |
| --- | --- |
| parent | Parent node or MXML_NO_PARENT |
| opaque | Opaque string |

### Returns

New node

## mxmlNewReal()

### Description

Create a new real number node. The new real number node is added to the end of the specified parent's child list. The constant MXML_NO_PARENT can be used to specify that the new real number node has no parent.

### Syntax

```
mxml_node_t *
mxmlNewReal(
    mxml_node_t * parent,
    double real);
```

### Arguments

| Name | Description |
|---|---|
| `parent` | Parent node or MXML_NO_PARENT |
| `real` | Real number value |

### Returns

New node

## mxmlNewText()

### Description

Create a new text fragment node. The new text node is added to the end of the specified parent's child list. The constant MXML_NO_PARENT can be used to specify that the new text node has no parent. The whitespace parameter is used to specify whether leading whitespace is present before the node. The text string must be nul−terminated and is copied into the new node.

### Syntax

```
mxml_node_t *
mxmlNewText(
    mxml_node_t * parent,
    int whitespace,
    const char * string);
```

### Arguments

| Name | Description |
|---|---|
| `parent` | Parent node or MXML_NO_PARENT |
| `whitespace` | 1 = leading whitespace, 0 = no whitespace |
| `string` | String |

### Returns

New node

## mxmlNewTextf()

### Description

Create a new formatted text fragment node. The new text node is added to the end of the specified parent's child list. The constant MXML_NO_PARENT can be used to specify that the new text node has no parent. The whitespace parameter is used to specify whether leading whitespace is present before the node. The format string must be nul−terminated and is formatted into the new node.

### Syntax

```
mxml_node_t *
mxmlNewTextf(
    mxml_node_t * parent,
    int whitespace,
    const char * format,
    ...);
```

### Arguments

| Name | Description |
|---|---|
| `parent` | Parent node or MXML_NO_PARENT |
| `whitespace` | 1 = leading whitespace, 0 = no whitespace |
| `format` | Printf−style frmat string |
| `...` | Additional args as needed |

### Returns

New node

## mxmlRemove()

### Description

Remove a node from its parent. Does not free memory used by the node − use mxmlDelete() for that. This function does nothing if the node has no parent.

### Syntax

```
void
mxmlRemove(
    mxml_node_t * node);
```

### Arguments

| Name | Description |
|---|---|
| node | Node to remove |

### Returns

Nothing.

## mxmlSaveAllocString()

### Description

Save an XML node tree to an allocated string. This function returns a pointer to a string containing the textual representation of the XML node tree. The string should be freed using the free() function when you are done with it. NULL is returned if the node would produce an empty string or if the string cannot be allocated.

### Syntax

```
char *
mxmlSaveAllocString(
    mxml_node_t * node,
    const char * (*cb)(mxml_node_t *node, int ws));
```

### Arguments

| Name | Description |
| --- | --- |
| node | Node to write |
| (*cb)(mxml_node_t *node, int ws) | Whitespace callback or MXML_NO_CALLBACK |

### Returns

Allocated string or NULL

## mxmlSaveFile()

### Description

Save an XML tree to a file. The callback argument specifies a function that returns a whitespace character or nul (0) before and after each element. If MXML_NO_CALLBACK is specified, whitespace will only be added before MXML_TEXT nodes with leading whitespace and before attribute names inside opening element tags.

### Syntax

```
int
mxmlSaveFile(
    mxml_node_t * node,
    FILE * fp,
    const char * (*cb)(mxml_node_t *node, int ws));
```

### Arguments

| Name | Description |
|---|---|
| node | Node to write |
| fp | File to write to |
| (*cb)(mxml_node_t *node, int ws) | Whitespace callback or MXML_NO_CALLBACK |

### Returns

0 on success, −1 on error.

## mxmlSaveString()

### Description

Save an XML node tree to a string. This function returns the total number of bytes that would be required for the string but only copies (bufsize − 1) characters into the specified buffer.

### Syntax

```
int
mxmlSaveString(
    mxml_node_t * node,
    char * buffer,
    int bufsize,
    const char * (*cb)(mxml_node_t *node, int ws));
```

### Arguments

| Name | Description |
|---|---|
| `node` | Node to write |
| `buffer` | String buffer |
| `bufsize` | Size of string buffer |
| `(*cb)(mxml_node_t *node, int ws)` | Whitespace callback or MXML_NO_CALLBACK |

### Returns

Size of string

## mxmlSetElement()

### Description

Set the name of an element node. The node is not changed if it is not an element node.

### Syntax

```
int
mxmlSetElement(
    mxml_node_t * node,
    const char * name);
```

### Arguments

| Name | Description |
|---|---|
| `node` | Node to set |
| `name` | New name string |

### Returns

0 on success, −1 on failure

## mxmlSetErrorCallback()

### Description

Set the error message callback.

### Syntax

```
void
mxmlSetErrorCallback(
    void (*cb)(const char *));
```

### Arguments

| Name | Description |
|---|---|
| (*cb)(const char *) | Error callback function |

### Returns

Nothing.

# mxmlSetInteger()

## Description

Set the value of an integer node. The node is not changed if it is not an integer node.

## Syntax

```
int
mxmlSetInteger(
    mxml_node_t * node,
    int integer);
```

## Arguments

| Name | Description |
|---|---|
| `node` | Node to set |
| `integer` | Integer value |

## Returns

0 on success, −1 on failure

# mxmlSetOpaque()

## Description

Set the value of an opaque node. The node is not changed if it is not an opaque node.

## Syntax

```
int
mxmlSetOpaque(
    mxml_node_t * node,
    const char * opaque);
```

## Arguments

| Name | Description |
| --- | --- |
| `node` | Node to set |
| `opaque` | Opaque string |

## Returns

0 on success, −1 on failure

# mxmlSetReal()

## Description

Set the value of a real number node. The node is not changed if it is not a real number node.

## Syntax

```
int
mxmlSetReal(
    mxml_node_t * node,
    double real);
```

## Arguments

| Name | Description |
|---|---|
| node | Node to set |
| real | Real number value |

## Returns

0 on success, −1 on failure

# mxmlSetText()

## Description

Set the value of a text node. The node is not changed if it is not a text node.

## Syntax

```
int
mxmlSetText(
    mxml_node_t * node,
    int whitespace,
    const char * string);
```

## Arguments

| Name | Description |
|---|---|
| `node` | Node to set |
| `whitespace` | 1 = leading whitespace, 0 = no whitespace |
| `string` | String |

## Returns

0 on success, −1 on failure

# mxmlSetTextf()

## Description

Set the value of a text node to a formatted string. The node is not changed if it is not a text node.

## Syntax

```
int
mxmlSetTextf(
    mxml_node_t * node,
    int whitespace,
    const char * format,
    ...);
```

## Arguments

| Name | Description |
|---|---|
| `node` | Node to set |
| `whitespace` | 1 = leading whitespace, 0 = no whitespace |
| `format` | Printf−style format string |
| `...` | Additional arguments as needed |

## Returns

0 on success, −1 on failure

## mxmlWalkNext()

### Description

Walk to the next logical node in the tree. The descend argument controls whether the first child is considered to be the next node. The top node argument constrains the walk to the node's children.

### Syntax

```
mxml_node_t *
mxmlWalkNext(
    mxml_node_t * node,
    mxml_node_t * top,
    int descend);
```

### Arguments

| Name | Description |
|---|---|
| node | Current node |
| top | Top node |
| descend | Descend into tree − MXML_DESCEND, MXML_NO_DESCEND, or MXML_DESCEND_FIRST |

### Returns

Next node or NULL

## mxmlWalkPrev()

### Description

Walk to the previous logical node in the tree. The descend argument controls whether the previous node's last child is considered to be the previous node. The top node argument constrains the walk to the node's children.

### Syntax

```
mxml_node_t *
mxmlWalkPrev(
    mxml_node_t * node,
    mxml_node_t * top,
    int descend);
```

### Arguments

| Name | Description |
|---|---|
| `node` | Current node |
| `top` | Top node |
| `descend` | Descend into tree − MXML_DESCEND, MXML_NO_DESCEND, or MXML_DESCEND_FIRST |

### Returns

Previous node or NULL

# Structures

- mxml_attr_s
- mxml_index_s
- mxml_node_s
- mxml_text_s
- mxml_value_s

# mxml_attr_s

## Description

An XML element attribute value.

## Definition

```
struct mxml_attr_s
{
  char * name;
  char * value;
};
```

## Members

| Name | Description |
|---|---|
| `name` | Attribute name |
| `value` | Attribute value |

# mxml_index_s

## Description

An XML node index.

## Definition

```
struct mxml_index_s
{
  int alloc_nodes;
  char * attr;
  int cur_node;
  mxml_node_t ** nodes;
  int num_nodes;
};
```

## Members

| Name | Description |
|---|---|
| alloc_nodes | Allocated nodes in index |
| attr | Attribute used for indexing or NULL |
| cur_node | Current node |
| nodes | Node array |
| num_nodes | Number of nodes in index |

## mxml_node_s

### Description

An XML node.

### Definition

```
struct mxml_node_s
{
  struct mxml_node_s * child;
  struct mxml_node_s * last_child;
  struct mxml_node_s * next;
  struct mxml_node_s * parent;
  struct mxml_node_s * prev;
  mxml_type_t type;
  mxml_value_t value;
};
```

### Members

| Name | Description |
|---|---|
| `child` | First child node |
| `last_child` | Last child node |
| `next` | Next node under same parent |
| `parent` | Parent node |
| `prev` | Previous node under same parent |
| `type` | Node type |
| `value` | Node value |

# mxml_text_s

## Description

An XML text value.

## Definition

```
struct mxml_text_s
{
  char * string;
  int whitespace;
};
```

## Members

| Name | Description |
| --- | --- |
| string | Fragment string |
| whitespace | Leading whitespace? |

## mxml_value_s

### Description

An XML element value.

### Definition

```
struct mxml_value_s
{
  mxml_attr_t * attrs;
  char * name;
  int num_attrs;
};
```

### Members

| Name | Description |
|---|---|
| attrs | Attributes |
| name | Name of element |
| num_attrs | Number of attributes |

# Types

- mxml_attr_t
- mxml_element_t
- mxml_index_t
- mxml_node_t
- mxml_text_t
- mxml_type_t
- mxml_value_t

## mxml_attr_t

### Description

An XML element attribute value.

### Definition

```
typedef struct mxml_attr_s mxml_attr_t;
```

## mxml_element_t

### Description

An XML element value.

### Definition

```
typedef struct mxml_value_s mxml_element_t;
```

## mxml_index_t

### Description

An XML node index.

### Definition

```
typedef struct mxml_index_s mxml_index_t;
```

## mxml_node_t

### Description

An XML node.

### Definition

```
typedef struct mxml_node_s mxml_node_t;
```

## mxml_text_t

### Description

An XML text value.

### Definition

```
typedef struct mxml_text_s mxml_text_t;
```

## mxml_type_t

### Description

The XML node type.

### Definition

```
typedef enum mxml_type_e mxml_type_t;
```

## mxml_value_t

### Description

An XML node value.

### Definition

```
typedef union mxml_value_u mxml_value_t;
```

# Unions

- `mxml_value_u`

# mxml_value_u

## Description

An XML node value.

## Definition

```
union mxml_value_u
{
  mxml_element_t element;
  int integer;
  char * opaque;
  double real;
  mxml_text_t text;
};
```

## Members

| Name | Description |
| --- | --- |
| `element` | Element |
| `integer` | Integer number |
| `opaque` | Opaque string |
| `real` | Real number |
| `text` | Text fragment |

# Variables

- num_callbacks

## num_callbacks

### Definition

```
static int num_callbacks = 1;
```