

# libcmml Reference Manual

## 0.8.1

Generated by Doxygen 1.3.4

Sun Jan 4 21:28:29 2004



# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>libcmml Main Page</b>                       | <b>1</b>  |
| 1.1      | Introduction . . . . .                         | 1         |
| 1.2      | Features of CMML . . . . .                     | 1         |
| 1.3      | Compiling and Installing libcmml . . . . .     | 2         |
| 1.4      | Programming with libcmml . . . . .             | 2         |
| 1.5      | The CMML DTD . . . . .                         | 2         |
| 1.6      | Licensing . . . . .                            | 2         |
| 1.7      | Acknowledgements . . . . .                     | 3         |
| <b>2</b> | <b>libcmml Module Index</b>                    | <b>5</b>  |
| 2.1      | libcmml Modules . . . . .                      | 5         |
| <b>3</b> | <b>libcmml Data Structure Index</b>            | <b>7</b>  |
| 3.1      | libcmml Data Structures . . . . .              | 7         |
| <b>4</b> | <b>libcmml File Index</b>                      | <b>9</b>  |
| 4.1      | libcmml File List . . . . .                    | 9         |
| <b>5</b> | <b>libcmml Module Documentation</b>            | <b>11</b> |
| 5.1      | Writing CMML files . . . . .                   | 11        |
| 5.2      | Concepts of time in CMML . . . . .             | 13        |
| 5.3      | Internationalisation support in CMML . . . . . | 15        |
| 5.4      | Multitrack input media in CMML . . . . .       | 17        |
| 5.5      | Multitrack annotations in CMML . . . . .       | 18        |
| <b>6</b> | <b>libcmml Data Structure Documentation</b>    | <b>19</b> |
| 6.1      | _CMML_List Struct Reference . . . . .          | 19        |
| 6.2      | CMML_Clip Struct Reference . . . . .           | 21        |
| 6.3      | CMML_Element Struct Reference . . . . .        | 25        |
| 6.4      | CMML_Error Struct Reference . . . . .          | 26        |

|          |   |           |
|----------|---|-----------|
| 6.5      | CMML_Head Struct Reference . . . . .          | 27        |
| 6.6      | CMML_ImportElement Struct Reference . . . . . | 29        |
| 6.7      | CMML_MetaElement Struct Reference . . . . .   | 31        |
| 6.8      | CMML_ParamElement Struct Reference . . . . .  | 32        |
| 6.9      | CMML_Preamble Struct Reference . . . . .      | 33        |
| 6.10     | CMML_Stream Struct Reference . . . . .        | 35        |
| 6.11     | CMML_Time Struct Reference . . . . .          | 36        |
| 6.12     | CMML_UTC Struct Reference . . . . .           | 37        |
| <b>7</b> | <b>libcmml File Documentation</b>             | <b>39</b> |
| 7.1      | cmml-fix.c File Reference . . . . .           | 39        |
| 7.2      | cmml-fortune.c File Reference . . . . .       | 42        |
| 7.3      | cmml-validate.c File Reference . . . . .      | 44        |
| 7.4      | cmml.h File Reference . . . . .               | 47        |
| <b>8</b> | <b>libcmml Example Documentation</b>          | <b>73</b> |
| 8.1      | cmml-parse.c . . . . .                        | 73        |
| 8.2      | cmml-peek-clip.c . . . . .                    | 75        |
| 8.3      | cmml-peek-offset.c . . . . .                  | 76        |
| 8.4      | cmml-peek-utc.c . . . . .                     | 77        |
| 8.5      | cmml-uri-file.c . . . . .                     | 78        |
| 8.6      | cmml-validate.c . . . . .                     | 80        |
| 8.7      | cmml-write.c . . . . .                        | 85        |
| 8.8      | cmml.dtd . . . . .                            | 87        |

# Chapter 1

## libcmml Main Page

### 1.1 Introduction

**libcmml** is a library that provides a complete programming interface including functions, data structures, and sloppy or strict error handling to parse a XML file in CMML. CMML is the Continuous Media Markup Language defined as part of the Continuous Media Web project (see <http://www.annodex.net/software/libcmml/>).

libcmml also includes the following command-line tools:

- `cmml-validate`, which takes as input a CMML file and tests it against the `cmml.dtd`
- `cmml-fix`, which fixes a sloppily written input CMML and creates a valid one if possible
- `cmml-fortune`, which creates a valid CMML file with random content

### 1.2 Features of CMML

The version of CMML that this version of libcmml supports is CMML 2.0 . It has the following features:

- html-like markup language for audio, video, and other time-continuous data files (call them "media files")
- provides markup to structure an input media file into clips by identification of time intervals
- URI hyperlinking to clips is possible
- provides structured annotations (meta tags) and unstructured annotations (free text) both for the complete media file and each clips
- URI hyperlinks from clips to other Web resources possible
- URI hyperlinks from clips to representative images (keyframes) possible
- internationalisation (i18n) support for markup
- multi-track composition directions for media files from several input media files possible
- several tracks of annotations (multi-track annotations) possible

- arbitrarily high temporal resolution for annotation and media tracks
- non-zero timbase association with media files possible
- wall-clock time association with media files possible

To learn more about CMML and how to write a CMML file, check out the `Modules` documentation.

## 1.3 Compiling and Installing libcmml

libcmml is developed in C under Linux and OS X but is easily portable. A Visual C++ MS Windows port is included in the tarball.

libcmml uses expat for parsing XML. Get the appropriate version for your OS from <http://expat.sourceforge.net/>.

libcmml uses doxygen to create documentation and docbook to create manpages.

To install on Linux, OS X, and other Unix-like OSs run the usual configure, make, make install sequence. Full details in the `INSTALL` file.

To install on MS Windows you have the choice between a MS Visual Studio version 6 workspace file, a MS Visual Studio version 7 solution, and using nmake with the `Win32/Makefile` file. Full details in the `README.win32` file.

## 1.4 Programming with libcmml

The libcmml **API documentation** can be found [here](#). The API of libcmml is based on a convenient callback based framework for parsing CMML files. This enables activation of actions for the stream, head and clip tags while reading a CMML file. After opening a CMML file for reading, you attach callbacks to each of these types of tags. Then, as bytes are read, libcmml will call your callbacks as appropriate. Check out the `code examples` to gain a better understanding.

## 1.5 The CMML DTD

The DTD for CMML can be found [here](#).

## 1.6 Licensing

libcmml is provided under the following BSD-style open source license:

```
Copyright (C) 2003 CSIRO Australia
```

```
Redistribution and use in source and binary forms, with or without  
modification, are permitted provided that the following conditions  
are met:
```

```
- Redistributions of source code must retain the above copyright  
notice, this list of conditions and the following disclaimer.
```

```
- Redistributions in binary form must reproduce the above copyright  
notice, this list of conditions and the following disclaimer in the
```

documentation and/or other materials provided with the distribution.

- Neither the name of the CSIRO nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE ORGANISATION OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## 1.7 Acknowledgements

Software was developed by CSIRO, Australia,  
Analytic Audio Systems research group in  
Division of Mathematical and Information Sciences,  
later Continuous Media Web research group in  
ICT Research Centre.  
(c) Copyright CSIRO 2002-

### Authors:

Silvia Pfeiffer <Silvia.Pfeiffer@csiro.au>  
Conrad Parker <Conrad.Parker@csiro.au>  
Andre Pang <Andre.Pang@csiro.au>

### Contributions:

Zen Kavanagh <Zen@illiminable.com>  
Andrew Nesbit <alnesbit@students.cs.mu.OZ.AU>  
Ben Leslie <benno@benno.id.au>  
Jamie Wilkinson <jaq@spacepants.org>





## Chapter 2

# libcmml Module Index

### 2.1 libcmml Modules

Here is a list of all modules:

|  |    |
|--|----|
| Writing CMML files . . . . .                   | 11 |
| Concepts of time in CMML . . . . .             | 13 |
| Internationalisation support in CMML . . . . . | 15 |
| Multitrack input media in CMML . . . . .       | 17 |
| Multitrack annotations in CMML . . . . .       | 18 |



## Chapter 3

# libcmml Data Structure Index

### 3.1 libcmml Data Structures

Here are the data structures with brief descriptions:

|                              |    |
|------------------------------|----|
| CMML_List . . . . .          | 19 |
| CMML_Clip . . . . .          | 21 |
| CMML_Element . . . . .       | 25 |
| CMML_Error . . . . .         | 26 |
| CMML_Head . . . . .          | 27 |
| CMML_ImportElement . . . . . | 29 |
| CMML_MetaElement . . . . .   | 31 |
| CMML_ParamElement . . . . .  | 32 |
| CMML_Preamble . . . . .      | 33 |
| CMML_Stream . . . . .        | 35 |
| CMML_Time . . . . .          | 36 |
| CMML_UTC . . . . .           | 37 |



## Chapter 4

# libcmml File Index

### 4.1 libcmml File List

Here is a list of all documented files with brief descriptions:

|                        |    |
|------------------------|----|
| <b>cmml-fix.c</b>      | 39 |
| <b>cmml-fortune.c</b>  | 42 |
| <b>cmml-validate.c</b> | 44 |
| <b>cmml.h</b>          | 47 |
| <b>cmml_dox.h</b>      | ?? |



## Chapter 5

# libcmml Module Documentation

### 5.1 Writing CMML files

The following simple example CMML file demonstrates the main features of CMML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE cmml SYSTEM "cmml.dtd">

<cmml lang="en">

<stream timebase="0">
  <import contenttype="video/mpeg" src="fish.mpg" start="0"/>
</stream>

<head>
  <title>Types of fish</title>
  <meta name="Producer" content="Joe Ordinary"/>
  <meta name="DC.Author" content="Joe's friend"/>
</head>

<clip id="intro" start="0">
  <a href="http://www.blah.au/fish.html">Read more about fish</a>
  <desc>This is the introduction to the film Joe made about fish.</desc>
</clip>

<clip id="dolphin" start="npt:3.5" end="npt:5:5.9">
  
  <desc>Here, Joe caught sight of a dolphin in the ocean.</desc>
  <meta name="Subject" content="dolphin"/>
</clip>

<clip id="goldfish" start="npt:5:5.9">
  <a href="http://www.blah.au/morefish.anx?id=goldfish">More video clips on goldfish.</a>
  
  <desc>Joe has a fishtank at home with many colourful fish. The common goldfish is one of them and Joe's favourite. He
  <meta name="Location" content="Joe's fishtank"/>
  <meta name="Subject" content="goldfish"/>
</clip>

</cmml>
```

### 5.1.1 The preamble

After the usual preamble including the xml processing instruction and the doctype declaration, the root **cmml** tag houses the markup.

### 5.1.2 The stream tag

The first markup will usually be a **stream** tag which names the source media file(s) that the markup of the CMML file relates to. The stream tag is optional as you may want to prepare a CMML file for a not yet existing media file, for a live media stream, or just as a template.

### 5.1.3 The head tag

The **head** tag is mandatory as with html. It will at least contain either a title or a base tag. The **title** tag contains a short description of the complete annotated work. More structured information for the work go into the **meta** tags. It is recommended to use existing meta schemes such as the Dublin Core for the structured annotations.

### 5.1.4 The clip tag

The following **clip** tags structure the work into temporal sections by specification of start and end times for each section. The end time is optional as a clip will be implicitly ended with the start of the next clip or the end of the file. You may attach an **a** tag to a clip to specify a related Web resource through its href attribute. You may attach an **img** tag to a clip to specify a URL to an image that represents the content of the clip visually. You may attach a **desc** tag to a clip to provide a textual description. More structured information for the clip go into the **meta** tags.

### 5.1.5 Finishing the CMML file

The closing of the cmml tag ends the CMML file.



## 5.2 Concepts of time in CMML

Using time specifications in CMML is powerful but somewhat confusing. So, we describe it here in more detail. CMML has specifications of time in the stream and the clip tag. There are two different concepts of time specifications possible in CMML: utc time and playback time.

### 5.2.1 The stream tag

The stream tag contains specifications of the INPUT media document(s) in the media tags. Conversely, the attributes of the stream tag refer to the OUTPUT annodex media document for which this CMML file provides the markup.

#### 5.2.1.1 Timebase attribute

The timebase attribute maps the first frame of the OUTPUT annodex document to a playback time to be displayed with the first frame. This playback time can be given as npt or some smpte specification given as a name-value pair, and it defaults to 0.

#### 5.2.1.2 Utc attribute

The optional utc attribute maps the timebase to a real-world clock time. It is given as a utc value.

#### 5.2.1.3 Start attribute of source tags

The start attribute specifies the insertion time of the specified INPUT document into the Annodex media file, i.e. the absolute time at which this media document (offset by seekoffset time) will start within the OUTPUT Annodexed file. It will usually be given as a npt or smpte time, which has to be greater than the timebase given in the stream tag (or just greater than 0 if there is not timebase given). The value is given as a name-value pair separated by a colon ":" though the name can be omitted if using npt. The start attribute can also be given as a plain utc time, which has to be some time after the utc time given in the stream tag. utc specification is not possible when the utc attribute in the stream tag is not given. The default value for the start attribute is "npt:0".

#### 5.2.1.4 Finishing the CMML file

The end attribute specifies when the inserted INPUT media document will end. The time specification is analogous to the start attribute. This attribute is optional as the default end of the INPUT media document is when it finishes.

#### 5.2.1.5 Location attribute of source tags

As the location is specified as a URI, it may contain temporal fragment offsets. These specify a temporal subpart of the INPUT media document which will be inserted into the Annodexed media stream.

### 5.2.2 The clip tag

The clip tag contains specifications of insertion time of the clip into the Annodexed media file.

#### 5.2.2.1 Start and end attributes

Insertion start and end times are given in the start and end attributes and they are specified in the same way as the start and end attributes of the source tag.

## 5.3 Internationalisation support in CMML

CMML is designed to provide full internationalisation (i18n) support, covering different character sets as well as languages and their differing directionality. Any tag or attribute that could end up containing text in a different language to the other tags may specify their own language.

As a CMML file is an XML file by definition, the xml processing instruction provides for a file-specific specification of the encoding format in its "encoding" tag. A potentially differing character set for an INPUT media file will be specified in the "contenttype" attribute of the **source** tag as a parameter to the MIME type.

Different languages and their directionality for display purposes are given for every tag that contains human-readable text as a value of either an attribute or a tag. They are specified in the **lang** and **dir** attributes.

The root **cmml** tag's "lang" and "dir" attributes provide the default language for the whole CMML file.

The **source** tag in the stream tag has i18n support as its "title" attribute may contain human readable text.

The **head** tag has i18n support to provide a default language to its contained tags such as the **title** and **meta** tags. Both of them may incidentally overrun the defaults by having a language specification of their own.

The **clip** tag again contains a default language for the clip itself. The **a**, **img**, **desc**, and **meta** tags may each override the default language.

Here is an example of a multi-lingual CMML file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE cmml SYSTEM "cmml.dtd">

<cmml lang="pl">

<stream timebase="npt:0">
  <import contenttype="video/mpeg" src="fish.mpg" start="npt:0"/>
</stream>

<head lang="en">
  <title>Types of fish</title>
  <meta name="Producer" content="Joe Ordinary"/>
  <meta name="DC.Author" content="Joe's friend"/>
</head>

<clip lang="de" id="intro" start="npt:0">
  <a href="http://www.blah.au/fish.html">Lesen Sie mehr &#252;ber Fische.</a>
  <desc lang="en">This is the introduction to the film Joe made about fish.</desc>
</clip>

<clip lang="en" id="goldfish" start="npt:5:5.9">
  <a href="http://www.blah.au/morefish.anx?id=goldfish">More video clips on goldfish.</a>
  
  <desc>Joe has a fishtank at home with many colourful fish. The common goldfish is one of them and Joe's favourite. Her
  <meta name="Location" content="Joe's fishtank"/>
  <meta name="Subject" content="goldfish"/>
  <meta lang="fr" name="sujet" content="poisson d'or"/>
</clip>

</cmml>
```

The document is generally in Polish (see lang attribute of cmml tag). The lang attribute of the head tag makes all head tags and attributes be English. The lang attribute of the first clip tag

makes all of them be German, though this is overrun by the lang attribute of the desc tag, making that one English. The last clip tag is in English, but it has a meta tag in French.

## 5.4 Multitrack input media in CMML

With CMML it is possible to create markup for several, potentially interleaved, media files. This is specified in the **stream** tag by giving several **source** tags which each point to a media file. The "start" tag specifies at what time the media file should be inserted. In case of temporally overlapping media files they are regarded as interleaved files.

This is an example of such a specification:

```
<stream timebase="npt:0">
  <source contenttype="video/x-theora" location="fish.ogg" start="npt:0"/>
  <source contenttype="audio/x-vorbis" location="fishsound.ogg" start="npt:0"/>
  <source contenttype="audio/x-speex" location="comments.spx" start="npt:7"/>
</stream>
```

## 5.5 Multitrack annotations in CMML

The default case for CMML is to have one annotation track with temporally non-overlapping clips of annotations. This is the case when the "track" attribute of the **clip** tag is not used.

It is however possible to specify several tracks of annotations within one CMML file. To that end, you need to give each track that you're using a name. This name needs to be written into the "track" attribute of the **clip** tags which belong onto that track. Not providing a track attribute attaches the clip to the default track.

Take care that the clips in one annotation track do not overlap in time.

This is an example of such a multi-track annotation specification:

```
<clip track="german" lang="de" id="intro_ge" start="npt:0">
  <a href="http://www.blah.au/fish.html">Lesen Sie mehr &#252;ber Fische.</a>
  <desc>Dies ist die Einleitung zum Film &#252;ber Fische von Joe.</desc>
</clip>

<clip track="default" lang="en" id="intro" start="npt:0">
  <a href="http://www.blah.au/fish.html">Read more about fish.</a>
  <desc>This is the introduction to the film Joe made about fish.</desc>
</clip>

<clip track="german" id="dolphin_ge" start="npt:3.5">
  <desc>Joe hat einen Delphin im Meer entdeckt.</desc>
  <meta name="Thema" content="Delphin"/>
</clip>

<clip id="dolphin" start="npt:3.5">
  <desc>Here, Joe caught sight of a dolphin in the ocean.</desc>
  <meta name="Subject" content="dolphin"/>
</clip>
```

## Chapter 6

# libcmml Data Structure Documentation

### 6.1 `_CMML_List` Struct Reference

```
#include <cmml.h>
```

#### Data Fields

- `CMML_List * prev`
- `CMML_List * next`
- `void * data`

#### 6.1.1 Detailed Description

USE THE DEFINED TYPE `CMML_List` FOR THIS!

`CMML_List`: A doubly linked list

#### 6.1.2 Field Documentation

##### 6.1.2.1 `CMML_List* _CMML_List::prev`

previous list member

##### 6.1.2.2 `CMML_List* _CMML_List::next`

next list member

##### 6.1.2.3 `void* _CMML_List::data`

actual member content

The documentation for this struct was generated from the following file:

- `cmml.h`



## 6.2 CMML\_Clip Struct Reference

```
#include <cmml.h>
```

### Data Fields

- char \* clip\_id
- char \* lang
- char \* dir
- char \* track
- CMML\_Time \* start\_time
- CMML\_Time \* end\_time
- CMML\_List \* meta
- char \* anchor\_id
- char \* anchor\_lang
- char \* anchor\_dir
- char \* anchor\_class
- char \* anchor\_href
- char \* anchor\_text
- char \* img\_id
- char \* img\_lang
- char \* img\_dir
- char \* img\_src
- char \* img\_alt
- char \* desc\_id
- char \* desc\_lang
- char \* desc\_dir
- char \* desc\_text

### 6.2.1 Detailed Description

CMML\_Clip: an clip element as presented in the CMML document.

#### Examples:

cmml-seek-clip.c, cmml-seek-offset.c, cmml-seek-utc.c, cmml-uri-file.c, cmml-validate.c, and cmml-write.c.

### 6.2.2 Field Documentation

#### 6.2.2.1 char\* CMML\_Clip::clip\_id

id attribute of clip

#### 6.2.2.2 char\* CMML\_Clip::lang

language attribute of desc element

**6.2.2.3 char\* CMML\_Clip::dir**

directionality of lang

**6.2.2.4 char\* CMML\_Clip::track**

track attribute of clip

**6.2.2.5 CMML\_Time\* CMML\_Clip::start\_time**

start time of this clip

**6.2.2.6 CMML\_Time\* CMML\_Clip::end\_time**

end time of this clip

**6.2.2.7 CMML\_List\* CMML\_Clip::meta**

list of meta elements

**6.2.2.8 char\* CMML\_Clip::anchor\_id**

id attribute of anchor

**6.2.2.9 char\* CMML\_Clip::anchor\_lang**

language of anchor

**6.2.2.10 char\* CMML\_Clip::anchor\_dir**

directionality of lang

**6.2.2.11 char\* CMML\_Clip::anchor\_class**

stylesheet of anchor

**6.2.2.12 char\* CMML\_Clip::anchor\_href**

href out of clip

**Examples:**

**cmml-write.c.**

**6.2.2.13 char\* CMML\_Clip::anchor\_text**

anchor text

**Examples:**

`cmml-write.c`.

**6.2.2.14 char\* CMML\_Clip::img\_id**

id attribute of image

**6.2.2.15 char\* CMML\_Clip::img\_lang**

language of img

**6.2.2.16 char\* CMML\_Clip::img\_dir**

directionality of lang

**6.2.2.17 char\* CMML\_Clip::img\_src**

keyframe image of clip

**6.2.2.18 char\* CMML\_Clip::img\_alt**

alternate text for image

**6.2.2.19 char\* CMML\_Clip::desc\_id**

id attribute of desc element

**6.2.2.20 char\* CMML\_Clip::desc\_lang**

language attribute of desc element

**6.2.2.21 char\* CMML\_Clip::desc\_dir**

directionality of lang

**6.2.2.22 char\* CMML\_Clip::desc\_text**

the description itself

**Examples:**

`cmml-seek-clip.c`, `cmml-seek-offset.c`, `cmml-seek-utc.c`, `cmml-uri-file.c`, and `cmml-write.c`.

The documentation for this struct was generated from the following file:

- `cmml.h`

## 6.3 CMML\_Element Struct Reference

```
#include <cmml.h>
```

### Data Fields

- **CMML\_Element\_Type** type
- union {
  - CMML\_Stream \* stream
  - CMML\_Head \* head
  - CMML\_Clip \* clip
- } e

#### 6.3.1 Detailed Description

CMML\_Element: wrapper type for any of the three elements which can occur in a CMML document (and are stored in CMML\_Status). Memory freed by `cmml_destroy_element()`

#### 6.3.2 Field Documentation

##### 6.3.2.1 CMML\_Element\_Type CMML\_Element::type

type of element to select in union

##### 6.3.2.2 CMML\_Stream\* CMML\_Element::stream

the stream tag

##### 6.3.2.3 CMML\_Head\* CMML\_Element::head

the head tag

##### 6.3.2.4 CMML\_Clip\* CMML\_Element::clip

the clip tag

##### 6.3.2.5 union { ... } CMML\_Element::e

holds either the stream, head or clip tag

The documentation for this struct was generated from the following file:

- `cmml.h`

## 6.4 CMML\_Error Struct Reference

```
#include <cmml.h>
```

### Data Fields

- **CMML\_Error\_Type type**
- **long line**
- **long col**

#### 6.4.1 Detailed Description

CMML\_Error: contains the error type plus the line and col numbers where they occurred.

##### Examples:

`cmml-validate.c.`

#### 6.4.2 Field Documentation

##### 6.4.2.1 CMML\_Error\_Type CMML\_Error::type

holds the error code

##### Examples:

`cmml-validate.c.`

##### 6.4.2.2 long CMML\_Error::line

line in which error occurred

##### 6.4.2.3 long CMML\_Error::col

column in which error occurred

The documentation for this struct was generated from the following file:

- `cmml.h`

## 6.5 CMML\_Head Struct Reference

```
#include <cmml.h>
```

### Data Fields

- char \* **head\_id**
- char \* **lang**
- char \* **dir**
- char \* **profile**
- char \* **title**
- char \* **title\_id**
- char \* **title\_lang**
- char \* **title\_dir**
- char \* **base\_id**
- char \* **base\_href**
- CMML\_List \* **meta**

### 6.5.1 Detailed Description

CMML\_Head: a head element as presented in the CMML document.

#### Examples:

`cmml-parse.c`, `cmml-validate.c`, and `cmml-write.c`.

### 6.5.2 Field Documentation

#### 6.5.2.1 char\* CMML\_Head::head\_id

id attribute of header

#### 6.5.2.2 char\* CMML\_Head::lang

language of header

#### 6.5.2.3 char\* CMML\_Head::dir

directionality of lang

#### 6.5.2.4 char\* CMML\_Head::profile

profile of header

#### 6.5.2.5 char\* CMML\_Head::title

title element of header

#### Examples:

`cmml-parse.c`, and `cmml-write.c`.

**6.5.2.6 char\* CMML\_Head::title\_id**

id attribute of title element

**6.5.2.7 char\* CMML\_Head::title\_lang**

language of title

**6.5.2.8 char\* CMML\_Head::title\_dir**

directionality of title\_lang

**6.5.2.9 char\* CMML\_Head::base\_id**

id attribute of base element

**6.5.2.10 char\* CMML\_Head::base\_href**

href attribute of base element

**6.5.2.11 CMML\_List\* CMML\_Head::meta**

list of meta elements

The documentation for this struct was generated from the following file:

- **cmml.h**



## 6.6 CMML\_ImportElement Struct Reference

```
#include <cmml.h>
```

### Data Fields

- char \* **id**
- char \* **lang**
- char \* **dir**
- char \* **granulerate**
- char \* **contenttype**
- char \* **src**
- CMML\_Time \* **start\_time**
- CMML\_Time \* **end\_time**
- char \* **title**
- CMML\_List \* **param**

#### 6.6.1 Detailed Description

CMML\_ImportElement: a struct to keep the details of the import tag

#### 6.6.2 Field Documentation

##### 6.6.2.1 char\* CMML\_ImportElement::id

id of import tag

##### 6.6.2.2 char\* CMML\_ImportElement::lang

language code of import tag

##### 6.6.2.3 char\* CMML\_ImportElement::dir

directionality of lang (ltr/rtl)

##### 6.6.2.4 char\* CMML\_ImportElement::granulerate

base temporal resolution in Hz

##### 6.6.2.5 char\* CMML\_ImportElement::contenttype

content type of the import bitstream

##### 6.6.2.6 char\* CMML\_ImportElement::src

URI to import document

**6.6.2.7 CMML\_Time\* CMML\_ImportElement::start\_time**

insertion time in annodex bitstream

**6.6.2.8 CMML\_Time\* CMML\_ImportElement::end\_time**

end time of this logical bitstream

**6.6.2.9 char\* CMML\_ImportElement::title**

comment on the import bitstream

**6.6.2.10 CMML\_List\* CMML\_ImportElement::param**

list of optional further nam-value metadata for the import bitstreams

The documentation for this struct was generated from the following file:

- **cmml.h**

## 6.7 CMML\_MetaElement Struct Reference

```
#include <cmml.h>
```

### Data Fields

- char \* **id**
- char \* **lang**
- char \* **dir**
- char \* **name**
- char \* **content**
- char \* **scheme**

### 6.7.1 Detailed Description

CMML\_MetaElement: a struct to keep the details of the meta tag in either a head or an clip

### 6.7.2 Field Documentation

#### 6.7.2.1 char\* CMML\_MetaElement::id

id attribute of meta element

#### 6.7.2.2 char\* CMML\_MetaElement::lang

language code of meta element

#### 6.7.2.3 char\* CMML\_MetaElement::dir

directionality of lang (ltr/rtl)

#### 6.7.2.4 char\* CMML\_MetaElement::name

property name of meta element

#### 6.7.2.5 char\* CMML\_MetaElement::content

property value of meta element

#### 6.7.2.6 char\* CMML\_MetaElement::scheme

scheme name of meta element

The documentation for this struct was generated from the following file:

- **cmml.h**

## 6.8 CMML\_ParamElement Struct Reference

```
#include <cmml.h>
```

### Data Fields

- char \* **id**
- char \* **name**
- char \* **value**

#### 6.8.1 Detailed Description

CMML\_ParamElement: a struct to keep the details of the param tag in a import element

#### 6.8.2 Field Documentation

##### 6.8.2.1 char\* CMML\_ParamElement::id

id of param element

##### 6.8.2.2 char\* CMML\_ParamElement::name

property name of param element

##### 6.8.2.3 char\* CMML\_ParamElement::value

property value of param element

The documentation for this struct was generated from the following file:

- **cmml.h**

## 6.9 CMML\_Preamble Struct Reference

```
#include <cmml.h>
```

### Data Fields

- char \* **xml\_version**
- char \* **xml\_encoding**
- int **xml\_standalone**
- int **doctype\_declared**
- char \* **cmml\_lang**
- char \* **cmml\_dir**
- char \* **cmml\_id**
- char \* **cmml\_xmlns**

### 6.9.1 Detailed Description

CMML\_Preamble: a struct to keep the general details of a CMML document available

#### Examples:

`cmml-validate.c`, and `cmml-write.c`.

### 6.9.2 Field Documentation

#### 6.9.2.1 char\* CMML\_Preamble::xml\_version

version attribute of xml proc instr

#### 6.9.2.2 char\* CMML\_Preamble::xml\_encoding

encoding attribute of xml proc instr

#### 6.9.2.3 int CMML\_Preamble::xml\_standalone

standalone attribute of xml proc instr

#### 6.9.2.4 int CMML\_Preamble::doctype\_declared

was doctype declared

#### 6.9.2.5 char\* CMML\_Preamble::cmml\_lang

lang attribute of cmml tag

#### 6.9.2.6 char\* CMML\_Preamble::cmml\_dir

dir attribute of cmml tag

**6.9.2.7 char\* CMML\_Preamble::cmml\_id**

id attribute of cmml tag

**6.9.2.8 char\* CMML\_Preamble::cmml\_xmlns**

xmlns attribute of cmml tag

The documentation for this struct was generated from the following file:

- **cmml.h**

## 6.10 CMML\_Stream Struct Reference

```
#include <cmml.h>
```

### Data Fields

- `char * id`
- `CMML_Time * timebase`
- `CMML_Time * utc`
- `CMML_List * import`

#### 6.10.1 Detailed Description

CMML\_Stream: an optional stream element containing information on the set of import bit-streams which are being annotated by the CMML data.

#### Examples:

`cmml-validate.c`.

#### 6.10.2 Field Documentation

##### 6.10.2.1 `char* CMML_Stream::id`

id attribute of stream element

##### 6.10.2.2 `CMML_Time* CMML_Stream::timebase`

timebase attribute of stream element

##### 6.10.2.3 `CMML_Time* CMML_Stream::utc`

utc attribute of stream element

##### 6.10.2.4 `CMML_List* CMML_Stream::import`

list of import elements

The documentation for this struct was generated from the following file:

- `cmml.h`

## 6.11 CMML\_Time Struct Reference

```
#include <cmml.h>
```

### Data Fields

- `char * tstr`
- `CMML_Time_Type type`
- union {
  - `CMML_UTC * utc`
  - `double sec`
- } `t`

#### 6.11.1 Detailed Description

`CMML_Time`: a time specification in CMML: either seconds or utc time; keeps the orininal time string for reuse, too

#### Examples:

`cmml-write.c`.

#### 6.11.2 Field Documentation

##### 6.11.2.1 `char* CMML_Time::tstr`

copy of original time string for reuse in printing

##### 6.11.2.2 `CMML_Time_Type CMML_Time::type`

the type of time for selecting the union content

##### 6.11.2.3 `CMML_UTC* CMML_Time::utc`

actual time in utc

##### 6.11.2.4 `double CMML_Time::sec`

actual time in seconds

##### 6.11.2.5 `union { ... } CMML_Time::t`

holds either a utc or a sec time

The documentation for this struct was generated from the following file:

- `cmml.h`



## 6.12 CMML\_\_UTC Struct Reference

```
#include <cmml.h>
```

### Data Fields

- short **tm\_hsec**
- short **tm\_sec**
- short **tm\_min**
- short **tm\_hour**
- short **tm\_mday**
- short **tm\_mon**
- short **tm\_year**

#### 6.12.1 Detailed Description

CMML\_\_UTC: utc time specification struct similar to the one defined in time.h

#### 6.12.2 Field Documentation

##### 6.12.2.1 short CMML\_\_UTC::tm\_hsec

hundreds of seconds [0-99]

##### 6.12.2.2 short CMML\_\_UTC::tm\_sec

seconds after the minute [0-59]

##### 6.12.2.3 short CMML\_\_UTC::tm\_min

minutes after the hour [0-59]

##### 6.12.2.4 short CMML\_\_UTC::tm\_hour

hours since midnight [0-23]

##### 6.12.2.5 short CMML\_\_UTC::tm\_mday

day of the month [1-31]

##### 6.12.2.6 short CMML\_\_UTC::tm\_mon

months since January [1-12]

#### 6.12.2.7 short CMML.UTC::tm\_year

years including century

The documentation for this struct was generated from the following file:

- `cmml.h`

# Chapter 7

## libcmml File Documentation

### 7.1 cmml-fix.c File Reference

```
#include <config.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <cmml.h>
```

#### Defines

- `#define BUFSIZE 100000`

#### Functions

- `void PrintUsage (char *prog)`
- `int read_stream (CMML *cmml, const CMML_Stream *stream, void *user_data)`
- `int read_head (CMML *cmml, const CMML_Head *head, void *user_data)`
- `int read_clip (CMML *cmml, const CMML_Clip *clip, void *user_data)`
- `int main (int argc, char *argv[])`

#### Variables

- `FILE * outfile`

### 7.1.1 Detailed Description

cmml-fix: parses a CMML instance document, validates it against the CMML.dtd ignoring sloppy errors and prints out a valid version of the file if possible.

Usage: cmml-fix [options] filename

Transform a sloppy CMML file to a valid one.

Possible options:

```
-i clip_id,  --id clip_id
               Start parsing from the named clip.
-s seconds,  --sec seconds
               Start parsing from the given seconds offset
-u utc,      --utc utc
               Start parsing from the given utc time
-o filename, --output filename
               Specify the output filename. The file is written
               to standard output by default.
-h, --help    Display this help information
-v, --version Display version information
```

### 7.1.2 Define Documentation

#### 7.1.2.1 #define BUFSIZE 100000

the size of the print buffer

### 7.1.3 Function Documentation

#### 7.1.3.1 void PrintUsage (char \* *prog*) [static]

PrintUsage: prints out help on how to use this program

Parameters:

*prog* the program's name

#### 7.1.3.2 int read\_stream (CMML \* *cmml*, const CMML\_Stream \* *stream*, void \* *user\_data*) [static]

read\_stream: the callback for a stream element

Parameters:

*cmml* the CMML\* handle in use

*stream* the stream element's content represented in a CMML\_Stream\*

*user\_data* user defined data

Returns:

0 on success, 1 on error

**7.1.3.3** `int read_head (CMML * cmml, const CMML_Head * head, void * user_data)` [static]

`read_head`: the callback for a head element

**Parameters:**

*cmml* the CMML\* handle in use

*head* the head element's content represented in a CMML\_Head\*

*user\_data* user defined data

**Returns:**

0 on success, 1 on error

**7.1.3.4** `int read_clip (CMML * cmml, const CMML_Clip * clip, void * user_data)` [static]

`read_clip`: the callback for a clip element

**Parameters:**

*cmml* the CMML\* handle in use

*clip* the clip element's content represented in a CMML\_Clip\*

*user\_data* user defined data

**Returns:**

0 on success, 1 on error

**7.1.3.5** `int main (int argc, char * argv[])`

main function of cmml-fix, which opens the CMML file, seeks to any given offsets, registers the callbacks, and then steps through the file in chunks of BUFSIZE size, during which the callbacks get activated as the relevant elements get parsed.

## 7.1.4 Variable Documentation

**7.1.4.1** `FILE* outfile` [static]

`outfile`: defines FILE pointer to print output to

## 7.2 cmml-fortune.c File Reference

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <ctype.h>
#include <string.h>
#include <unistd.h>
#include <cmml.h>
```

### Defines

- `#define DEFAULT_DURATION "00:30"`
- `#define DEFAULT_ENCODING "UTF-8"`
- `#define DEFAULT_SHORT_COMMAND "fortune -s -n 80"`
- `#define DEFAULT_LONG_COMMAND "fortune -l"`
- `#define BUFSIZE 10000`

### Functions

- `void PrintUsage (char *prog)`
- `char * get_fortune (char *command)`
- `int main (int argc, char *argv[])`

#### 7.2.1 Detailed Description

cmml-fortune: generates a valid CMML document with random content.

Usage: cmml-fortune [options]  
Generate a random CMML file using an external program  
to provide text (fortune cookies by default).

Possible options:

- d timespec, --duration timespec  
Specify the duration of the generated CMML file  
(‘00:30’ by default)
- e encoding, --encoding encoding  
Specify the encoding of the generated CMML file  
(‘UTF-8’ by default)
- o filename, --output filename  
Specify the output filename. The file is written  
to standard output by default.
- s command, --short-command command  
Specify the command to use to generate short text  
for the title and anchors (‘fortune -s -n 80’ by default)
- l command, --long-command command  
Specify the command to use to generate long text  
for descriptions (‘fortune -l’ by default)
- h, --help Display this help and exit
- v, --version Display version information and exit

## 7.2.2 Define Documentation

### 7.2.2.1 `#define DEFAULT_DURATION "00:30"`

DEFAULT\_DURATION: default duration of created CMML file

### 7.2.2.2 `#define DEFAULT_ENCODING "UTF-8"`

DEFAULT\_ENCODING: default encoding format of created CMML file

### 7.2.2.3 `#define DEFAULT_SHORT_COMMAND "fortune -s -n 80"`

DEFAULT\_SHORT\_COMMAND: default command to create short descriptions

### 7.2.2.4 `#define DEFAULT_LONG_COMMAND "fortune -l"`

DEFAULT\_LONG\_COMMAND: default command to create long descriptions

### 7.2.2.5 `#define BUFSIZE 10000`

BUFSIZE: default buffer size for printing

## 7.2.3 Function Documentation

### 7.2.3.1 `void PrintUsage (char * prog) [static]`

PrintUsage: prints out help on how to use this program

**Parameters:**

*prog* the program's name

### 7.2.3.2 `char* get_fortune (char * command) [static]`

get\_fortune: generates random text using the given command

**Parameters:**

*command* the command to use to generate text

**Returns:**

the generated text

### 7.2.3.3 `int main (int argc, char * argv[])`

main function of cmml-fortune, which opens the CMML file, seeks to any given offsets, registers the callbacks, and then steps through the file in chunks of BUFSIZE size, during which the callbacks get activated as the relevant elements get parsed.

## 7.3 cmml-validate.c File Reference

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <ctype.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <cmml.h>
```

### Defines

- `#define BUFSIZE 100000`

### Functions

- `void PrintUsage (char *prog)`
- `int read_stream (CMML *cmml, const CMML_Stream *stream, void *user_data)`
- `int read_head (CMML *cmml, const CMML_Head *head, void *user_data)`
- `int read_clip (CMML *cmml, const CMML_Clip *clip, void *user_data)`
- `int main (int argc, char *argv[])`

### Variables

- `int verbose`

#### 7.3.1 Detailed Description

cmml-validate: parses a CMML instance document and validates it against the CMML.dtd returning true/false and in case of fault the faulty tag including line and col number. Also spits out warnings for strange stuff.

Usage: cmml-validate [Options] filename  
Validate a CMML file.

Possible options:

```
-i clip_id, --id clip_id
    Start parsing from the named clip.
-s seconds, --sec seconds
    Start parsing from the given seconds offset
-u utc,      --utc utc
    Start parsing from the given utc time
-b, --verbose Output parsed file to stdout
-h, --help   Display this help information
-v, --version Display version information
```

View source code at: `cmml-validate.c`



## 7.3.2 Define Documentation

### 7.3.2.1 `#define BUFSIZE 100000`

the size of the print buffer

## 7.3.3 Function Documentation

### 7.3.3.1 `void PrintUsage (char * prog) [static]`

PrintUsage: prints out help on how to use this program

**Parameters:**

*prog* the program's name

**Examples:**

`cmml-validate.c`.

### 7.3.3.2 `int read_stream (CMML * cmml, const CMML_Stream * stream, void * user_data) [static]`

read\_stream: the callback for a stream element

**Parameters:**

*cmml* the CMML\* handle in use

*stream* the stream element's content represented in a CMML\_Stream\*

*user\_data* user defined data

**Returns:**

0 on success, 1 on error

**Examples:**

`cmml-validate.c`.

### 7.3.3.3 `int read_head (CMML * cmml, const CMML_Head * head, void * user_data) [static]`

read\_head: the callback for a head element

**Parameters:**

*cmml* the CMML\* handle in use

*head* the head element's content represented in a CMML\_Head\*

*user\_data* user defined data

**Returns:**

0 on success, 1 on error

**Examples:**

`cmml-parse.c`, and `cmml-validate.c`.

**7.3.3.4** `int read_clip (CMML * cmml, const CMML_Clip * clip, void * user_data)`  
[static]

`read_clip`: the callback for a clip element

**Parameters:**

*cmml* the CMML\* handle in use

*clip* the clip element's content represented in a CMML\_Clip\*

*user\_data* user defined data

**Returns:**

0 on success, 1 on error

**Examples:**

`cmml-seek-clip.c`, `cmml-seek-offset.c`, `cmml-seek-utc.c`, `cmml-uri-file.c`, and `cmml-validate.c`.

**7.3.3.5** `int main (int argc, char * argv[])`

main function of `cmml-validate`, which opens the CMML file, seeks to any given offsets, registers the callbacks, and then steps through the file in chunks of BUFSIZE size, during which the callbacks get activated as the relevant elements get parsed.

**Examples:**

`cmml-parse.c`, `cmml-seek-clip.c`, `cmml-seek-offset.c`, `cmml-seek-utc.c`, `cmml-uri-file.c`, `cmml-validate.c`, and `cmml-write.c`.

## 7.3.4 Variable Documentation

**7.3.4.1** `int verbose` [static]

`verbose`: turns on printing of parsed CMML file

## 7.4 cmml.h File Reference

```
#include <stdio.h>
```

```
#include <expat.h>
```

### Data Structures

- struct **CMML\_UTC**
- struct **CMML\_Time**
- struct **\_CMML\_List**
- struct **CMML\_Preamble**
- struct **CMML\_ParamElement**
- struct **CMML\_ImportElement**
- struct **CMML\_Stream**
- struct **CMML\_MetaElement**
- struct **CMML\_Head**
- struct **CMML\_Clip**
- struct **CMML\_Element**
- struct **CMML\_Error**

### Typedefs

- typedef void **CMML**
- typedef enum **\_CMML\_Time\_Type** **CMML\_Time\_Type**
- typedef **\_CMML\_List** **CMML\_List**
- typedef void **(\* CMML\_CloneFunc)**(void \*data)
- typedef void **(\* CMML\_FreeFunc)**(void \*data)
- typedef int **(\* CMML\_CmpFunc)**(void \*cmp\_ctx, void \*s1, void \*s2)
- typedef enum **\_CMML\_Element\_Type** **CMML\_Element\_Type**
- typedef enum **\_CMML\_Error\_Type** **CMML\_Error\_Type**
- typedef int **(\* CMMLReadStream)**(CMML \*cmml, const **CMML\_Stream** \*stream, void \*user\_data)
- typedef int **(\* CMMLReadHead)**(CMML \*cmml, const **CMML\_Head** \*head, void \*user\_data)
- typedef int **(\* CMMLReadClip)**(CMML \*cmml, const **CMML\_Clip** \*clip, void \*user\_data)

### Enumerations

- enum **\_CMML\_Time\_Type** { **CMML\_SEC\_TIME**, **CMML\_UTC\_TIME** }
- enum **\_CMML\_Element\_Type** {  
**CMML\_NONE**, **CMML\_CMML**, **CMML\_STREAM**, **CMML\_IMPORT**,  
**CMML\_HEAD**, **CMML\_CLIP** }
- enum **\_CMML\_Error\_Type** {  
**CMML\_OK**, **CMML\_EOF**, **CMML\_READ\_ERROR**, **CMML\_TIME\_ERROR**,  
**CMML\_MALLOC\_ERROR**, **CMML\_EXPAT\_ERROR**, **CMML\_PARSE\_ERROR**, **CMML\_NO\_CMML\_TAG**,

```

CMML_NO_HEAD_TAG, CMML_STREAM_NOT_FIRST, CMML_
HEAD_AFTER_CLIP, CMML_DUPLICATE_STREAM,
CMML_DUPLICATE_HEAD, CMML_FORMAT_ERROR, CMML_
UNKNOWN_TAG, CMML_TAG_IGNORED,
CMML_XMLNS_REDEFINED, CMML_NONSEQUENTIAL_CLIP }

```

## Functions

- **CMML \*** `cmml_open` (`char *XMLfilename`)
- **CMML \*** `cmml_new` (`FILE *file`)
- **FILE \*** `cmml_destroy` (**CMML \***`cmml`)
- **CMML \*** `cmml_close` (**CMML \***`cmml`)
- **int** `cmml_set_read_callbacks` (**CMML \***`cmml`, **CMMLReadStream** `read_stream`, **CMMLReadHead** `read_head`, **CMMLReadClip** `read_clip`, `void *user_data`)
- **long** `cmml_read` (**CMML \***`cmml`, **long** `n`)
- **void** `cmml_set_sloppy` (**CMML \***`cmml`, **int** `value`)
- **CMML\_Preamble \*** `cmml_get_preamble` (**CMML \***`cmml`)
- **CMML\_Stream \*** `cmml_get_last_stream` (**CMML \***`cmml`)
- **CMML\_Head \*** `cmml_get_last_head` (**CMML \***`cmml`)
- **CMML\_Clip \*** `cmml_get_last_clip` (**CMML \***`cmml`)
- **CMML\_Clip \*** `cmml_get_previous_clip` (**CMML \***`cmml`)
- **CMML\_Error \*** `cmml_get_last_error` (**CMML \***`cmml`)
- **void** `cmml_error_clear` (**CMML \***`cmml`)
- **int** `cmml_set_window` (**CMML \***`cmml`, **CMML\_Time** `*start`, **CMML\_Time** `*end`)
- **double** `cmml_skip_to_secs` (**CMML \***`cmml`, **double** `seconds`)
- **double** `cmml_skip_to_utc` (**CMML \***`cmml`, **const char \***`utc`)
- **double** `cmml_skip_to_id` (**CMML \***`cmml`, **const char \***`id`)
- **double** `cmml_skip_to_offset` (**CMML \***`cmml`, **const char \***`offset`)
- **CMML\_Preamble \*** `cmml_preamble_new` (**char \***`encoding`, **char \***`id`, **char \***`lang`, **char \***`dir`)
- **CMML\_Element \*** `cmml_element_new` (**CMML\_Element\_Type** `type`)
- **CMML\_Stream \*** `cmml_stream_new` (**void**)
- **CMML\_Head \*** `cmml_head_new` (**void**)
- **CMML\_Clip \*** `cmml_clip_new` (**CMML\_Time** `*start_time`, **CMML\_Time** `*end_time`)
- **CMML\_Error \*** `cmml_error_new` (**CMML\_Error\_Type** `type`)
- **CMML\_Preamble \*** `cmml_preamble_clone` (**CMML\_Preamble \***`src`)
- **CMML\_Element \*** `cmml_element_clone` (**CMML\_Element \***`src`)
- **CMML\_Stream \*** `cmml_stream_clone` (**CMML\_Stream \***`src`)
- **CMML\_Head \*** `cmml_head_clone` (**CMML\_Head \***`src`)
- **CMML\_Clip \*** `cmml_clip_clone` (**CMML\_Clip \***`src`)
- **void** `cmml_preamble_destroy` (**CMML\_Preamble \***`preamble`)
- **void** `cmml_element_destroy` (**CMML\_Element \***`element`)
- **void** `cmml_stream_destroy` (**CMML\_Stream \***`stream`)
- **void** `cmml_head_destroy` (**CMML\_Head \***`head`)
- **void** `cmml_clip_destroy` (**CMML\_Clip \***`clip`)
- **void** `cmml_error_destroy` (**CMML\_Error \***`error`)
- **int** `cmml_preamble_snprint` (**char \***`buf`, **int** `n`, **CMML\_Preamble \***`pre`)
- **int** `cmml_element_snprint` (**char \***`buf`, **int** `n`, **CMML\_Element \***`elem`)
- **int** `cmml_stream_snprint` (**char \***`buf`, **int** `n`, **CMML\_Stream \***`stream`)

- `int cmml_stream_pretty_snprint (char *buf, int n, CMML_Stream *stream)`
- `int cmml_head_snprint (char *buf, int n, CMML_Head *head)`
- `int cmml_head_pretty_snprint (char *buf, int n, CMML_Head *head)`
- `int cmml_clip_snprint (char *buf, int n, CMML_Clip *clip)`
- `int cmml_clip_pretty_snprint (char *buf, int n, CMML_Clip *clip)`
- `int cmml_error_snprint (char *buf, int n, CMML_Error *error, CMML *cmml)`
- `CMML_List * cmml_list_new (void)`
- `CMML_List * cmml_list_clone (CMML_List *list)`
- `CMML_List * cmml_list_clone_with (CMML_List *list, CMML_CloneFunc clone)`
- `CMML_List * cmml_list_tail (CMML_List *list)`
- `CMML_List * cmml_list_prepend (CMML_List *list, void *data)`
- `CMML_List * cmml_list_append (CMML_List *list, void *data)`
- `CMML_List * cmml_list_add_before (CMML_List *list, void *data, CMML_List *node)`
- `CMML_List * cmml_list_add_after (CMML_List *list, void *data, CMML_List *node)`
- `CMML_List * cmml_list_find (CMML_List *list, void *data)`
- `CMML_List * cmml_list_remove (CMML_List *list, CMML_List *node)`
- `int cmml_list_length (CMML_List *list)`
- `int cmml_list_is_empty (CMML_List *list)`
- `int cmml_list_is_singleton (CMML_List *list)`
- `CMML_List * cmml_list_free_with (CMML_List *list, CMML_FreeFunc free_func)`
- `CMML_List * cmml_list_free (CMML_List *list)`
- `CMML_Time * cmml_time_new (const char *s)`
- `CMML_Time * cmml_sec_new (const char *s)`
- `CMML_Time * cmml_utc_new (const char *s)`
- `int cmml_time_interval_new (const char *s, CMML_Time **t_start, CMML_Time **t_end)`
- `CMML_Time * cmml_time_new_in_sec (const char *s, CMML_Time *ref, double base)`
- `CMML_Time * cmml_time_utc_to_sec (CMML_Time *t, CMML_Time *ref, double base)`
- `void cmml_time_free (CMML_Time *t)`
- `CMML_Time * cmml_time_clone (CMML_Time *t)`
- `double cmml_sec_parse (const char *s)`
- `CMML_UTC * cmml_utc_parse (const char *s)`
- `CMML_UTC * cmml_utc_clone (CMML_UTC *t)`
- `double cmml_utc_diff (CMML_UTC *t2, CMML_UTC *t1)`
- `int cmml_npt_snprint (char *buf, int n, double seconds)`
- `int cmml_utc_snprint (char *buf, int n, CMML_UTC *t)`
- `int cmml_utc_pretty_snprint (char *buf, int n, CMML_UTC *t)`

### 7.4.1 Detailed Description

`cmml.h`(p. 47) includes definitions of the public API and data types required to parse CMML 2.0 files.

## 7.4.2 Typedef Documentation

### 7.4.2.1 typedef void CMML

A CMML handle.

### 7.4.2.2 typedef enum \_CMML\_Time\_Type CMML\_Time\_Type

CMML\_Time\_Type: enumerates the different time types

### 7.4.2.3 typedef struct \_CMML\_List CMML\_List

CMML\_List: A doubly linked list

### 7.4.2.4 typedef void\*(\* CMML\_CloneFunc)(void \* data)

CMML\_CloneFunc: Signature of a cloning function for CMML\_List.

### 7.4.2.5 typedef void\*(\* CMML\_FreeFunc)(void \* data)

CMML\_FreeFunc: Signature of a freeing function.

### 7.4.2.6 typedef int(\* CMML\_CmpFunc)(void \* cmp\_ctx, void \* s1, void \* s2)

CMML\_CmpFunc: Signature of a comparison function for CMML\_List to compares two scalars, returning: +ve,  $s1 > s2$  0,  $s1 == s2$  -ve,  $s1 < s2$

### 7.4.2.7 typedef enum \_CMML\_Element\_Type CMML\_Element\_Type

CMML\_Element\_Type: indication of what the CMML\_Element(p. 25) actually represents.

### 7.4.2.8 typedef enum \_CMML\_Error\_Type CMML\_Error\_Type

CMML\_Error\_Type: indication of the type of error that occurred. There are three classes of errors:

- general processing feedback (CMML\_OK, CMML\_EOF)
- fatal processing feedback (these errors render the CMML file useless)
- tolerable processing errors (these lead to ignoring certain tags) The user may select to

### 7.4.2.9 typedef int(\* CMMLReadStream)(CMML \*cmml, const CMML\_Stream \*stream, void \* user\_data)

CMMLReadStream: Signature for a callback called when the CMML stream element is parsed

#### Parameters:

*cmml* the CMML\* handle in use

*stream* the stream element's content represented in a CMML\_Stream\*

*user\_data* user defined data

**Returns:**

0 on success, 1 on error

**7.4.2.10** `typedef int(* CMMLReadHead)(CMML *cmml, const CMML_Head *head, void * user_data)`

CMMLReadHead: Signature for a callback called when the CMML head element is parsed

**Parameters:**

*cmml* the CMML\* handle in use

*head* the head element's content represented in a CMML\_Head\*

*user\_data* user defined data

**Returns:**

0 on success, 1 on error

**7.4.2.11** `typedef int(* CMMLReadClip)(CMML *cmml, const CMML_Clip *clip, void * user_data)`

CMMLReadClip: Signature for a callback called when a CMML clip element is parsed

**Parameters:**

*cmml* the CMML\* handle in use

*clip* the clip element's content represented in a CMML\_Clip\*

*user\_data* user defined data

**Returns:**

0 on success, 1 on error

## 7.4.3 Enumeration Type Documentation

**7.4.3.1** `enum _CMML_Time_Type`

CMML\_Time\_Type: enumerates the different time types

**Enumeration values:**

*CMML\_SEC\_TIME* time is seconds

*CMML\_UTC\_TIME* time is utc

### 7.4.3.2 enum \_CMML\_Element\_Type

CMML\_Element\_Type: indication of what the **CMML\_Element**(p.25) actually represents.

Enumeration values:

**CMML\_NONE** no element  
**CMML\_CMML** cmml element  
**CMML\_STREAM** stream element  
**CMML\_IMPORT** import element  
**CMML\_HEAD** head element  
**CMML\_CLIP** clip element

### 7.4.3.3 enum \_CMML\_Error\_Type

CMML\_Error\_Type: indication of the type of error that occurred. There are three classes of errors:

- general processing feedback (CMML\_OK, CMML\_EOF)
- fatal processing feedback (these errors render the CMML file useless)
- tolerable processing errors (these lead to ignoring certain tags) The user may select to

Enumeration values:

**CMML\_OK** everything went fine and EOF was not encountered  
**CMML\_EOF** EOF, but everything went fine  
**CMML\_READ\_ERROR** Reading the raw CMML doc into a buffer  
**CMML\_TIME\_ERROR** Error in formatting of time in a required tag  
**CMML\_MALLOC\_ERROR** Memory allocation error  
**CMML\_EXPAT\_ERROR** Expat reported an error  
**CMML\_PARSE\_ERROR** General parsing error  
**CMML\_NO\_CMML\_TAG** No cmml tag was found  
**CMML\_NO\_HEAD\_TAG** No head tag was found  
**CMML\_STREAM\_NOT\_FIRST** there is a head or a tag before the stream  
**CMML\_HEAD\_AFTER\_CLIP** there is a clip tag before the head tag  
**CMML\_DUPLICATE\_STREAM** there are two stream tags  
**CMML\_DUPLICATE\_HEAD** there are two head tags  
**CMML\_FORMAT\_ERROR** error in the formatting of an implied attribute  
**CMML\_UNKNOWN\_TAG** skipping an unknown tag  
**CMML\_TAG\_IGNORED** skipping a faulty tag  
**CMML\_XMLNS\_REDEFINED** xmlns in cmml tag is wrongly redefined  
**CMML\_NONSEQUENTIAL\_CLIP** clip with start time in the "past" skipped



## 7.4.4 Function Documentation

### 7.4.4.1 CMML\* cmml\_open (char \* *XMLfilename*)

`cmml_open`: Takes a filename and opens it as read-only. Then checks whether it is an XML file satisfying the CMML specification with the `xml` directive and the `cmml` tag. It returns a CMML object. If you're working on an already opened `FILE*`, use `cmml_open_stdio` instead.

**Parameters:**

*XMLfilename* a CMML file to open

**Returns:**

a CMML\* handle

**Examples:**

`cmml-seek-clip.c`, `cmml-seek-offset.c`, `cmml-seek-utc.c`, `cmml-uri-file.c`, and `cmml-validate.c`.

### 7.4.4.2 CMML\* cmml\_new (FILE \* *file*)

`cmml_attach_parser`: Takes a `FILE` pointer to an already-opened CMML document. Returns a CMML object containing internal initialised data structures, etc, which is used in subsequent operations on the CMML file. This function must be called exactly once, after the file descriptor has been opened.

**Parameters:**

*file* an open `FILE` pointer

**Returns:**

a CMML\* handle

**Examples:**

`cmml-parse.c`, and `cmml-validate.c`.

### 7.4.4.3 FILE\* cmml\_destroy (CMML \* *cmml*)

`cmml_detach_parser`: Close a CMML object. Return the `FILE` pointer from which the CMML object was derived.

**Parameters:**

*cmml* a CMML\* handle

**Returns:**

the open `FILE` pointer

**Examples:**

`cmml-parse.c`.

#### 7.4.4.4 CMML\* `cmml_close` (CMML \* *cmml*)

`cmml_close`: Close a CMML object.

**Parameters:**

*cmml* a CMML\* handle

**Returns:**

NULL on success or the unchanged CMML object (e.g. in case of a system error on `close()`)

**Examples:**

`cmml-seek-clip.c`, `cmml-seek-offset.c`, `cmml-seek-utc.c`, `cmml-uri-file.c`, and `cmml-validate.c`.

#### 7.4.4.5 `int cmml_set_read_callbacks` (CMML \* *cmml*, CMMLReadStream *read\_stream*, CMMLReadHead *read\_head*, CMMLReadClip *read\_clip*, void \* *user\_data*)

`cmml_set_read_callbacks`: registration function for callbacks.

**Parameters:**

*cmml* a CMML\* handle

*read\_stream* a CMMLReadStream callback function for the stream tag

*read\_head* a CMMLReadHead callback function for the head tag

*read\_clip* a CMMLReadClip callback function for the a tag

*user\_data* a pointer to a user defined object being passed to the callbacks

**Returns:**

0 on success, -1 on failure

**Examples:**

`cmml-parse.c`, `cmml-seek-clip.c`, `cmml-seek-offset.c`, `cmml-seek-utc.c`, `cmml-uri-file.c`, and `cmml-validate.c`.

#### 7.4.4.6 `long cmml_read` (CMML \* *cmml*, long *n*)

`cmml_read`: read *n* bytes from a file descriptor and parse them. Activates registered callbacks when reaching stream, head or clip end tags and blocks until they have completed. Stores parsed info in status. If an error or EOF occurs, returns -1 in which case you can check the error code with `cmml_last_error()`.

**Parameters:**

*cmml* a CMML\* handle

*n* number of bytes to read

**Returns:**

the actual number of bytes read

**Examples:**

`cmml-parse.c`, `cmml-seek-clip.c`, `cmml-seek-offset.c`, `cmml-seek-utc.c`, `cmml-uri-file.c`, and `cmml-validate.c`.

#### 7.4.4.7 void cmml\_set\_sloppy (CMML \* *cmml*, int *value*)

cmml\_set\_sloppy: Sets the parsing and error indication to ignore sloppy errors, which are defined below the CMML\_FORMAT\_ERROR in the CMML\_ErrorType struct

**Parameters:**

*cmml* a CMML\* handle

*value* a bool value to set the sloppy handling

**Examples:**

cmml-validate.c.

#### 7.4.4.8 CMML\_Preamble\* cmml\_get\_preamble (CMML \* *cmml*)

cmml\_get\_preamble: returns a structure containing some of the parsing information received out of a stream.

**Parameters:**

*cmml* a CMML\* handle

**Returns:**

1 if CMML tag specified, 0 otherwise

**Examples:**

cmml-validate.c.

#### 7.4.4.9 CMML\_Stream\* cmml\_get\_last\_stream (CMML \* *cmml*)

cmml\_get\_last\_stream: Return a copy of the current stream element.

**Parameters:**

*cmml* a CMML\* handle

**Returns:**

a CMML\_Stream\* structure with the stream element details; NULL if none existed

#### 7.4.4.10 CMML\_Head\* cmml\_get\_last\_head (CMML \* *cmml*)

cmml\_get\_last\_head: Return a copy of the current head element.

**Parameters:**

*cmml* a CMML\* handle

**Returns:**

a CMML\_Head\* structure with the head element details; NULL if none existed

**7.4.4.11 CMML\_Clip\* cmml\_get\_last\_clip (CMML \* *cmml*)**

cmml\_get\_last\_clip: Return a copy of the current clip element.

**Parameters:**

*cmml* a CMML\* handle

**Returns:**

a CMML\_Clip\* structure with the clip element details; NULL if none existed

**7.4.4.12 CMML\_Clip\* cmml\_get\_previous\_clip (CMML \* *cmml*)**

cmml\_get\_previous\_clip: Return a copy of the previous clip element.

**Parameters:**

*cmml* a CMML\* handle

**Returns:**

a CMML\_Clip\* structure with the clip element details; NULL if none existed

**7.4.4.13 CMML\_Error\* cmml\_get\_last\_error (CMML \* *cmml*)**

cmml\_get\_last\_error: Return the error status of the most recently executed cmml command.

**Parameters:**

*cmml* a CMML\* handle

**Returns:**

a CMML\_Error\* structure with the error details; NULL if none happened

**Examples:**

cmml-validate.c.

**7.4.4.14 void cmml\_error\_clear (CMML \* *cmml*)**

cmml\_error\_clear: Clear the last error.

**Parameters:**

*cmml* a CMML\* handle

**7.4.4.15 int cmml\_set\_window (CMML \* *cmml*, CMML\_Time \* *start*, CMML\_Time \* *end*)**

cmml\_set\_window: sets the active temporal interval of a cmml stream; all clips outside that interval get ignored

**Parameters:**

*cmml* a CMML\* handle

*start* the start time of the interval given as CMML\_Time\* structure

*end* the end time of the interval given as CMML\_Time\* structure

**Returns:**

2 if both times could be set, 1 for one only, and 0 if none

#### 7.4.4.16 double cmml\_skip\_to\_secs (CMML \* *cmml*, double *seconds*)

cmml\_skip\_to\_secs: seeks forward thru clip tags to required time offset (no backwards seeking: streaming!).

**Parameters:**

*cmml* a CMML\* handle

*seconds* the time offset to seek to

**Returns:**

the actual time offset seeked to or -1 on error

**Examples:**

cmml-seek-offset.c, and cmml-validate.c.

#### 7.4.4.17 double cmml\_skip\_to\_utc (CMML \* *cmml*, const char \* *utc*)

cmml\_skip\_to\_utc: seeks forward thru clip tags to required time offset (no backwards seeking: streaming!).

**Parameters:**

*cmml* a CMML\* handle

*utc* the time offset to seek to

**Returns:**

the actual time offset seeked to in seconds or -1 on error

**Examples:**

cmml-seek-utc.c, and cmml-validate.c.

#### 7.4.4.18 double cmml\_skip\_to\_id (CMML \* *cmml*, const char \* *id*)

cmml\_skip\_to\_id: seeks forward thru clip tags to required id (no backwards seeking: streaming!).

**Parameters:**

*cmml* a CMML\* handle

*id* the name of the clip tag to seek to

**Returns:**

the actual time offset seeked to in seconds or -1 on error

**Examples:**

cmml-seek-clip.c, and cmml-validate.c.

**7.4.4.19** `double cmml_skip_to_offset (CMML * cmml, const char * offset)`

`cmml_skip_to_offset`: seek forward thru clip tags and time to required fragment offset (no backwards seeking: streaming!).

**Parameters:**

*cmml* a CMML\* handl

*offset* the offset given as id or time spec

**Returns:**

the actual time offset seeked to in seconds or -1 on error

**Examples:**

`cmml-uri-file.c`.

**7.4.4.20** `CMML_Preamble* cmml_preamble_new (char * encoding, char * id, char * lang, char * dir)`

`cmml_preamble_new`: Create a new **CMML\_Preamble**(p.33).

**Parameters:**

*encoding* the character encoding to be used; UTF-8 if NULL

*id* the id attribute of the cmml tag; not used if NULL

*lang* the lang attribute of the cmml tag; not used if NULL

*dir* the dir attribute of the cmml tag; not used if NULL

**Returns:**

a CMML preamble

**Examples:**

`cmml-write.c`.

**7.4.4.21** `CMML_Element* cmml_element_new (CMML_Element_Type type)`

`cmml_element_new`: Create a new **CMML\_Element**(p.25).

**Parameters:**

*type* the element type to create (head, stream, or clip)

**Returns:**

a element of the requested type

**7.4.4.22** `CMML_Stream* cmml_stream_new (void)`

`cmml_stream_new`: Create a new **CMML\_Stream**(p.35).

**Returns:**

a stream element

**7.4.4.23 CMML\_Head\* cmml\_head\_new (void)**

cmml\_head\_new: Create a new **CMML\_Head**(p. 27).

**Returns:**

a head element

**Examples:**

cmml-write.c.

**7.4.4.24 CMML\_Clip\* cmml\_clip\_new (CMML\_Time \* *start\_time*,  
CMML\_Time \* *end\_time*)**

cmml\_clip\_new: Create a new **CMML\_Anchr**.

**Parameters:**

*start\_time* the start time at which to create an clip

*end\_time* the end time at which to end the clip

**Returns:**

a clip element

**Examples:**

cmml-write.c.

**7.4.4.25 CMML\_Error\* cmml\_error\_new (CMML\_Error\_Type *type*)**

cmml\_error\_new: Create a new **CMML\_Error**(p. 26).

**Parameters:**

*type* the type of error to create as from the given list

**Returns:**

an error type

**7.4.4.26 CMML\_Preamble\* cmml\_preamble\_clone (CMML\_Preamble \* *src*)**

cmml\_preamble\_clone: Create a copy of a **CMML\_Preamble**(p. 33).

**Parameters:**

*src* the original preamble

**Returns:**

a copy of a CMML preamble

**7.4.4.27 CMML\_Element\* cmml\_element\_clone (CMML\_Element \* *src*)**

cmml\_element\_clone: Create a copy of an element.

**Parameters:**

*src* the original element

**Returns:**

a copy of the element

**7.4.4.28 CMML\_Stream\* cmml\_stream\_clone (CMML\_Stream \* *src*)**

cmml\_stream\_clone: Create a copy of a stream element.

**Parameters:**

*src* the original stream element

**Returns:**

a copy of the stream element

**7.4.4.29 CMML\_Head\* cmml\_head\_clone (CMML\_Head \* *src*)**

cmml\_head\_clone: Create a copy of a head element.

**Parameters:**

*src* the original head element

**Returns:**

a copy of the head element

**7.4.4.30 CMML\_Clip\* cmml\_clip\_clone (CMML\_Clip \* *src*)**

cmml\_clip\_clone: Create a copy of a clip element.

**Parameters:**

*src* the original clip element

**Returns:**

a copy of the clip element

**7.4.4.31 void cmml\_preamble\_destroy (CMML\_Preamble \* *preamble*)**

cmml\_preamble\_destroy: Free all memory associated with a CMML\_Preamble(p.33).

**Parameters:**

*preamble* the preamble to free

**Examples:**

cmml-write.c.



**7.4.4.32 void cmml\_element\_destroy (CMML\_Element \* *element*)**

cmml\_element\_destroy: Free all memory associated with a CMML\_Element(p. 25).

**Parameters:**

*element* the element to free

**7.4.4.33 void cmml\_stream\_destroy (CMML\_Stream \* *stream*)**

cmml\_stream\_destroy: Free a stream element.

**Parameters:**

*stream* the stream element to free

**7.4.4.34 void cmml\_head\_destroy (CMML\_Head \* *head*)**

cmml\_head\_destroy: Free a head element.

**Parameters:**

*head* the head element to free

**Examples:**

cmml-write.c.

**7.4.4.35 void cmml\_clip\_destroy (CMML\_Clip \* *clip*)**

cmml\_clip\_destroy: Free a clip element.

**Parameters:**

*clip* the clip element to free

**Examples:**

cmml-write.c.

**7.4.4.36 void cmml\_error\_destroy (CMML\_Error \* *error*)**

cmml\_error\_destroy: Free an error structure.

**Parameters:**

*error* the error structure to free

**7.4.4.37** `int cmml_preamble_snprint (char * buf, int n, CMML_Preamble * pre)`

`cmml_preamble_snprint`: Prints the preamble as given in the `cmml` structure into the buffer `buf` of size `n`.

**Parameters:**

*buf* the buffer to print into  
*n* the size of the buffer `buf`  
*pre* the preamble to print

**Returns:**

the number of characters written to the buffer (including the terminating `\0`) or -1 if buffer too small.

**Examples:**

`cmml-validate.c`, and `cmml-write.c`.

**7.4.4.38** `int cmml_element_snprint (char * buf, int n, CMML_Element * elem)`

`cmml_element_snprint`: Given a buffer `buf` of size `n`, write a CMML representation of a `CMML_Element`(p. 25) into it.

**Parameters:**

*buf* the buffer to print into  
*n* the size of the buffer `buf`  
*elem* the element to print

**Returns:**

the number of characters written to the buffer (including the terminating `\0`) or -1 if buffer too small.

**7.4.4.39** `int cmml_stream_snprint (char * buf, int n, CMML_Stream * stream)`

`cmml_stream_snprint`: Prints a stream element into a buffer.

**Parameters:**

*buf* the buffer to print into  
*n* the size of the buffer `buf`  
*stream* the stream element structure

**Returns:**

the number of characters written to the buffer (including the terminating `\0`) or -1 if buffer too small.

#### 7.4.4.40 `int cmml_stream_pretty_sprint (char * buf, int n, CMML_Stream * stream)`

`cmml_stream_pretty_sprint`: Prints a stream element into a buffer.

**Parameters:**

*buf* the buffer to print into  
*n* the size of the buffer *buf*  
*stream* the stream element structure

**Returns:**

the number of characters written to the buffer (including the terminating `\0`) or -1 if buffer too small.

**Examples:**

`cmml-validate.c`.

#### 7.4.4.41 `int cmml_head_sprint (char * buf, int n, CMML_Head * head)`

`cmml_head_sprint`: Prints a head element into a buffer.

**Parameters:**

*buf* the buffer to print into  
*n* the size of the buffer *buf*  
*head* the head element structure

**Returns:**

the number of characters written to the buffer (including the terminating `\0`) or -1 if buffer too small.

**Examples:**

`cmml-write.c`.

#### 7.4.4.42 `int cmml_head_pretty_sprint (char * buf, int n, CMML_Head * head)`

`cmml_head_pretty_sprint`: Prints a head element into a buffer.

**Parameters:**

*buf* the buffer to print into  
*n* the size of the buffer *buf*  
*head* the head element structure

**Returns:**

the number of characters written to the buffer (including the terminating `\0`) or -1 if buffer too small.

**Examples:**

`cmml-validate.c`.

**7.4.4.43** `int cmml_clip_snpint (char * buf, int n, CMML_Clip * clip)`

`cmml_clip_snpint`: Prints a clip element into a buffer.

**Parameters:**

*buf* the buffer to print into  
*n* the size of the buffer *buf*  
*clip* the clip element structure

**Returns:**

the number of characters written to the buffer (including the terminating `\0`) or -1 if buffer too small.

**Examples:**

`cmml-write.c`.

**7.4.4.44** `int cmml_clip_pretty_snpint (char * buf, int n, CMML_Clip * clip)`

`cmml_clip_pretty_snpint`: Prints a clip element into a buffer.

**Parameters:**

*buf* the buffer to print into  
*n* the size of the buffer *buf*  
*clip* the clip element structure

**Returns:**

the number of characters written to the buffer (including the terminating `\0`) or -1 if buffer too small.

**Examples:**

`cmml-validate.c`.

**7.4.4.45** `int cmml_error_snpint (char * buf, int n, CMML_Error * error, CMML * cmml)`

`cmml_error_snpint`: Prints a string description of the `CMML_Error`(p. 26).

**Parameters:**

*buf* the buffer to print into  
*n* the size of the buffer *buf*  
*error* the error structure  
*cmml* the cmml status structure (required for expat errors)

**Returns:**

the number of characters written to the buffer (including the terminating `\0`) or -1 if buffer too small.

**Examples:**

`cmml-validate.c`.

**7.4.4.46** CMML\_List\* cmml\_list\_new (void)

cmml\_list\_new: Create a new list.

**Returns:**

a new list

**7.4.4.47** CMML\_List\* cmml\_list\_clone (CMML\_List \* *list*)

cmml\_list\_clone: Copy a list using the default cloning function.

**Parameters:**

*list* the list to clone

**Returns:**

a newly cloned list

**7.4.4.48** CMML\_List\* cmml\_list\_clone\_with (CMML\_List \* *list*,  
CMML\_CloneFunc *clone*)

cmml\_list\_clone\_with: Clone a list using a custom clone function.

**Parameters:**

*list* the list to clone

*clone* the function to use to clone a list item

**Returns:**

a newly cloned list

**7.4.4.49** CMML\_List\* cmml\_list\_tail (CMML\_List \* *list*)

cmml\_list\_tail: Returns the tail element of a list.

**Parameters:**

*list* the list

**Returns:**

the tail element

**7.4.4.50** CMML\_List\* cmml\_list\_prepend (CMML\_List \* *list*, void \* *data*)

cmml\_list\_prepend: Prepend a new node to a list containing given data.

**Parameters:**

*list* the list

*data* the data element of the newly created node

**Returns:**

the new list head

**7.4.4.51** `CMML_List* cmml_list_append (CMML_List * list, void * data)`

`cmml_list_append`: Append a new node to a list containing given data.

**Parameters:**

*list* the list

*data* the data element of the newly created node

**Returns:**

the head of the list

**7.4.4.52** `CMML_List* cmml_list_add_before (CMML_List * list, void * data, CMML_List * node)`

`cmml_list_add_before`: Add a new node containing given data before a given node

**Parameters:**

*list* the list

*data* the data element of the newly created node

*node* the node before which to add the newly created node

**Returns:**

the head of the list (which may have changed)

**7.4.4.53** `CMML_List* cmml_list_add_after (CMML_List * list, void * data, CMML_List * node)`

`cmml_list_add_after`: Add a new node containing given data after a given node.

**Parameters:**

*list* the list

*data* the data element of the newly created node

*node* the node after which to add the newly created node

**Returns:**

the head of the list

**7.4.4.54** `CMML_List* cmml_list_find (CMML_List * list, void * data)`

`cmml_list_find`: Find the first node containing given data in a list.

**Parameters:**

*list* the list

*data* the data element to find

**Returns:**

the first node containing given data, or NULL if it is not found

**7.4.4.55** `CMML_List* cmml_list_remove (CMML_List * list, CMML_List * node)`

`cmml_list_remove`: Remove a node from a list.

**Parameters:**

*list* the list

*node* the node to remove

**Returns:**

the head of the list (which may have changed)

**7.4.4.56** `int cmml_list_length (CMML_List * list)`

`cmml_list_length`: Query the number of items in a list.

**Parameters:**

*list* the list

**Returns:**

the number of nodes in the list

**7.4.4.57** `int cmml_list_is_empty (CMML_List * list)`

`cmml_list_is_empty`: Query if a list is empty, ie. contains no items.

**Parameters:**

*list* the list

**Returns:**

1 if the list is empty, 0 otherwise

**7.4.4.58** `int cmml_list_is_singleton (CMML_List * list)`

`cmml_list_is_singleton`: Query if the list is singleton, ie. contains exactly one item

**Parameters:**

*list* the list

**Returns:**

1 if the list is singleton, 0 otherwise

**7.4.4.59** `CMML_List* cmml_list_free_with (CMML_List * list, CMML_FreeFunc free_func)`

`cmml_list_free_with`: Free a list, using a given function to free each data element

**Parameters:**

*list* the list

*free\_func* a function to free each data element

**Returns:**

NULL on success

**7.4.4.60 CMML\_List\* cmml\_list\_free (CMML\_List \* *list*)**

cmml\_list\_free: Free a list, using `anx_free()` to free each data element.

**Parameters:**

*list* the list

**Returns:**

NULL on success else the original list.

**7.4.4.61 CMML\_Time\* cmml\_time\_new (const char \* *s*)**

cmml\_time\_new: Creates a time struct from a string that contains a name-value time spec pair.

**Parameters:**

*s* name-value time specification string

**Returns:**

a time struct with appropriate time type (utc or seconds).

**Examples:**

`cmml-write.c`.

**7.4.4.62 CMML\_Time\* cmml\_sec\_new (const char \* *s*)**

cmml\_sec\_new: Creates a time struct only from name-value sec specs.

**Parameters:**

*s* name-value time specification string

**Returns:**

a time struct.

**7.4.4.63 CMML\_Time\* cmml\_utc\_new (const char \* *s*)**

cmml\_utc\_new: Creates a time struct only from name-value utc specs.

**Parameters:**

*s* name-value time specification string

**Returns:**

a time struct.



**7.4.4.64** `int cmml_time_interval_new (const char * s, CMML_Time ** t_start, CMML_Time ** t_end)`

`cmml_time_interval_new`: Handles start and end times for clip tags, where we will only store sec offsets.

**Parameters:**

- s* the time construct; examples: npt:40-79 or smpte-25:00:20:20-00:21:30,00:40:21
- t\_start* the start time returned
- t\_end* the end time (if any) returned

**Returns:**

creates two time constructs if the string was a time range, otherwise just a *t\_start* time, and returns the number of times created or -1 otherwise.

**7.4.4.65** `CMML_Time* cmml_time_new_in_sec (const char * s, CMML_Time * ref, double base)`

`cmml_time_new_in_sec`: For parsing/converting start and end time specs in clips into a seconds representation, which is the only one libcmml stores internally for clips.

**Parameters:**

- s* the string that contains the time spec
- ref* the reference utc time if given in the stream tag
- base* the timebase in seconds as given in the stream tag

**Returns:**

a time structure in seconds

**7.4.4.66** `CMML_Time* cmml_time_utc_to_sec (CMML_Time * t, CMML_Time * ref, double base)`

`cmml_time_utc_to_sec`: Convert a time from utc to seconds with reference to a timebase

**Parameters:**

- t* the given time struct
- ref* the reference utc time if given in the stream tag
- base* the timebase in seconds as given in the stream tag

**Returns:**

a time structure in seconds

**7.4.4.67** `void cmml_time_free (CMML_Time * t)`

`cmml_time_free`: Free time struct

**Parameters:**

- t* the given time struct

**7.4.4.68 CMML\_Time\* cmml\_time\_clone (CMML\_Time \* *t*)**

cmml\_time\_clone: Copying times

**Parameters:**

*t* the given time struct

**Returns:**

the cloned time struct

**7.4.4.69 double cmml\_sec\_parse (const char \* *s*)**

cmml\_sec\_parse: for parsing sec specs with "npt:", "smpte:" etc.

**Parameters:**

*s* the string containing the name-value time specification

**Returns:**

the parsed seconds as a double

**7.4.4.70 CMML\_UTC\* cmml\_utc\_parse (const char \* *s*)**

cmml\_utc\_parse: for parsing utc times with "clock:"

**Parameters:**

*s* the string containing only the utc string as YYYYMMDDTHHmmss.hhZ

**Returns:**

a utc time structure

**7.4.4.71 CMML\_UTC\* cmml\_utc\_clone (CMML\_UTC \* *t*)**

cmml\_utc\_clone: for copying utc times

**Parameters:**

*t* the src utc time structure

**Returns:**

the copied utc time structure

**7.4.4.72 double cmml\_utc\_diff (CMML\_UTC \* *t2*, CMML\_UTC \* *t1*)**

cmml\_utc\_diff: for calculating the difference between two utc times as t2-t1 in seconds

**Parameters:**

*t2* the more recent utc time

*t1* the less recent utc time

**Returns:**

the difference between the two utc times in seconds

**7.4.4.73 int cmml\_npt\_snprint (char \* *buf*, int *n*, double *seconds*)**

cmml\_npt\_snprint: prints npt time to a string

**Parameters:**

- buf* buffer to print string to
- n* the size of the buffer *buf*
- seconds* the npt time in seconds as a double

**Returns:**

number of characters written to the buffer (including the terminating \0) or a negative value on failure.

**Examples:**

cmml-write.c.

**7.4.4.74 int cmml\_utc\_snprint (char \* *buf*, int *n*, CMML\_UTC \* *t*)**

cmml\_utc\_snprint: for printing utc times

**Parameters:**

- buf* the buffer to print into
- n* the size of the buffer *buf*
- t* the utc time structure

**Returns:**

the number of characters written to the buffer (including the terminating \0) or -1 if buffer too small.

**7.4.4.75 int cmml\_utc\_pretty\_snprint (char \* *buf*, int *n*, CMML\_UTC \* *t*)**

cmml\_utc\_pretty\_snprint: for printing utc times

**Parameters:**

- buf* the buffer to print into
- n* the size of the buffer *buf*
- t* the utc time structure

**Returns:**

the number of characters written to the buffer (including the terminating \0) or -1 if buffer too small.



## Chapter 8

# libcmml Example Documentation

### 8.1 cmml-parse.c

#### Reading from a CMML file

libcmml provides a convenient callback based framework for reading CMML files. After opening a CMML file for reading, you can attach various callbacks relevant to the parts of the file you are interested in, including the stream element, head element, and clip elements. Then, as bytes are read, libcmml will call your callbacks as appropriate.

The functions that need to be called and the respective data structures are defined in **cmml.h**(p. 47).

The general sequence of API calls is the following:

- open a CMML object using **cmml\_open()**(p. 53) on a local file or **cmml\_new()**(p. 53) with a FILE pointer to an already opened file.
- attach read callbacks using **cmml\_set\_read\_callbacks()**(p. 54)
- call **cmml\_read()**(p. 54) repeatedly until it returns 0 or -1.
- close the CMML object with **cmml\_destroy()**(p. 53) returning the FILE pointer or use **cmml\_close()**(p. 54) to close both the object and the open file.

This procedure is illustrated in **cmml-parse.c**, which prints the title attribute of a CMML tag :

```
#include <stdio.h>

#include <cmml.h>

#define BUFSIZE 100000

static int
read_head (CMML * cmml, const CMML_Head * head, void * user_data) {
    puts(head->title);
    return 0;
}

int main(int argc, char *argv[])
{
    char *filename = NULL;
```

```
FILE * cmml_file;
CMML * doc;
long n = 0;

if (argc != 2) {
    fprintf (stderr, "Usage: %s <CMMLfile>\n", argv[0]);
    exit (1);
}
filename = argv[1];

cmml_file = fopen(filename, "r");
doc = cmml_new(cmml_file);

/* alternatively use:
 * doc = cmml_open(filename);
 */

cmml_set_read_callbacks (doc, NULL, read_head, NULL, NULL);

while (((n = cmml_read (doc, BUFSIZE)) > 0));

cmml_file = cmml_destroy(doc);
fclose(cmml_file);

/* alternatively use:
 * cmml_close(doc);
 */

exit(0);
}
```

## 8.2 cmml-seek-clip.c

### Seeking to an clip in a CMML file

Sometimes you don't want to access all the clips available in a CMML file, but only a specific one. libcmml provides an API for this functionality through the `cmml_skip_to_id()` (p. 57) function.

The procedure is illustrated in `cmml-seek-clip.c`, which seeks to a clip of a given name and if found prints out the descriptions of this and all the following clips:

```
#include <stdio.h>

#include <cmml.h>

#define BUFSIZE 100000

static int
read_clip (CMML * cmml, const CMML_Clip * clip, void * user_data) {
    puts(clip->desc_text);
    return 0;
}

int main(int argc, char *argv[])
{
    char *filename = NULL;
    char *clip_id = NULL;
    CMML * doc;
    long n = 0;

    if (argc < 2) {
        fprintf (stderr, "Usage: %s <CMMLfile> <clipID>\n", argv[0]);
        exit (1);
    }
    filename = argv[1];
    clip_id = argv[2];

    doc = cmml_open(filename);

    /* seek to clip_id; if not found, to file end */
    if (clip_id != NULL) {
        cmml_skip_to_id (doc, clip_id);
    }

    cmml_set_read_callbacks (doc, NULL, NULL, read_clip, NULL);

    while ((n = cmml_read (doc, BUFSIZE)) > 0);

    cmml_close(doc);

    return 0;
}
```

## 8.3 cmml-seek-offset.c

### Seeking to a temporal offset into a CMML file

Sometimes you'll need to seek to a temporal offset in seconds into a CMML file. Note: The seconds offset is calculated with respect to the timebase attribute of the stream tag. libcmml provides an API for this functionality through the `cmml_skip_to_secs()`(p.57) function.

The procedure is illustrated in `cmml-seek-offset.c`, which seeks to an offset given in seconds and prints out the descriptions of all the following clips:

```
#include <stdio.h>

#include <cmml.h>

#define BUFSIZE 100000

static int
read_clip (CMML * cmml, const CMML_Clip * clip, void * user_data) {
    puts(clip->desc_text);
    return 0;
}

int main(int argc, char *argv[])
{
    char *filename = NULL;
    CMML * doc;
    double seconds = 0.0;
    long n = 0;

    if (argc < 2) {
        fprintf (stderr, "Usage: %s <CMMLfile> <seconds>\n", argv[0]);
        exit (1);
    }
    filename = argv[1];
    if (argv[2]) seconds = atof(argv[2]);

    doc = cmml_open(filename);

    /* seek to time offset; if not found, to file end */
    cmml_skip_to_secs (doc, seconds);

    cmml_set_read_callbacks (doc, NULL, NULL, read_clip, NULL);

    while ((n = cmml_read (doc, BUFSIZE)) > 0);

    cmml_close(doc);

    return 0;
}
```



## 8.4 cmml-seek-utc.c

### Seeking to a utc time offset into a CMML file

Sometimes you'll need to seek to a temporal offset in utc into a CMML file. Note: The utc offset is calculated with respect to the utc and timebase attributes of the stream tag. libcmml provides an API for this functionality through the `cmml_skip_to_utc()`(p. 57) function.

The procedure is illustrated in `cmml-seek-utc.c`, which seeks to an offset given in utc and prints out the descriptions of all the following clips:

```
#include <stdio.h>

#include <cmml.h>

#define BUFSIZE 100000

static int
read_clip (CMML * cmml, const CMML_Clip * clip, void * user_data) {
    puts(clip->desc_text);
    return 0;
}

int main(int argc, char *argv[])
{
    char *filename = NULL;
    CMML * doc;
    char *utc = NULL;
    long n = 0;

    if (argc < 2) {
        fprintf (stderr, "Usage: %s <CMMLfile> <utc>\n", argv[0]);
        exit (1);
    }
    filename = argv[1];
    utc = argv[2];

    doc = cmml_open(filename);

    /* seek to time offset; if not found, to file end */
    cmml_skip_to_utc (doc, utc);

    cmml_set_read_callbacks (doc, NULL, NULL, read_clip, NULL);

    while ((n = cmml_read (doc, BUFSIZE)) > 0);

    cmml_close(doc);

    return 0;
}
```

## 8.5 cmml-uri-file.c

### Parse a CMML file given through a file uri (optionally with a fragment offset)

This example program demonstrates how a CMML file that is given through a file uri and optionally contains a fragment offset can be interpreted. The example can be extended to other schemes such as http and to cover uri queries, too.

The procedure is illustrated in cmml-uri-file.c, which opens a file given through a file uri, and optionally seeks to an offset given the uri fragment specifier. It then prints out the descriptions of all the following clips:

```
#include <stdio.h>

#include <cmml.h>
#include <string.h>

#define BUFSIZE 100000

typedef struct {
    char *scheme;
    char *authority;
    char *path;
    char *querystr;
    char *fragstr;
} URI;

static URI *
parse_file_uri (const char *uri_string)
{
    const char *location;
    const char *locbegin;
    int length;
    URI *result;
    locbegin = uri_string;
    result = (URI*) calloc(sizeof(URI), sizeof(char));

    /* ignore file:// and authority parts to get path */
    location = strstr (locbegin, "://");
    locbegin = location+3;
    length = strlen(locbegin);
    location = strchr(locbegin, '#'); /* XXX: ignore queries for the moment */
    if (location != NULL) length = location - locbegin;
    result->path = (char*) calloc (length+1, sizeof(char));
    result->path = strncpy(result->path, locbegin, length);
    result->path[length] = '\0';

    if (location != NULL) {
        /* fragment given */
        length = strlen(location);
        result->fragstr = NULL;
        result->fragstr = (char*) calloc (length, sizeof(char));
        result->fragstr = strncpy(result->fragstr, location+1, length);
    } else {
        result->fragstr = NULL;
    }
    return result;
}

static int
read_clip (CMML * cmml, const CMML_Clip * clip, void * user_data) {
    puts(clip->desc_text);
    return 0;
}
```

```
int main(int argc, char *argv[])
{
    char *uri_string = NULL;
    URI * uri;
    CMML * doc;
    long n = 0;

    if (argc < 2) {
        fprintf (stderr, "Usage: %s <file://filename#fragment>\n", argv[0]);
        exit (1);
    }
    uri_string = argv[1];

    uri = parse_file_uri(uri_string);

    doc = cmml_open(uri->path);

    /* if fragment given, forward to that */
    if (uri->fragstr != NULL) cmml_skip_to_offset(doc, uri->fragstr);

    cmml_set_read_callbacks (doc, NULL, NULL, read_clip, NULL);

    while ((n = cmml_read (doc, BUFSIZE)) > 0);

    cmml_close(doc);

    exit(0);
}
```

## 8.6 cmml-validate.c

This is the full source code of the cmml-validate program, which parses a CMML instance document and validates it against the cmml.dtd returning true/false. In case of an error the faulty tag including line and col number is reported. It also spits out warnings for strange stuff.

```
/* Copyright (C) 2003 CSIRO Australia

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

- Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

- Neither the name of the CSIRO nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
‘‘AS IS’’ AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE ORGANISATION OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

#include "config.h"

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <ctype.h>
#include <string.h>
#include <errno.h>

#ifdef WIN32
#include <unistd.h>
#endif

#ifdef HAVE_GETOPT_H
#include <getopt.h>
#endif

#include <cmml.h>

#define BUFSIZE 100000

/*
#define DEBUG
*/

static int verbose;

static void
PrintUsage(char *prog) {
    fprintf(stderr, "Usage: %s [options] filename\n", prog);
```

```

    fprintf(stderr, "Validate a CMML file.\n\n");
    fprintf(stderr, "Possible options:\n");
#ifdef HAVE_GETOPT_LONG
    fprintf(stderr, "  -i clip_id, --id clip_id\n");
    fprintf(stderr, "                Start parsing from the named clip.\n");
    fprintf(stderr, "  -s seconds, --sec seconds\n");
    fprintf(stderr, "                Start parsing from the given seconds offset\n");
    fprintf(stderr, "  -u utc,      --utc utc\n");
    fprintf(stderr, "                Start parsing from the given utc time\n");
    fprintf(stderr, "  -b, --verbose Output parsed file to stdout\n");
    fprintf(stderr, "  -h, --help   Display this help information\n");
    fprintf(stderr, "  -v, --version Display version information\n");
#else
    fprintf(stderr, "  -i clip_id    Start parsing from the named clip\n");
    fprintf(stderr, "  -s seconds    Start parsing from the given seconds offset\n");
    fprintf(stderr, "  -u utc        Start parsing from the given utc time\n");
    fprintf(stderr, "  -b           Output parsed file to stdout\n");
    fprintf(stderr, "  -h           Display this help information\n");
    fprintf(stderr, "  -v           Display version information\n");
#endif
    fprintf(stderr, "\nPlease report bugs to <libcmml-devel@cmis.csiro.au>.\n");
    exit(1);
}

static int
read_stream (CMML * cmml, const CMML_Stream * stream, void * user_data) {
    char buf[BUFSIZE];
    CMML_Error * err;
    if ((err = cmml_get_last_error(cmml)) != NULL) {
        cmml_error_snprint(buf, BUFSIZE, err, cmml);
        fprintf(stderr, "cmml-validate: Parsing stream tag %s\n", buf);
        fprintf(stderr, "cmml-validate: Non-recoverable error\n");
        return -1;
    } else {
        /* print stream */
        if (verbose) {
            cmml_stream_pretty_snprint (buf, BUFSIZE, (CMML_Stream *) stream);
            fprintf(stdout, "%s\n", buf);
        }
    }
    return 0;
}

static int
read_head (CMML * cmml, const CMML_Head * head, void * user_data) {
    char buf[BUFSIZE];
    CMML_Error * err;
    if ((err = cmml_get_last_error(cmml)) != NULL) {
        cmml_error_snprint(buf, BUFSIZE, err, cmml);
        fprintf(stderr, "cmml-validate: Parsing head tag %s\n", buf);
        fprintf(stderr, "cmml-validate: Non-recoverable error\n");
        return -1;
    } else {
        if (verbose) {
            cmml_head_pretty_snprint (buf, BUFSIZE, (CMML_Head *) head);
            fprintf(stdout, "%s\n", buf);
        }
    }
    return 0;
}

static int
read_clip (CMML * cmml, const CMML_Clip * clip, void * user_data) {
    char buf[BUFSIZE];
    CMML_Error * err;
    if ((err = cmml_get_last_error(cmml)) != NULL) {
        cmml_error_snprint(buf, BUFSIZE, err, cmml);
        fprintf(stderr, "cmml-validate: Parsing clip %s\n", buf);
    }
}

```

```

    fprintf(stderr, "cmml-validate: Skipping clip\n\n");
    return -1;
} else {
    if (verbose) {
        cmml_clip_pretty_snprint (buf, BUFSIZE, (CMML_Clip *) clip);
        fprintf(stdout, "%s\n", buf);
    }
}
return 0;
}

int main(int argc, char *argv[])
{
    char *pathfile = NULL;
    int i;
    char buf[BUFSIZE];
    CMML * doc;
    CMML_Error * err;
    CMML_Preamble * pre;
    long n = 0;
    char * clip_id = NULL;
    double secs = -1.0;
    char * utc = NULL;
    int sloppy = 0;
    verbose = 0;

    while (1) {
        char * optstring = "hvbyi:s:u:";

#ifdef HAVE_GETOPT_LONG
        static struct option long_options[] = {
            {"help",no_argument,0,'h'},
            {"version",no_argument,0, 'v'},
            {"verbose",no_argument,0,'b'},
            {"sloppy",no_argument,0,'y'},
            {"id",required_argument,0,'i'},
            {"sec",required_argument,0,'s'},
            {"utc",required_argument,0,'u'},
            {0,0,0,0}
        };

        i = getopt_long(argc, argv, optstring, long_options, NULL);
#else
        i = getopt(argc, argv, optstring);
#endif

        if (i == -1) break;
        if (i == ':') PrintUsage(argv[0]);

        switch (i) {
            case 'h': /* help */
                PrintUsage(argv[0]);
                break;
            case 'v': /* version */
                fprintf(stdout, "cmml-validate version " VERSION "\n");
                fprintf(stdout, "# cmml-validate, Copyright (C) 2003 CSIRO Australia www.csiro.au ; www.annodex.net\n");
                break;
            case 'i': /* clip_id */
                clip_id = optarg;
                break;
            case 's': /* seconds */
                if (!isalpha(optarg[0])) {
                    secs = atof(optarg);
                }
                break;
            case 'u': /* utc */
                utc = optarg;

```

```

        break;
    case 'y': /* sloppy */
        sloppy = 1;
        break;
    case 'b': /* verbose */
        verbose = 1;
        break;
    default:
        break;
    }
}

/* more arguments that were not parsed */
if (optind > argc) {
    PrintUsage(argv[0]);
}

/* no filename given? */
if (optind == argc) {
    pathfile = "-";
} else {
    pathfile = argv[optind++];
}

/* try open file for parsing and setup cmml parsing */
errno=0;
if (strcmp (pathfile, "-") == 0) {
    doc = cmml_new (stdin);
} else {
    doc = cmml_open (pathfile);
}

if (doc == NULL) {
    if (errno == 0) {
        fprintf(stderr, "%s: %s: CMML error opening file\n", argv[0], pathfile);
    } else {
        fprintf(stderr, "%s: %s: %s\n", argv[0], pathfile, strerror(errno));
    }
    PrintUsage(argv[0]);
}

/* turn on sloppy parsing if requested */
if (sloppy) {
    cmml_set_sloppy(doc, 1);
}

/* print preamble */
if (verbose) {
    pre = cmml_get_preamble(doc);
    cmml_preamble_snprint(buf, BUFSIZE, pre);
    fprintf(stdout, "%s\n", buf);
}

/* seek to clip_id; if not found, to file end */
if (clip_id != NULL) {
    /* register callbacks */
    cmml_set_read_callbacks (doc, read_stream, read_head, NULL, NULL);
    cmml_skip_to_id (doc, clip_id);
}

/* seek to time offset; if not found, to file end */
if (secs > 0 || utc != NULL) {
    /* register callbacks */
    cmml_set_read_callbacks (doc, read_stream, read_head, NULL, NULL);
    if (secs > 0) {
        cmml_skip_to_secs (doc, secs);
    } else { /* if (utc != NULL) { */

```

```
        cmml_skip_to_utc (doc, utc);
    }
}

/* register callbacks */
cmml_set_read_callbacks (doc, read_stream, read_head, read_clip, NULL);

/* read document frame-wise and check against CMML.dtd */
while ((n = cmml_read (doc, BUFSIZE)) > 0) {
    /* if error reading, print and exit */
    if ((err = cmml_get_last_error(doc)) != NULL && err->type != CMML_EOF) {
        char *filename;
        filename = (strrchr(pathfile, '/') == NULL ? pathfile
                        : strrchr(pathfile, '/')+1);
        cmml_error_snprint(buf, BUFSIZE, err, doc);
        fprintf (stderr, "%s:%s\n", filename, buf);
        goto cleanup;
    }
}

err = cmml_get_last_error(doc);
if (n == -1 || (err!=NULL && err->type != CMML_EOF)) {
    char *filename;
    filename = (strrchr(pathfile, '/') == NULL ? pathfile
                    : strrchr(pathfile, '/')+1);
    cmml_error_snprint(buf, BUFSIZE, err, doc);
    fprintf (stderr, "%s:%s\n", filename, buf);
    goto cleanup;
}

/* write end tag */
if (verbose) {
    fprintf(stdout, "</cmml>\n");
}

cleanup:
/* clean up */
cmml_close(doc);

return 0;
}
```



## 8.7 cmml-write.c

### Writing a CMML file

Although libcmml was created mainly for the purpose of parsing existing CMML files, it also provides rudimentary support for creating CMML files. This basically consists in providing the data structures for the different tags and API functions to print those tags. There is no validation or sequence checking available for the writing side.

The general sequence of API calls is the following:

- create a **CMML\_Preamble**(p. 33) with **cmml\_preamble\_new()**(p. 58) and print it with **cmml\_preamble\_snprint()**(p. 62)
- create a **CMML\_Head**(p. 27) with **cmml\_head\_new()**(p. 59), change the attribute and element contents in the **CMML\_Head**(p. 27) structure, and print it with **cmml\_head\_snprint()**(p. 63) or **cmml\_head\_pretty\_snprint()**(p. 63)
- create one or more **CMML\_Clip**(p. 21) with **cmml\_clip\_new()**(p. 59), change the attribute and element contents in the **CMML\_Clip**(p. 21) structures, and print them with **cmml\_clip\_snprint()**(p. 64) or **cmml\_clip\_pretty\_snprint()**(p. 64)
- print the final "</cmml>" tag

This procedure is illustrated in cmml-write.c:

```
/* Copyright (C) 2003 CSIRO Australia

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

- Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

- Neither the name of the CSIRO nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
‘AS IS’ AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE ORGANISATION OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

#include <stdio.h>
#include <string.h>

#include <cmml.h>

#define BUFSIZE 100000
```

```
int main(int argc, char *argv[])
{
    CMML_Preamble * pre = NULL;
    CMML_Head * head = NULL;
    CMML_Clip * clip = NULL;
    CMML_Time * start = NULL, * end = NULL;
    char buf[BUFSIZE];

    /* write preamble */
    pre = cmml_preamble_new("UTF-8", NULL, NULL, NULL);
    cmml_preamble_snprint (buf, BUFSIZE, pre);
    puts(buf);
    cmml_preamble_destroy(pre);

    /* write head */
    head = cmml_head_new();
    head->title = strdup("This is a test file");
    cmml_head_snprint (buf, BUFSIZE, head);
    puts(buf);
    cmml_head_destroy(head);

    /* write a clip for the interval 0.6 to 3.5 seconds */
    cmml_npt_snprint (buf, BUFSIZE, 0.6);
    start = cmml_time_new (buf);

    cmml_npt_snprint (buf, BUFSIZE, 3.5);
    end = cmml_time_new (buf);

    clip = cmml_clip_new(start, end);
    clip->anchor_href = strdup("http://www.annodex.net/");
    clip->anchor_text = strdup("hyperlink to annodex.net");
    clip->desc_text = strdup("This is a clip");
    cmml_clip_snprint (buf, BUFSIZE, clip);
    puts (buf);
    cmml_clip_destroy(clip);

    printf ("</cmml>\n");

    exit(0);
}
```

## 8.8 cmml.dtd

The DTD for CMML is quite short and produces flat hierarchies. It follows an intention to create the minimal necessary markup and although tags were copied from HTML, attributes and elements were selected carefully. The inclusion of further tags is possible, but only when there is a proven need for it. Incompatible changes will result in a change of the major version number of CMML, while compatible changes, such as optional extensions, only change the minor version number.

To get a better understanding of the components in libcmml it is recommended to read the DTD:

```
<!--

Continuous Media Markup Language CMML version 2.0 DTD
Authoring language for ANNODEx(TM) media.

Namespace = http://www.annodex.net/cmml

Copyright (c) 2001
Commonwealth Scientific and Industrial Research Organisation
(CSIRO), Australia.
All Rights Reserved.

This DTD module is identified by the PUBLIC and SYSTEM identifiers:

PUBLIC "-//CSIRO//DTD CMML 2.0//EN"
SYSTEM "http://www.annodex.net/DTD/cmml_2_0.dtd"

$Revision: 1.2 $
$Date: 2004/01/01 05:25:33 $
-->

<!-- ***** -->
<!-- Definition of Imported Names -->
<!-- ***** -->

<!-- media type, as per [RFC2045] -->
<!ENTITY % ContentType "CDATA">

<!-- a Uniform Resource Identifier, see [RFC2396] -->
<!ENTITY % URI "CDATA">

<!-- a language code, as per [RFC1766] -->
<!ENTITY % LanguageCode "NMTOKEN">

<!-- internationalization attributes
xml:lang language code (as per XML 1.0 spec)
dir direction for weak/neutral text
-->
<!ENTITY % i18n
"lang %LanguageCode; #IMPLIED
dir (ltr|rtl) #IMPLIED"
>

<!-- timestamps similar to [RFC2326]
"smpte-24:" SMPTE time with a 24 fps basis
"smpte-24-drop:" SMPTE time with a 24/1.001 fps basis
"smpte-25:" SMPTE time with a 25 fps basis
"smpte-30:" SMPTE time with a 30 fps basis
"smpte-30-drop:" SMPTE time with a 30/1.001 fps basis
"smpte-50:" SMPTE time with a 50 fps basis
"smpte-60:" SMPTE time with a 60 fps basis
"smpte-60-drop:" SMPTE time with a 60/1.001 fps basis
"npt:" npt-time
"clock:" utc-time

Playbacktime is specified as a smpte-time
```

or npt-time only.

UTCtime is specified as in [RFC2326], but without the "clock" identifier

```
-->
<!ENTITY % Timestamp      "CDATA">
<!ENTITY % Playebacktime  "CDATA">
<!ENTITY % UTCtime        "CDATA">

<!-- ***** -->
<!-- Document Structure -->
<!-- ***** -->

<!-- ROOT ELEMENT: -->
<!-- cmml tag containing sequence of head and a tags -->
<!-- ===== -->
<!-- i18n = the default language for the whole document including
           the id tag of the cmml element -->
<!-- xmlns = namespace of the cmml tags -->
<!ELEMENT cmml (stream?, head, clip*)>
<ATTLIST cmml
  %i18n;
  id          ID              #IMPLIED
  xmlns       %URI;          #FIXED 'http://www.annodex.net/cmml'
>

<!-- ***** -->
<!-- Definition of stream element -->
<!-- ***** -->

<!-- STREAM tag providing timing information for the ANNOEX file -->
<!-- (will be stored in the binary headers of the ANX bitstreams) -->
<!-- ===== -->
<!-- (has no text attributes and thus no i18n; id tag follows default
      language specified in cmml tag) -->
<!-- timebase = base time associated with the first frame of the media
              document from which subsequent time references (such as
              in clip tags) will be taken relative to -->
<!-- utc      = a mapping of the first frame to clock time;
              specifications of utc time offsets into the document as
              in a URI will be taken relative to this -->
<!ELEMENT stream (import*)>
<ATTLIST stream
  id          ID              #IMPLIED
  timebase    %Playbacktime; "0"
  utc         %UTCtime;      #IMPLIED
>

<!-- IMPORT tag giving descriptions on an input bitstream (empty content) -->
<!-- ===== -->
<!-- i18n      = the language of the import tag's and the contained param
              tags' attribute values -->
<!-- granulerate = the base temporal resolution of the bitstream (e.g.
              its framerate for video or samplerate for audio) -->
<!-- contenttype = encoding format of the input document (a MIME type and
              a character encoding separated by semicolon) -->
<!-- src       = URI to the media document -->
<!-- start     = the start time of the media bitstream specified
              in src -->
<!-- end       = the end time of the media bitstream specified
              in src -->
<!-- title     = human readable comment on the import bitstream -->
<!ELEMENT import (param*)>
<ATTLIST import
  %i18n;
```

```

    id          ID          #IMPLIED
    granulate   CDATA       #IMPLIED
    contenttype %ContentType; #IMPLIED
    src         %URI;       #REQUIRED
    start       %Timestamp;  "0"
    end         %Timestamp;  #IMPLIED
    title       CDATA       #IMPLIED
  >

<!-- PARAM description tags of an input bitstream (empty content) -->
<!-- (name-value pairs e.g. comments on recording quality or so) -->
<!-- ===== -->
<!-- (internationalisation inherited from the parent import tag) -->
<!-- name = identifies a property name; does not list legal values for this
        attribute -->
<!-- value = specifies a property's value; does not list legal values for
        this attribute -->
<!ELEMENT param EMPTY>
<ATTLIST param
  id          ID          #IMPLIED
  name       CDATA       #REQUIRED
  value      CDATA       #REQUIRED
>

<!-- ***** -->
<!-- Definition of document head -->
<!-- ***** -->

<!-- head tag containing description of a specific media document -->
<!-- ===== -->
<!-- i18n = the base language of the head's attribute values and text
        content -->
<!-- profile = space-separated list of URIs to locate meta tag schemes -->
<!ELEMENT head (meta*,
  ((title, meta*, (base, meta*)?) |
  (base, meta*, (title, meta*)?)))>
<ATTLIST head
  %i18n;
  id          ID          #IMPLIED
  profile     %URI;       #IMPLIED
>

<!-- TITLE tag giving descriptive title of the media document -->
<!-- ===== -->
<!-- i18n = the language of the title text -->
<!ELEMENT title (#PCDATA)>
<ATTLIST title
  %i18n;
  id          ID          #IMPLIED
>

<!-- BASE URI of the document (empty content) -->
<!-- ===== -->
<!-- (internationalisation inherited from the parent head tag) -->
<!-- href = URI associated with the document; all relative URI references
        get interpreted relative to this base -->
<!ELEMENT base EMPTY>
<ATTLIST base
  id          ID          #IMPLIED
  href       %URI;       #REQUIRED
>

<!-- META description tags of the document (empty content) -->
<!-- ===== -->
<!-- i18n = the language of the meta attributes -->
<!-- name = identifies a property name; does not list legal values for this

```

```

        attribute -->
<!-- content = specifies a property's value; does not list legal values for
        this attribute -->
<!-- scheme = names a scheme to be used to interpret the property's value
        (see the profiles tag in the head element for locating these) -->
<!ELEMENT meta EMPTY>
<!-- ATTLIST meta
%is18n;
    id          ID          #IMPLIED
    name        NMTOKEN     #IMPLIED
    content     CDATA       #REQUIRED
    scheme      CDATA       #IMPLIED
>

<!-- ***** -->
<!-- Definition of clip tags -->
<!-- ***** -->

<!-- Clip tag containing information for a specific fragment -->
<!-- ===== -->
<!-- though meta, a, img and desc are given in specific order
        here, their order is acutally random -->
<!-- i18n    = the base language of the clip's attribute values and
        of its content elements -->
<!-- id      = name of the clip used in URI clip references -->
<!-- track   = defines different sets of clip tags; clip tags of same
        type cannot overlap temporally -->
<!-- start   = specifies the start time of the clip; specified in
        time relative to the timebase of the header
        [NOT INCLUDED IN ANNOEX DOCUMENT] -->
<!-- end     = specifies the end time of the clip; specified in
        time relative to the timebase of the header
        [NOT INCLUDED IN ANNOEX DOCUMENT] -->
<!ELEMENT clip (meta*, a?, img?, desc?)>
<!-- ATTLIST clip
%is18n;
    id          ID          #IMPLIED
    track        CDATA      "default"
    start        %Timestamp; #REQUIRED
    end          %Timestamp; #IMPLIED
>

<!-- A tag containing information for a specific clip -->
<!-- ===== -->
<!-- a tag contains anchor text being a textual description of the link
        between the current element (the source anchor) and the destination
        anchor given by the href attribute -->
<!-- i18n    = language of the anchor's attribute values and anchor text -->
<!-- class   = overriding style sheet defaults for this instance -->
<!-- href    = specifies the location of a Web resource, thus defining a
        link between the current element (the source anchor) and the
        destination anchor given by this attribute -->
<!ELEMENT a (#PCDATA)>
<!-- ATTLIST a
%is18n;
    id          ID          #IMPLIED
    class        CDATA      #IMPLIED
    href         %URI;      #REQUIRED
>

<!-- IMG tag to include a representative image for the clip -->
<!-- ===== -->
<!-- i18n = language of the image's attribute values -->
<!-- src  = reference to the image -->
<!-- alt  = alternative text for the image (accessibility) -->
<!ELEMENT img EMPTY>
<!-- ATTLIST img

```

```
%i18n;
id          ID          #IMPLIED
src         %URI;       #REQUIRED
alt         CDATA       #IMPLIED
>

<!-- DESC human-readable, textual description of the clip (annotation) -->
<!-- ===== -->
<!-- i18n = language of the data in the description -->
<!ELEMENT desc (#PCDATA)>
<!-- ATTLIST desc
% i18n;
id          ID          #IMPLIED
-->
```

# Index

- `_CMML_Element_Type`
    - `cmml.h`, 51
  - `_CMML_Error_Type`
    - `cmml.h`, 52
  - `_CMML_List`, 19
    - `data`, 19
    - `next`, 19
    - `prev`, 19
  - `_CMML_Time_Type`
    - `cmml.h`, 51
- `anchor_class`
  - `CMML_Clip`, 22
- `anchor_dir`
  - `CMML_Clip`, 22
- `anchor_href`
  - `CMML_Clip`, 22
- `anchor_id`
  - `CMML_Clip`, 22
- `anchor_lang`
  - `CMML_Clip`, 22
- `anchor_text`
  - `CMML_Clip`, 22
- `base_href`
  - `CMML_Head`, 28
- `base_id`
  - `CMML_Head`, 28
- `BUFSIZE`
  - `cmml-fix.c`, 40
  - `cmml-fortune.c`, 43
  - `cmml-validate.c`, 45
- `clip`
  - `CMML_Element`, 25
- `clip_id`
  - `CMML_Clip`, 21
- `CMML`
  - `cmml.h`, 50
- `cmml-fix.c`, 39
  - `BUFSIZE`, 40
  - `main`, 41
  - `outfile`, 41
  - `PrintUsage`, 40
  - `read_clip`, 41
  - `read_head`, 40
  - `read_stream`, 40
- `cmml-fortune.c`, 42
  - `BUFSIZE`, 43
  - `DEFAULT_DURATION`, 43
  - `DEFAULT_ENCODING`, 43
  - `DEFAULT_LONG_COMMAND`, 43
  - `DEFAULT_SHORT_COMMAND`, 43
  - `get_fortune`, 43
  - `main`, 43
  - `PrintUsage`, 43
- `cmml-validate.c`, 44
  - `BUFSIZE`, 45
  - `main`, 46
  - `PrintUsage`, 45
  - `read_clip`, 45
  - `read_head`, 45
  - `read_stream`, 45
  - `verbose`, 46
- `cmml.h`, 47
  - `_CMML_Element_Type`, 51
  - `_CMML_Error_Type`, 52
  - `_CMML_Time_Type`, 51
  - `CMML`, 50
  - `CMML_CLIP`, 52
  - `cmml_clip_clone`, 60
  - `cmml_clip_destroy`, 61
  - `cmml_clip_new`, 59
  - `cmml_clip_pretty_snprint`, 64
  - `cmml_clip_snprint`, 63
  - `CMML_CloneFunc`, 50
  - `cmml_close`, 53
  - `CMML_CMML`, 52
  - `CMML_CmpFunc`, 50
  - `cmml_destroy`, 53
  - `CMML_DUPLICATE_HEAD`, 52
  - `CMML_DUPLICATE_STREAM`, 52
  - `cmml_element_clone`, 59
  - `cmml_element_destroy`, 60
  - `cmml_element_new`, 58
  - `cmml_element_snprint`, 62
  - `CMML_Element_Type`, 50
  - `CMML_EOF`, 52
  - `cmml_error_clear`, 56
  - `cmml_error_destroy`, 61



- cmml\_error\_new, 59
- cmml\_error\_snprint, 64
- CMML\_Error\_Type, 50
- CMML\_EXPAT\_ERROR, 52
- CMML\_FORMAT\_ERROR, 52
- CMML\_FreeFunc, 50
- cmml\_get\_last\_clip, 55
- cmml\_get\_last\_error, 56
- cmml\_get\_last\_head, 55
- cmml\_get\_last\_stream, 55
- cmml\_get\_preamble, 55
- cmml\_get\_previous\_clip, 56
- CMML\_HEAD, 52
- CMML\_HEAD\_AFTER\_CLIP, 52
- cmml\_head\_clone, 60
- cmml\_head\_destroy, 61
- cmml\_head\_new, 58
- cmml\_head\_pretty\_snprint, 63
- cmml\_head\_snprint, 63
- CMML\_IMPORT, 52
- CMML\_List, 50
- cmml\_list\_add\_after, 66
- cmml\_list\_add\_before, 66
- cmml\_list\_append, 65
- cmml\_list\_clone, 65
- cmml\_list\_clone\_with, 65
- cmml\_list\_find, 66
- cmml\_list\_free, 68
- cmml\_list\_free\_with, 67
- cmml\_list\_is\_empty, 67
- cmml\_list\_is\_singleton, 67
- cmml\_list\_length, 67
- cmml\_list\_new, 64
- cmml\_list\_prepend, 65
- cmml\_list\_remove, 66
- cmml\_list\_tail, 65
- CMML\_MALLOC\_ERROR, 52
- cmml\_new, 53
- CMML\_NO\_CMML\_TAG, 52
- CMML\_NO\_HEAD\_TAG, 52
- CMML\_NONE, 52
- CMML\_NONSEQUENTIAL\_CLIP, 52
- cmml\_npt\_snprint, 70
- CMML\_OK, 52
- cmml\_open, 53
- CMML\_PARSE\_ERROR, 52
- cmml\_preamble\_clone, 59
- cmml\_preamble\_destroy, 60
- cmml\_preamble\_new, 58
- cmml\_preamble\_snprint, 61
- cmml\_read, 54
- CMML\_READ\_ERROR, 52
- cmml\_sec\_new, 68
- cmml\_sec\_parse, 70
- CMML\_SEC\_TIME, 51
- cmml\_set\_read\_callbacks, 54
- cmml\_set\_sloppy, 54
- cmml\_set\_window, 56
- cmml\_skip\_to\_id, 57
- cmml\_skip\_to\_offset, 57
- cmml\_skip\_to\_secs, 57
- cmml\_skip\_to\_utc, 57
- CMML\_STREAM, 52
- cmml\_stream\_clone, 60
- cmml\_stream\_destroy, 61
- cmml\_stream\_new, 58
- CMML\_STREAM\_NOT\_FIRST, 52
- cmml\_stream\_pretty\_snprint, 62
- cmml\_stream\_snprint, 62
- CMML\_TAG\_IGNORED, 52
- cmml\_time\_clone, 69
- CMML\_TIME\_ERROR, 52
- cmml\_time\_free, 69
- cmml\_time\_interval\_new, 68
- cmml\_time\_new, 68
- cmml\_time\_new\_in\_sec, 69
- CMML\_Time\_Type, 50
- cmml\_time\_utc\_to\_sec, 69
- CMML\_UNKNOWN\_TAG, 52
- cmml\_utc\_clone, 70
- cmml\_utc\_diff, 70
- cmml\_utc\_new, 68
- cmml\_utc\_parse, 70
- cmml\_utc\_pretty\_snprint, 71
- cmml\_utc\_snprint, 71
- CMML\_UTC\_TIME, 51
- CMML\_XMLNS\_REDEFINED, 52
- CMMLReadClip, 51
- CMMLReadHead, 51
- CMMLReadStream, 50
- CMML\_CLIP
  - cmml.h, 52
- CMML\_Clip, 21
  - anchor\_class, 22
  - anchor\_dir, 22
  - anchor\_href, 22
  - anchor\_id, 22
  - anchor\_lang, 22
  - anchor\_text, 22
  - clip\_id, 21
  - desc\_dir, 23
  - desc\_id, 23
  - desc\_lang, 23
  - desc\_text, 23
  - dir, 21
  - end\_time, 22
  - img\_alt, 23

- img\_dir, 23
- img\_id, 23
- img\_lang, 23
- img\_src, 23
- lang, 21
- meta, 22
- start\_time, 22
- track, 22
- cmml\_clip\_clone
  - cmml.h, 60
- cmml\_clip\_destroy
  - cmml.h, 61
- cmml\_clip\_new
  - cmml.h, 59
- cmml\_clip\_pretty\_snprint
  - cmml.h, 64
- cmml\_clip\_snprint
  - cmml.h, 63
- CMML\_CloneFunc
  - cmml.h, 50
- cmml\_close
  - cmml.h, 53
- CMML\_CMML
  - cmml.h, 52
- CMML\_CmpFunc
  - cmml.h, 50
- cmml\_destroy
  - cmml.h, 53
- cmml\_dir
  - CMML\_Preamble, 33
- CMML\_DUPLICATE\_HEAD
  - cmml.h, 52
- CMML\_DUPLICATE\_STREAM
  - cmml.h, 52
- CMML\_Element, 25
  - clip, 25
  - e, 25
  - head, 25
  - stream, 25
  - type, 25
- cmml\_element\_clone
  - cmml.h, 59
- cmml\_element\_destroy
  - cmml.h, 60
- cmml\_element\_new
  - cmml.h, 58
- cmml\_element\_snprint
  - cmml.h, 62
- CMML\_Element\_Type
  - cmml.h, 50
- CMML\_EOF
  - cmml.h, 52
- CMML\_Error, 26
  - col, 26
  - line, 26
  - type, 26
- cmml\_error\_clear
  - cmml.h, 56
- cmml\_error\_destroy
  - cmml.h, 61
- cmml\_error\_new
  - cmml.h, 59
- cmml\_error\_snprint
  - cmml.h, 64
- CMML\_Error\_Type
  - cmml.h, 50
- CMML\_EXPAT\_ERROR
  - cmml.h, 52
- CMML\_FORMAT\_ERROR
  - cmml.h, 52
- CMML\_FreeFunc
  - cmml.h, 50
- cmml\_get\_last\_clip
  - cmml.h, 55
- cmml\_get\_last\_error
  - cmml.h, 56
- cmml\_get\_last\_head
  - cmml.h, 55
- cmml\_get\_last\_stream
  - cmml.h, 55
- cmml\_get\_preamble
  - cmml.h, 55
- cmml\_get\_previous\_clip
  - cmml.h, 56
- CMML\_HEAD
  - cmml.h, 52
- CMML\_Head, 27
  - base\_href, 28
  - base\_id, 28
  - dir, 27
  - head\_id, 27
  - lang, 27
  - meta, 28
  - profile, 27
  - title, 27
  - title\_dir, 28
  - title\_id, 27
  - title\_lang, 28
- CMML\_HEAD\_AFTER\_CLIP
  - cmml.h, 52
- cmml\_head\_clone
  - cmml.h, 60
- cmml\_head\_destroy
  - cmml.h, 61
- cmml\_head\_new
  - cmml.h, 58
- cmml\_head\_pretty\_snprint
  - cmml.h, 63

- cmml\_head\_sprintf
  - cmml.h, 63
- cmml\_id
  - CMML\_Preamble, 33
- CMML\_IMPORT
  - cmml.h, 52
- CMML\_ImportElement, 29
- CMML\_ImportElement
  - contenttype, 29
  - dir, 29
  - end\_time, 30
  - granulrate, 29
  - id, 29
  - lang, 29
  - param, 30
  - src, 29
  - start\_time, 29
  - title, 30
- cmml\_lang
  - CMML\_Preamble, 33
- CMML\_List
  - cmml.h, 50
- cmml\_list\_add\_after
  - cmml.h, 66
- cmml\_list\_add\_before
  - cmml.h, 66
- cmml\_list\_append
  - cmml.h, 65
- cmml\_list\_clone
  - cmml.h, 65
- cmml\_list\_clone\_with
  - cmml.h, 65
- cmml\_list\_find
  - cmml.h, 66
- cmml\_list\_free
  - cmml.h, 68
- cmml\_list\_free\_with
  - cmml.h, 67
- cmml\_list\_is\_empty
  - cmml.h, 67
- cmml\_list\_is\_singleton
  - cmml.h, 67
- cmml\_list\_length
  - cmml.h, 67
- cmml\_list\_new
  - cmml.h, 64
- cmml\_list\_prepend
  - cmml.h, 65
- cmml\_list\_remove
  - cmml.h, 66
- cmml\_list\_tail
  - cmml.h, 65
- CMML\_MALLOC\_ERROR
  - cmml.h, 52
- CMML\_MetaElement, 31
- CMML\_MetaElement
  - content, 31
  - dir, 31
  - id, 31
  - lang, 31
  - name, 31
  - scheme, 31
- cmml\_new
  - cmml.h, 53
- CMML\_NO\_CMML\_TAG
  - cmml.h, 52
- CMML\_NO\_HEAD\_TAG
  - cmml.h, 52
- CMML\_NONE
  - cmml.h, 52
- CMML\_NONSEQUENTIAL\_CLIP
  - cmml.h, 52
- cmml\_npt\_sprintf
  - cmml.h, 70
- CMML\_OK
  - cmml.h, 52
- cmml\_open
  - cmml.h, 53
- CMML\_ParamElement, 32
- CMML\_ParamElement
  - id, 32
  - name, 32
  - value, 32
- CMML\_PARSE\_ERROR
  - cmml.h, 52
- CMML\_Preamble, 33
  - cmml\_dir, 33
  - cmml\_id, 33
  - cmml\_lang, 33
  - cmml\_xmlns, 34
  - doctype\_declared, 33
  - xml\_encoding, 33
  - xml\_standalone, 33
  - xml\_version, 33
- cmml\_preamble\_clone
  - cmml.h, 59
- cmml\_preamble\_destroy
  - cmml.h, 60
- cmml\_preamble\_new
  - cmml.h, 58
- cmml\_preamble\_sprintf
  - cmml.h, 61
- cmml\_read
  - cmml.h, 54
- CMML\_READ\_ERROR
  - cmml.h, 52
- cmml\_sec\_new
  - cmml.h, 68

- cmml\_sec\_parse
  - cmml.h, 70
- CMML\_SEC\_TIME
  - cmml.h, 51
- cmml\_set\_read\_callbacks
  - cmml.h, 54
- cmml\_set\_sloppy
  - cmml.h, 54
- cmml\_set\_window
  - cmml.h, 56
- cmml\_skip\_to\_id
  - cmml.h, 57
- cmml\_skip\_to\_offset
  - cmml.h, 57
- cmml\_skip\_to\_secs
  - cmml.h, 57
- cmml\_skip\_to\_utc
  - cmml.h, 57
- CMML\_STREAM
  - cmml.h, 52
- CMML\_Stream, 35
  - id, 35
  - import, 35
  - timebase, 35
  - utc, 35
- cmml\_stream\_clone
  - cmml.h, 60
- cmml\_stream\_destroy
  - cmml.h, 61
- cmml\_stream\_new
  - cmml.h, 58
- CMML\_STREAM\_NOT\_FIRST
  - cmml.h, 52
- cmml\_stream\_pretty\_snprint
  - cmml.h, 62
- cmml\_stream\_snprint
  - cmml.h, 62
- CMML\_TAG\_IGNORED
  - cmml.h, 52
- CMML\_Time, 36
  - sec, 36
  - t, 36
  - tstr, 36
  - type, 36
  - utc, 36
- cmml\_time\_clone
  - cmml.h, 69
- CMML\_TIME\_ERROR
  - cmml.h, 52
- cmml\_time\_free
  - cmml.h, 69
- cmml\_time\_interval\_new
  - cmml.h, 68
- cmml\_time\_new
  - cmml.h, 68
- cmml\_time\_new\_in\_sec
  - cmml.h, 69
- CMML\_Time\_Type
  - cmml.h, 50
- cmml\_time\_utc\_to\_sec
  - cmml.h, 69
- CMML\_UNKNOWN\_TAG
  - cmml.h, 52
- CMML\_UTC, 37
  - tm\_hour, 37
  - tm\_hsec, 37
  - tm\_mday, 37
  - tm\_min, 37
  - tm\_mon, 37
  - tm\_sec, 37
  - tm\_year, 37
- cmml\_utc\_clone
  - cmml.h, 70
- cmml\_utc\_diff
  - cmml.h, 70
- cmml\_utc\_new
  - cmml.h, 68
- cmml\_utc\_parse
  - cmml.h, 70
- cmml\_utc\_pretty\_snprint
  - cmml.h, 71
- cmml\_utc\_snprint
  - cmml.h, 71
- CMML\_UTC\_TIME
  - cmml.h, 51
- cmml\_xmlns
  - CMML\_Preamble, 34
- CMML\_XMLNS\_REDEFINED
  - cmml.h, 52
- CMMLReadClip
  - cmml.h, 51
- CMMLReadHead
  - cmml.h, 51
- CMMLReadStream
  - cmml.h, 50
- col
  - CMML\_Error, 26
- Concepts of time in CMML, 13
- content
  - CMML\_MetaElement, 31
- contenttype
  - CMML\_ImportElement, 29
- data
  - \_CMML\_List, 19
- DEFAULT\_DURATION
  - cmml-fortune.c, 43
- DEFAULT\_ENCODING

- cmml-fortune.c, 43
- DEFAULT\_LONG\_COMMAND
  - cmml-fortune.c, 43
- DEFAULT\_SHORT\_COMMAND
  - cmml-fortune.c, 43
- desc\_dir
  - CMML\_Clip, 23
- desc\_id
  - CMML\_Clip, 23
- desc\_lang
  - CMML\_Clip, 23
- desc\_text
  - CMML\_Clip, 23
- dir
  - CMML\_Clip, 21
  - CMML\_Head, 27
  - CMML\_ImportElement, 29
  - CMML\_MetaElement, 31
- doctype\_declared
  - CMML\_Preamble, 33
- e
  - CMML\_Element, 25
- end\_time
  - CMML\_Clip, 22
  - CMML\_ImportElement, 30
- get\_fortune
  - cmml-fortune.c, 43
- granulate
  - CMML\_ImportElement, 29
- head
  - CMML\_Element, 25
- head\_id
  - CMML\_Head, 27
- id
  - CMML\_ImportElement, 29
  - CMML\_MetaElement, 31
  - CMML\_ParamElement, 32
  - CMML\_Stream, 35
- img\_alt
  - CMML\_Clip, 23
- img\_dir
  - CMML\_Clip, 23
- img\_id
  - CMML\_Clip, 23
- img\_lang
  - CMML\_Clip, 23
- img\_src
  - CMML\_Clip, 23
- import
  - CMML\_Stream, 35
- Internationalisation support in CMML, 15
- lang
  - CMML\_Clip, 21
  - CMML\_Head, 27
  - CMML\_ImportElement, 29
  - CMML\_MetaElement, 31
- line
  - CMML\_Error, 26
- main
  - cmml-fix.c, 41
  - cmml-fortune.c, 43
  - cmml-validate.c, 46
- meta
  - CMML\_Clip, 22
  - CMML\_Head, 28
- Multitrack annotations in CMML, 18
- Multitrack input media in CMML, 17
- name
  - CMML\_MetaElement, 31
  - CMML\_ParamElement, 32
- next
  - \_CMML\_List, 19
- outfile
  - cmml-fix.c, 41
- param
  - CMML\_ImportElement, 30
- prev
  - \_CMML\_List, 19
- PrintUsage
  - cmml-fix.c, 40
  - cmml-fortune.c, 43
  - cmml-validate.c, 45
- profile
  - CMML\_Head, 27
- read\_clip
  - cmml-fix.c, 41
  - cmml-validate.c, 45
- read\_head
  - cmml-fix.c, 40
  - cmml-validate.c, 45
- read\_stream
  - cmml-fix.c, 40
  - cmml-validate.c, 45
- scheme
  - CMML\_MetaElement, 31
- sec
  - CMML\_Time, 36
- src

- CMML\_ImportElement, 29
- start\_time
  - CMML\_Clip, 22
  - CMML\_ImportElement, 29
- stream
  - CMML\_Element, 25
- t
  - CMML\_Time, 36
- timebase
  - CMML\_Stream, 35
- title
  - CMML\_Head, 27
  - CMML\_ImportElement, 30
- title\_dir
  - CMML\_Head, 28
- title\_id
  - CMML\_Head, 27
- title\_lang
  - CMML\_Head, 28
- tm\_hour
  - CMML\_UTC, 37
- tm\_hsec
  - CMML\_UTC, 37
- tm\_mday
  - CMML\_UTC, 37
- tm\_min
  - CMML\_UTC, 37
- tm\_mon
  - CMML\_UTC, 37
- tm\_sec
  - CMML\_UTC, 37
- tm\_year
  - CMML\_UTC, 37
- track
  - CMML\_Clip, 22
- tstr
  - CMML\_Time, 36
- type
  - CMML\_Element, 25
  - CMML\_Error, 26
  - CMML\_Time, 36
- utc
  - CMML\_Stream, 35
  - CMML\_Time, 36
- value
  - CMML\_ParamElement, 32
- verbose
  - cmml-validate.c, 46
- Writing CMML files, 11
- xml\_encoding
- CMML\_Preamble, 33
- xml\_standalone
  - CMML\_Preamble, 33
- xml\_version
  - CMML\_Preamble, 33