libcmml Reference Manual

0.9.0

Generated by Doxygen 1.3.8

Mon Apr 18 13:03:36 2005

Contents

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-timeshift`, which shifts the start and end times of all the clip tags by a second value
- `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 unstructed 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: `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.
```

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.

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

Contributions:
Zen Kavanagh <ogg@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:

Chapter 3

libcmml Data Structure Index

3.1 libcmml Data Structures

Here are the data structures with brief descriptions:

Chapter 4

libcmml File Index

4.1 libcmml File List

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

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" dir="ltr" id="simple" granulerate="1000/1">

<stream id="fish" basetime="0">
  <import id="videosrc" lang="en" dir="ltr" title="Video fish"
          granulerate="25/1" contenttype="video/mpeg"
          src="fish.mpg" start="0" end="360">
    <param id="vheight" name="video.height" value="250"/>
    <param id="vwidth"  name="video.width"  value="180"/>
  </import>
</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">
  <img src="dolphin.jpg"/>
  <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>
  <img src="http://www.blah.au/goldfish.jpg"/>
  <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 Basetime attribute

The basetime 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 basetime 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 basetime given in the stream tag (or just greater than 0 if there is not basetime 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 basetime="npt:0">
  <import contenttype="video/mpeg" src="fish.mpg" start="npt:0"/>
</stream>

<head lang="en" dir="ltr" profile="http://www.annodex.net/profile.tst">
  <title lang="en" dir="ltr" id="fishtitle">Types of fish</title>
  <base id="base" href="http://www.annodex.net/"/>
  <meta name="Producer" content="Joe Ordinary"/>
  <meta name="DC.Author" content="Joe's friend"/>
  <meta id="dc" lang="en" dir="ltr" name="DC.DESCRIPTION"
        content="A composition of fish" scheme="Dublin Core"/>
  <link id="linktest" lang="en" dir="ltr" title="test the link tag"
        class="head.link" href="i18n.css" type="stylesheet"
        rel="stylesheet" rev="dunno" media="screen"/>
</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 class="clip.a" title="there's always more fish"
     href="http://www.blah.au/morefish.anx?id=goldfish">More video clips on goldfish.</a>
  <img class="clip.img" title="sweety" src="http://www.blah.au/goldfish.jpg"/>
  <desc class="clip.desc" title="tanked fish">Joe has a fishtank at home with many colourful fish. The common goldfish
  <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 atttribute 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 basetime="npt:0">
  <import contenttype="video/x-theora" src="fish.ogg" start="npt:0"/>
  <import contenttype="audio/x-vorbis" src="fishsound.ogg" start="npt:0"/>
  <import contenttype="audio/x-speex" src="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 * **class**
- char * **title**
- char * **lang**
- char * **dir**
- char * **track**
- **CMML_Time** * **start_time**
- **CMML_Time** * **end_time**
- **CMML_List** * **meta**
- char * **anchor_id**
- char * **anchor_class**
- char * **anchor_title**
- char * **anchor_lang**
- char * **anchor_dir**
- char * **anchor_href**
- char * **anchor_text**
- char * **img_id**
- char * **img_class**
- char * **img_title**
- char * **img_lang**
- char * **img_dir**
- char * **img_src**
- char * **img_alt**
- char * **desc_id**
- char * **desc_class**
- char * **desc_title**
- 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::class

class attribute of clip

6.2.2.3 char* CMML_Clip::title

title attribute of clip

6.2.2.4 char* CMML_Clip::lang

language attribute of desc element

6.2.2.5 char* CMML_Clip::dir

directionality of lang

6.2.2.6 char* CMML_Clip::track

track attribute of clip

6.2.2.7 CMML_Time* CMML_Clip::start_time

start time of this clip

6.2.2.8 CMML_Time* CMML_Clip::end_time

end time of this clip

6.2.2.9 CMML_List* CMML_Clip::meta

list of meta elements

6.2.2.10 char* CMML_Clip::anchor_id

id attribute of anchor

6.2.2.11 char* CMML_Clip::anchor_class

class attribute of clip

6.2.2.12 char* CMML_Clip::anchor_title

title attribute of clip

6.2.2.13 char* CMML_Clip::anchor_lang

language of anchor

6.2.2.14 char* CMML_Clip::anchor_dir

directionality of lang

6.2.2.15 char* CMML_Clip::anchor_href

href out of clip

Examples:
cmml-write.c.

6.2.2.16 char* CMML_Clip::anchor_text

anchor text

Examples:
cmml-write.c.

6.2.2.17 char* CMML_Clip::img_id

id attribute of image

6.2.2.18 char* CMML_Clip::img_class

class attribute of image

6.2.2.19 char* CMML_Clip::img_title

title attribute of image

6.2.2.20 char* CMML_Clip::img_lang

language of img

6.2.2.21 char* CMML_Clip::img_dir

directionality of lang

6.2.2.22 char* CMML_Clip::img_src

keyframe image of clip

6.2.2.23 char* CMML_Clip::img_alt

alternate text for image

6.2.2.24 char* CMML_Clip::desc_id

id attribute of desc

6.2.2.25 char* CMML_Clip::desc_class

class attribute of desc

6.2.2.26 char* CMML_Clip::desc_title

title attribute of desc

6.2.2.27 char* CMML_Clip::desc_lang

language attribute of desc element

6.2.2.28 char* CMML_Clip::desc_dir

directionality of lang

6.2.2.29 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 occured.

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 occured

6.4.2.3 long CMML_Error::col

column in which error occured

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**
- **CMML_List** * **link**

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

6.5.2.12 CMML_List* CMML_Head::link

list of link 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_LinkElement Struct Reference

`#include <cmml.h>`

Data Fields

- char * **id**
- char * **class**
- char * **title**
- char * **lang**
- char * **dir**
- char * **href**
- char * **type**
- char * **rel**
- char * **rev**
- char * **media**

6.7.1 Detailed Description

CMML_LinkElement: a struct to keep the details of the link tag in either a head or a clip

6.7.2 Field Documentation

6.7.2.1 char* CMML_LinkElement::id

id attribute of link

6.7.2.2 char* CMML_LinkElement::class

class attribute of link

6.7.2.3 char* CMML_LinkElement::title

title attribute of link

6.7.2.4 char* CMML_LinkElement::lang

language code of link

6.7.2.5 char* CMML_LinkElement::dir

directionality of lang (ltr/rtl)

6.7.2.6 char* CMML_LinkElement::href

href attribute of link

6.7.2.7 char* CMML_LinkElement::type

type attribute of link

6.7.2.8 char* CMML_LinkElement::rel

rel attribute of link

6.7.2.9 char* CMML_LinkElement::rev

rev attribute of link

6.7.2.10 char* CMML_LinkElement::media

media attribute of link

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

- **cmml.h**

6.8 CMML_MetaElement Struct Reference

`#include <cmml.h>`

Data Fields

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

6.8.1 Detailed Description

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

6.8.2 Field Documentation

6.8.2.1 char* CMML_MetaElement::id

id attribute of meta element

6.8.2.2 char* CMML_MetaElement::lang

language code of meta element

6.8.2.3 char* CMML_MetaElement::dir

directionality of lang (ltr/rtl)

6.8.2.4 char* CMML_MetaElement::name

property name of meta element

6.8.2.5 char* CMML_MetaElement::content

property value of meta element

6.8.2.6 char* CMML_MetaElement::scheme

scheme name of meta element

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

- **cmml.h**

6.9 CMML_ParamElement Struct Reference

`#include <cmml.h>`

Data Fields

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

6.9.1 Detailed Description

CMML_ParamElement: a struct to keep the details of the param tag in a import element

6.9.2 Field Documentation

6.9.2.1 char* CMML_ParamElement::id

id of param element

6.9.2.2 char* CMML_ParamElement::name

property name of param element

6.9.2.3 char* CMML_ParamElement::value

property value of param element

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

- **cmml.h**

6.10 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**
- char * **cmml_granulerate**

6.10.1 Detailed Description

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

Examples:
cmml-validate.c, and **cmml-write.c**.

6.10.2 Field Documentation

6.10.2.1 char* CMML_Preamble::xml_version

version attribute of xml proc instr

6.10.2.2 char* CMML_Preamble::xml_encoding

encoding attribute of xml proc instr

6.10.2.3 int CMML_Preamble::xml_standalone

standalone attribute of xml proc instr

6.10.2.4 int CMML_Preamble::doctype_declared

was doctype declared

6.10.2.5 char* CMML_Preamble::cmml_lang

lang attribute of cmml tag

6.10.2.6 char* CMML_Preamble::cmml_dir

dir attribute of cmml tag

6.10.2.7 char* CMML_Preamble::cmml_id

id attribute of cmml tag

6.10.2.8 char* CMML_Preamble::cmml_xmlns

xmlns attribute of cmml tag

6.10.2.9 char* CMML_Preamble::cmml_granulerate

granulerate attribute of cmml tag

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

- **cmml.h**

6.11 CMML_Stream Struct Reference

`#include <cmml.h>`

Data Fields

- char * **id**
- **CMML_Time * basetime**
- **CMML_Time * utc**
- **CMML_List * import**

6.11.1 Detailed Description

CMML_Stream: an optional stream element containing information on the set of import bitstreams which are being annotated by the CMML data.

Examples:
cmml-validate.c.

6.11.2 Field Documentation

6.11.2.1 char* CMML_Stream::id

id attribute of stream element

6.11.2.2 CMML_Time* CMML_Stream::basetime

basetime attribute of stream element

6.11.2.3 CMML_Time* CMML_Stream::utc

utc attribute of stream element

6.11.2.4 CMML_List* CMML_Stream::import

list of import elements

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

- **cmml.h**

6.12 CMML_Time Struct Reference

`#include <cmml.h>`

Data Fields

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

6.12.1 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.

6.12.2 Field Documentation

6.12.2.1 char* CMML_Time::tstr

copy of original time string for reuse in printing

6.12.2.2 CMML_Time_Type CMML_Time::type

the type of time for selecting the union content

6.12.2.3 CMML_UTC* CMML_Time::utc

actual time in utc

6.12.2.4 double CMML_Time::sec

actual time in seconds

6.12.2.5 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**

6.13 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**

6.13.1 Detailed Description

CMML_UTC: utc time specification struct similar to the one defined in time.h

6.13.2 Field Documentation

6.13.2.1 short CMML_UTC::tm_hsec

hundreds of seconds [0-99]

6.13.2.2 short CMML_UTC::tm_sec

seconds after the minute [0-59]

6.13.2.3 short CMML_UTC::tm_min

minutes after the hour [0-59]

6.13.2.4 short CMML_UTC::tm_hour

hours since midnight [0-23]

6.13.2.5 short CMML_UTC::tm_mday

day of the month [1-31]

6.13.2.6 short CMML_UTC::tm_mon

months since January [1-12]

6.13.2.7 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.

7.3 cmml-timeshift.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**
- double **secs**

7.3.1 Detailed Description

cmml-timeshift: parses a CMML instance document and applies a timeshift to all the clip tags.

```
Usage: cmml-timeshift [options] filename
Apply a timeshift to all the clip tags.

Possible options:
  -s offset, --sec offset
                Specify the offset in seconds to add to clip tags
  -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
```

7.3.2 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

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

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

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

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

main function of cmml-timeshift, 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.

7.3.4 Variable Documentation

7.3.4.1 FILE* outfile [static]

outfile: defines FILE pointer to print output to

7.3.4.2 double secs [static]

secs: number of seconds to offset the clip start/end times

7.4 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**

7.4.1 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`

7.4.2 Define Documentation

7.4.2.1 #define BUFSIZE 100000

the size of the print buffer

7.4.3 Function Documentation

7.4.3.1 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.

7.4.3.2 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.

7.4.3.3 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**.

7.4.3.4 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**.

7.4.3.5 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**.

7.4.4 Variable Documentation

7.4.4.1 int verbose [static]

verbose: turns on printing of parsed CMML file

7.5 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_LinkElement**
- 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, char *granulerate)
- **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_time_new_secs** (double seconds)
- **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)

7.5.1 Detailed Description

cmml.h(p. 52) includes definitions of the public API and data types required to parse CMML 2.0 files.

7.5.2 Typedef Documentation

7.5.2.1 typedef void CMML

A CMML handle.

7.5.2.2 typedef enum _CMML_Time_Type CMML_Time_Type

CMML_Time_Type: enumerates the different time types

7.5.2.3 typedef struct _CMML_List CMML_List

CMML_List: A doubly linked list

7.5.2.4 typedef void*(* CMML_CloneFunc)(void *data)

CMML_CloneFunc: Signature of a cloning function for CMML_List.

7.5.2.5 typedef void*(* CMML_FreeFunc)(void *data)

CMML_FreeFunc: Signature of a freeing function.

7.5.2.6 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

7.5.2.7 typedef enum _CMML_Element_Type CMML_Element_Type

CMML_Element_Type: indication of what the **CMML_Element**(p. 25) actually represents.

7.5.2.8 typedef enum _CMML_Error_Type CMML_Error_Type

CMML_Error_Type: indication of the type of error that occured. 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

7.5.2.9 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

7.5.2.10 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

7.5.2.11 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

7.5.3 Enumeration Type Documentation

7.5.3.1 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

7.5.3.2 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

7.5.3.3 enum _CMML_Error_Type

CMML_Error_Type: indication of the type of error that occured. 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

7.5.4 Function Documentation

7.5.4.1 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**.

7.5.4.2 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**.

7.5.4.3 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.

7.5.4.4 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**.

7.5.4.5 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**.

7.5.4.6 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**.

7.5.4.7 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.

7.5.4.8 CMML_Preamble* cmml_get_preamble (CMML * *cmml*)

cmml_get_preamble: returns a structure containing some of the parsing informtion received out of a stream.

Parameters:
cmml a CMML* handle

Returns:
1 if CMML tag specificed, 0 otherwise

Examples:
cmml-validate.c.

7.5.4.9 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

7.5.4.10 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

7.5.4.11 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

7.5.4.12 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

7.5.4.13 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.

7.5.4.14 void cmml_error_clear (CMML * *cmml*)

cmml_error_clear: Clear the last error.

Parameters:
cmml a CMML* handle

7.5.4.15 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

7.5.4.16 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**.

7.5.4.17 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**.

7.5.4.18 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**.

7.5.4.19 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.

7.5.4.20 CMML_Preamble* cmml_preamble_new (char * *encoding*, char * *id*, char * *lang*, char * *dir*, char * *granulerate*)

cmml_preamble_new: Create a new **CMML_Preamble**(p. 35).

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

granulerate the granulerate attribute of the cmml tag; not used if NULL

Returns:

a CMML preamble

Examples:

cmml-write.c.

7.5.4.21 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

7.5.4.22 CMML_Stream* cmml_stream_new (void)

cmml_stream_new: Create a new **CMML_Stream**(p. 37).

Returns:

a stream element

7.5.4.23 CMML_Head* cmml_head_new (void)

cmml_head_new: Create a new **CMML_Head**(p. 27).

Returns:
a head element

Examples:
cmml-write.c.

7.5.4.24 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.

7.5.4.25 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

7.5.4.26 CMML_Preamble* cmml_preamble_clone (CMML_Preamble * *src*)

cmml_preamble_clone: Create a copy of a **CMML_Preamble**(p. 35).

Parameters:
src the original preamble

Returns:
a copy of a CMML preamble

7.5.4.27 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

7.5.4.28 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

7.5.4.29 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

7.5.4.30 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

7.5.4.31 void cmml_preamble_destroy (CMML_Preamble * *preamble*)

cmml_preamble_destroy: Free all memory associated with a **CMML_Preamble**(p. 35).

Parameters:
preamble the preamble to free

Examples:
cmml-write.c.

7.5.4.32 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

7.5.4.33 void cmml_stream_destroy (CMML_Stream * *stream*)

cmml_stream_destroy: Free a stream element.

Parameters:
stream the stream element to free

7.5.4.34 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.

7.5.4.35 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.

7.5.4.36 void cmml_error_destroy (CMML_Error * *error*)

cmml_error_destroy: Free an error streucture.

Parameters:
error the error structure to free

7.5.4.37 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**.

7.5.4.38 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.

7.5.4.39 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.

7.5.4.40 int cmml_stream_pretty_snprint (char * *buf*, int *n*, CMML_Stream * *stream*)

cmml_stream_pretty_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.

Examples:

cmml-validate.c.

7.5.4.41 int cmml_head_snprint (char * *buf*, int *n*, CMML_Head * *head*)

cmml_head_snprint: 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.

7.5.4.42 int cmml_head_pretty_snprint (char * *buf*, int *n*, CMML_Head * *head*)

cmml_head_pretty_snprint: 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.

7.5.4.43 int cmml_clip_snprint (char * *buf*, int *n*, CMML_Clip * *clip*)

cmml_clip_snprint: 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.

7.5.4.44 int cmml_clip_pretty_snprint (char * *buf*, int *n*, CMML_Clip * *clip*)

cmml_clip_pretty_snprint: 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.

7.5.4.45 int cmml_error_snprint (char * *buf*, int *n*, CMML_Error * *error*, CMML * *cmml*)

cmml_error_snprint: 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.

7.5.4.46 CMML_List* cmml_list_new (void)

cmml_list_new: Create a new list.

Returns:
a new list

7.5.4.47 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

7.5.4.48 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

7.5.4.49 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

7.5.4.50 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

7.5.4.51 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

7.5.4.52 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)

7.5.4.53 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

7.5.4.54 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

7.5.4.55 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)

7.5.4.56 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

7.5.4.57 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

7.5.4.58 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

7.5.4.59 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

7.5.4.60 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.

7.5.4.61 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.

7.5.4.62 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.

7.5.4.63 CMML_Time* cmml_time_new_secs (double *seconds*)

cmml_time_new_sec: Creates a time struct from a npt sec specs.

Parameters:
seconds sec spec to be used for creating a new time struct

Returns:
a time struct.

7.5.4.64 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.

7.5.4.65 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.

7.5.4.66 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 basetime in seconds as given in the stream tag

Returns:

a time structure in seconds

7.5.4.67 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 basetime

Parameters:

t the given time struct

ref the reference utc time if given in the stream tag

base the basetime in seconds as given in the stream tag

Returns:

a time structure in seconds

7.5.4.68 void cmml_time_free (CMML_Time * t)

cmml_time_free: Free time struct

Parameters:
t the given time struct

7.5.4.69 CMML_Time* cmml_time_clone (CMML_Time * t)

cmml_time_clone: Copying times

Parameters:
t the given time struct

Returns:
the cloned time struct

7.5.4.70 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

7.5.4.71 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

7.5.4.72 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

7.5.4.73 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

7.5.4.74 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.

7.5.4.75 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.

7.5.4.76 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. 52).

The general sequence of API calls is the following:

- open a CMML object using **cmml_open()**(p. 58) on a local file or **cmml_new()**(p. 58) with a FILE pointer to an already opened file.
- attach read callbacks using **cmml_set_read_callbacks()**(p. 59)
- call **cmml_read()**(p. 59) repeatedly until it returns 0 or -1.
- close the CMML object with **cmml_destroy()**(p. 58) returning the FILE pointer or use **cmml_close()**(p. 59) 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. 62) 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 basetime attribute of the stream tag. libcmml provides an API for this functionality through the **cmml_skip_to_secs()**(p. 62) 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 basetime attributes of the stream tag. libcmml provides an API for this functionality through the **cmml_skip_to_utc()**(p. 62) 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>

#ifndef 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 (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. 35) with **cmml_preamble_new()**(p. 63) and print it with **cmml_preamble_snprint()**(p. 67)
- create a **CMML_Head**(p. 27) with **cmml_head_new()**(p. 64), change the attribute and element contents in the **CMML_Head**(p. 27) structure, and print it with **cmml_head_snprint()**(p. 68) or **cmml_head_pretty_snprint()**(p. 68)
- create one or more **CMML_Clip**(p. 21) with **cmml_clip_new()**(p. 64), change the attribute and element contents in the **CMML_Clip**(p. 21) structures, and print them with **cmml_clip_snprint()**(p. 69) or **cmml_clip_pretty_snprint()**(p. 69)
- 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, 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 % Playbacktime "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 ANNODEX 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
  granulerate 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
  %i18n;
  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 ANNODEX DOCUMENT] -->
<!-- end      = specifies the end time of the clip; specified in
                time relative to the timebase of the header
                [NOT INCLUDED IN ANNODEX DOCUMENT] -->
<!ELEMENT clip (meta*, a?, img?, desc?)>
<!ATTLIST clip
  %i18n;
  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
  %i18n;
  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