Bento4 SDK documentation

Axiomatic Systems, LLC

2012

Table of Contents

1 WHY USE BENTO4?

1.1 Overview (README.txt)

Bento4/AP4 is a C++ class library designed to read and write ISO-MP4 files. This format is defined in [ISO/IEC 14496]. The format is a derivative of the Apple Quicktime file format. Because of that, Bento4 can be used to read and write a number of Quicktime files as well, even though some Quicktime specific features are not supported. In addition, Bento4 supports a number of extensions as defined in various other specifications. This includes some support for ISMA Encrytion and Decryption as defined in the [ISMA E&A] specification, OMA 2.0 DCF/PDCF Encryption and Decryption as defined in the [OMA 2.0 DCF/PDCF] specification, and iTunes compatible metadata. The SDK includes a number of command line tools, built using the class library, that serve as general purpose tools as well as examples of how to use the API.

The SDK is designed to be cross-platform. The code is very portable; it can be compiled with any sufficiently modern C++ compiler. The code does not rely on any external library; all the code necessary to compile the SDK and its tools is included in the standard distribution. The standard distribution contains makefiles for unix-like operating systems, including Linux, project files for Microsoft Visual Studio, and an XCode project for MacOS X. There is also support for building the library with the SCons build system.

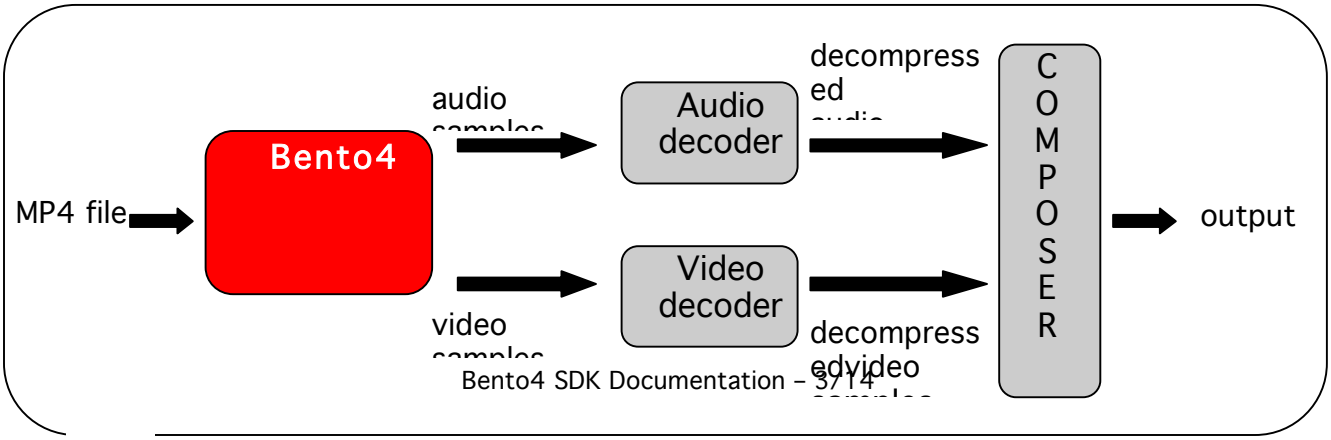
You can find out more about the license in LICENSE.txt (see Section 8). You can also read some of the API documentation produced from the source code comments using Doxygen. The doxygen output is available as a windows CHM file in Bento4.chm, and a set of HTML pages zipped together in Bento4-HTML.zip (to start, open the file named index.html with an HTML browser).

1.2 Typical Bento4 use cases

Here are some typical use cases for the Bento4 library.

1.2.1 Reading an MP4 file in a player

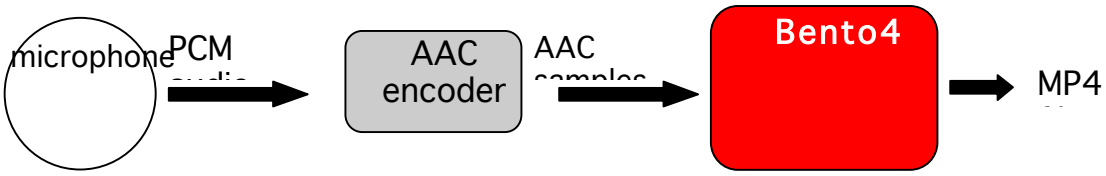
Bento4 is able to take an MP4 file as an input and extract audio and video samples for the relevant decoder to process as well as the information needed to initialize the



decoders.

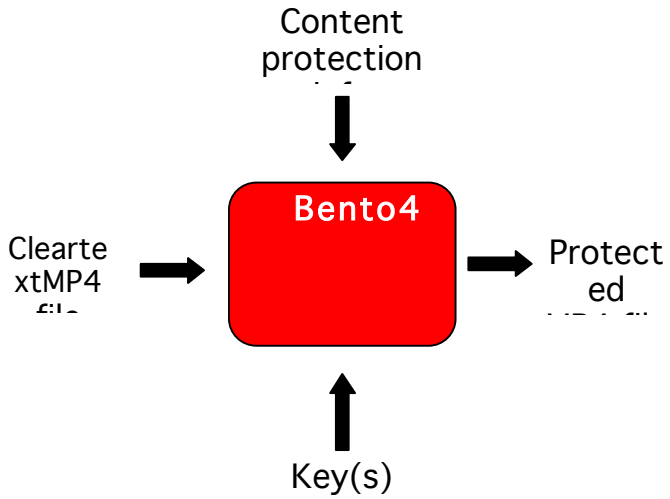
1.2.2 Writing an MP4 file from a source

Bento4 is able to package an MP4 file from an encoder output. Below is an example of a complete setup.



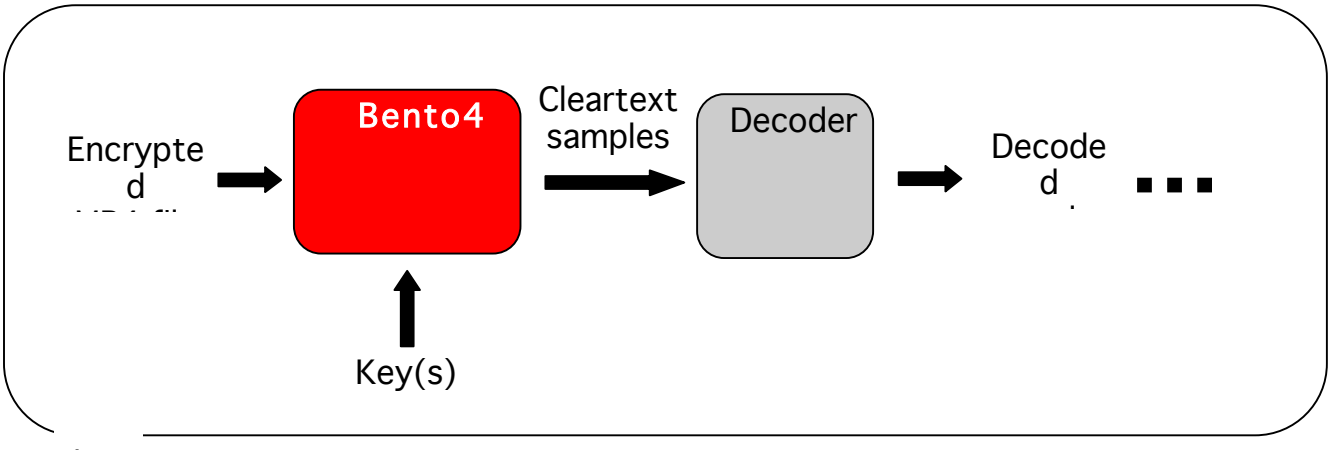
1.2.3 Encrypting a file and adding content protection information

Bento4 is able to encrypt a cleartext MP4 file given a set of keys, as well as to package additional information such as content protection data.



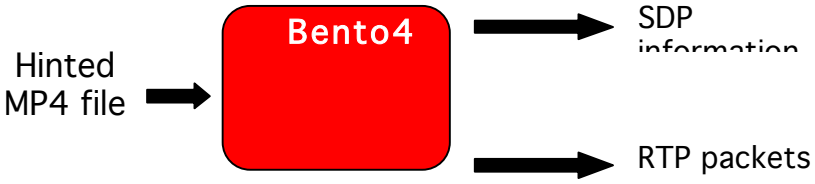
1.2.4 Decrypting a file sample per sample in a player

Given the set of relevant keys, Bento4 can decrypt the content of an encrypted MP4 file sample by sample for the decoder to process.



1.2.5 Reading hint tracks of an MP4 file in an RTSP/RTP server

From a hinted MP4 file, Bento4 is able to extract SDP information to be sent for the RTSP DESCRIBE response and RTP packets containing the media data (see [ISMA]).



1.3 Downloading Bento4

Bento4 and all relevant documentation are available for download from the following location: http://www.bento4.com

1.4 Dependencies and requirements

Bento4 is a standalone multi-platform project. It bears no source-level dependencies to any other software package. See Section 3 for building instructions.

2 Overview of source tree

This section provides an overview of the main directories in the source tree.

2.1 High-level view of the source tree

The standard Bento4 distribution source tree unfolds as such:

- Build
  - Makefiles
  - Targets
  - Tools
- Documents
- Source
  - C++
    - Apps
    - Codecs
    - Core
    - Crypto
    - MetaData
    - System
    - Test
  - Java

2.2 Build directory

The Build directory contains all necessary files for a successful build of the contents of the Source directory. For instructions, see Section 3 below.

2.3 Documents directory

All relevant documentation is located in the Documents directory.

2.4 Source directory

As reflected in the Source directory, Bento4 comes in two flavors: C++ and Java. The C++ implementation is to be found in the C++ directory. A very limited port in Java is located in the Java directory. This port will be documented in future versions of this document.

2.4.1 Library sources

The System directory contains system implementation specific source files.

The Core directory contains the files necessary to build the Bento4 core library. Any application which uses Bento4 should link against the library built from these source files as well as the system library, and may also link against other libraries provided in Bento4 (see below).

The Codecs directory contains codec-specific sources that are used to create the codec library. This library currently contains only AAC ADTS and AVC/H.264 utils but may be augmented in the future.

The Crypto directory contains the files used by Bento4 for low-level encryption and decryption of the content. They are used to create the crypto library.

The files relative to iTunes metadata are in the MetaData directory. They are used to

create the metadata library.

2.4.2 Application sources

The Test directory contains application sources that test some parts of the overall project.

The Apps directory contains the source files to all the applications available with Bento4. See Section 7 for detailed example applications.

3 BUILDING BENTO4

This section reviews the tools available for building Bento4. Detailed instructions relative to the build system used are given, as well as platform-specific information.

3.1 Overview and build tools

3.1.1 Organization

The build files are located in the Build directory. The Targets directory contains the build files for the targets supported by Bento4. The target names follow the convention:

```
<architecture>-<vendor>-<os>[-<build-tool>]
```

3.1.2 Microsoft Visual Studio

For Windows, Microsoft Visual Studio 2005 professional is used (MSVS 2005).

3.1.3 SCons

Scons is a build tool written in Python.

Installing SCons

- Make sure that you have Python 2.4 or higher installed on your system.
- Download SCons 0.98 or higher (the latest version at the time of writing is 1.2.0) from

http://sourceforge.net/project/downloading.php?groupname=scons

- Open a terminal and type the following instructions (where the `$` sign symbolizes the beginning of a line in the terminal window):

```
$ tar xvfz scons-1.2.0.tar.gz
$ cd scons-1.2.0
$ sudo python setup.py install
```

Using SCons

- Go to your target directory.
- For Debug, type:

```
$ scons -u
```

- For Release, type:

```
$ scons -u build_config=Release
```

3.1.4 make

- Go to your target directory.
- For Debug, type:
  ```
  $ make
  ```
- For Release, type:
  ```
  $ make AP4_BUILD_CONFIG=Release
  ```

3.1.5 XCode (Mac OS X only)

- Download XCode from http://developer.apple.com/tools/.
- Install it.

3.2 Platform specific instructions

3.2.1 Windows desktop (x86-microsoft-win32-vs2005 and x86-microsoft-win32-vs2010)

- Requirements: Microsoft Visual Studio 2005 or Visual Studio 2010.
- Open the Bento4.sln solution file with MSVS 2005 or MSVS 2010.
- Build.

3.2.2 Cygwin (x86-unknown-cygwin)

- Requirements:
  - g++
  - make (from cygwin)
- Build with make according to the instructions in Section 3.1.

3.2.3 Linux x86 (x86-unknown-linux)

- Requirements:
  - g++
  - make or SCons
- Build with make or SCons according to the instructions in Section 3.1.

3.2.4 Apple Mac OS X

- Requirements: developer tools should be installed.
- Build with SCons or XCode according to the instructions in section 3.1.

4 The Bento4 API

Bento4 uses a simple C++ API detailed in [Bento4Doxy].

5 READING A FILE: A SIMPLE WALK-THROUGH

The class AP4_File represents all the information about an MP4 file. Internally, a tree of AP4_Atom objects plus other helper objects holds the actual information.

To create an instance of the class, the caller must pass a reference to an AP4_ByteStream object that represents the file data storage. The SDK includes two subclasses of the abstract AP4_ByteStream class: AP4_FileByteStream for reading/writing disk-based files and AP4_MemoryByteStream for working with in-memory file images.

Once you have created an AP4_File object, you can get to the media data by accessing the AP4_Track objects of its AP4_Movie (see AP4_File::GetMovie). The AP4_Track class exposes the necessary methods for you to access the AP4_SampleDescription (typically to initialize your decoder) and to get the AP4_Sample objects from the track. These AP4_Sample objects give you the meta information you need (such as timestamps) as well as the sample data they point to.

You can also explore the entire tree of atoms by calling AP4_File::Inspect, passing an instance of AP4_AtomInspector. AP4_AtomInspector is an abstract base class for a visitor of the tree of AP4_Atom objects. The SDK includes a concrete subclass, AP4_PrintInspector, which can be used to print out a text representation of the atoms as they are visited. See the Mp4Dump command line application for an example.

6 WRITING A FILE: A SIMPLE WALK-THROUGH

To create a new MP4 file, you should first create an AP4_SyntheticSampleTable sample table for each track in your file. Specify the sample description for the media samples in each track by calling AP4_SyntheticSampleTable::AddSampleDescription with the description for the samples of that track.

Samples can then be added to the sample table with the AP4_SyntheticSampleTable::AddSample method. Once all the samples of all the tracks have been added to the sample tables, you can create an AP4_Movie, and an AP4_Track from each sample table, add the track to the movie, and finally create an AP4_File from the movie object.

Finally, the AP4_File object can be serialized to a byte stream (such as an AP4_FileByteStream) using the AP4_FileWriter class. See the Aac2Mp4 application or the AvcTrackWriterTest test as examples.

7 EXAMPLE APPLICATIONS

The following are example applications provided with Bento4.

7.1 Aac2Mp4

Aac2Mp4 converts an AAC ADTS file into an MP4 file. It is used as follows:

```
usage: aac2mp4 <input> <output>
```

where <input> is your AAC ADTS file and <output> your MP4 file.

Example:

```
aac2mp4 my_aac_file.aac my_mp4_file.mp4
```

7.2 Mp42Aac

Mp42Aac converts an MP4 file into an AAC ADTS file. It is used as follows:

```
usage: mp42aac [options] <input> <output>
Options:
--key <hex>: 128-bit decryption key (in hex: 32 chars)
```

where <input> is your MP4 file and <output> your AAC ADTS file. In case the MP4 file is encrypted, use the --key option (a 128-bit decryption key – 32 hexadecimal characters) to decrypt it before the conversion.

Example:

```
mp42aac --key 000102030405060708090a0b0c0d0e0f my_mp4_file.mp4 my_aac_file.aac
```

7.3 Mp4Edit

Mp4Edit allows you to edit an MP4 file by inserting, removing or replacing one or more atoms in a file. It is used as follows:

```
usage: mp4edit [commands] <input> <output>
where commands include one or more of:
     --insert <atom_path>:<atom_source_file>[:<position>]
     --remove <atom_path>
     --replace <atom_path>:<atom_source_file>
```

where <input> is the MP4 file you would like to edit, and <output> the edited MP4 file. The optional commands (insert, remove or replace) are called with the following parameters: *atom_path* identifies the location of the atom to edit, and the *atom_source_file* refers to the file containing the data you would like to insert or replace current data with. The optional parameter <position> indicates where the new atom should be inserted in the parent atom; by default, the new data will be appended to the end of parent atom. All three commands can be run at the same time.

Example:

```
mp4edit --insert moov/trak[0]/mdia/stbl:my_atom_file.atom:2 --remove
moov/trak[4]/mdia/stbl my_input_file.mp4 my_output_file.mp4
```

7.4 Mp4Encrypt

Given a set of keys, Mp4Encrypt will repackage a cleartext MP4 file into an encrypted MP4 file. It is used as follows:

```
usage: mp4encrypt --method <method> [options] <input> <output>
--method: <method> is OMA-PDCF-CBC, OMA-PDCF-CTR or ISMA-IAEC
Options:
--show-progress: show progress details
--key <n>:<k>:<iv>
     Specifies the key to use for a track.
     <n> is a track ID, <k> a 128-bit key in hex (32 characters) and <iv> a
     64-bit IV or salting key in hex (16 characters)
     (several --key options can be used, one for each track)
--property <n>:<name>:<value>
     Specifies a named string property for a track
     <n> is a track ID, <name> a property name, and <value> is the property
     value
     (several --property options can be used, one or more for each track)
--kms-uri <uri>
     Specifies the KMS URI for the ISMA-IAEC method
```

where <input> is your cleartext MP4 file, and <output> the corresponding encrypted MP4 file. The method used for encryption should be chosen among OMA-PDCF-CBC, OMA-PDCF-CTR and ISMA-IAEC. Optionally, the --show-progress parameter will give you a verbose update on the encryption process. The key to be used for encryption is specified using a track identifier, a 128-bit key in hex (32 characters) and <iv> a 64-bit IV or salting key in hex (16 characters). The kms-uri parameter is used only for the ISMA-IAEC method. The --property option is only available for the OMA-PDCF-CBC and OMA-PDCF-CTR methods. Currently, it can only be set to ContentId, whose value is equal to ContentId. If an unsupported property is specified, it will be ignored.

Example:

```
mp4encrypt --method OMA-PDCF-CTR --key
0:000102030405060708090a0b0c0d0e0f:0001020304050607 --property
0:ContentId:ContentId01
```

7.5 Mp4Extract

Mp4Extract extracts an atom from an input MP4 file and copies it in an output MP4 file.

```
usage: mp4extract [options] <atom_path> <input> <output>
options:
--payload-only : omit the atom header
```

Example:

```
mp4extract --payload-only moov/trak[0]/mdia/stbl my_input_file.mp4
my_output_file.mp4
```

7.6 Mp4Decrypt

Provided the relevant set of keys are produced, Mp4Decrypt repackages an

encrypted MP4 file into a cleartext MP4 file.

```
usage: mp4decrypt [--key <n>:<k>] <input> <output>
--show-progress: show progress details
--key: <n> is a track index, <k> a 128-bit key in hex
     (several --key options can be used, one for each track)
```

where <input> is your encrypted MP4 file, and <output> is the corresponding cleartext MP4 file. Optionally, the --show-progress parameter will give you a verbose update on the decryption process. The key parameter is specified using a track index and a 128-bit key in hexadecimal characters. Several key options can be used, given that each track may have its own key.

Example:

```
mp4decrypt --key 0:000102030405060708090a0b0c0d0e0f --show-progress
my_encrypted_file.mp4  my_decrypted_file.mp4
```

7.7 Mp4Dump

Mp4Dump is a command line application that outputs on your terminal a tree view of all the atoms contained in an MP4 file. In case the file is encrypted, it can decrypted given the relevant set of keys.

```
usage: mp4dump [options] <input>
options are:
--track <track_id>[:<key>]
     writes the track data into a file (<mp4filename>.<track_id>)
     and optionally tries to decrypt it with the key (128-bit in hex)
     (several --track options can be used, one for each track)
```

Example:

```
mp4dump --track 0:000102030405060708090a0b0c0d0e0f my_input_file.mp4
```

7.8 Mp4Info

Mp4Info allows the user to access high-level information about the input (duration, number of tracks, codec type, etc.).

```
usage: mp4info [options] <input>
Options:
--verbose: show sample details
```

where <input> is the MP4 file containing the data on which you are requesting information. The verbose option lets you see the detailed results of your query.

Example:

```
mp4info --verbose my_input_file.mp4
```

7.9 Mp4RtpHintInfo

Mp4RtpHintInfo outputs the information needed for an RTSP/RTP session from a hinted movie.

```
usage: mp4rtphintinfo [--trackid <hinttrackid>] <input>
```

where <input> is the MP4 file on which you are requesting information. The --trackid option lets you specify the hint track to be processed. If the --trackid option is specified, the program will verify that the track pointed is indeed a hint track and if so will:

- display the SDP information for this track;
- dump the RTP Packets generated from this hint track in the file <input>-track-<hinttrackid>.rtp.

Example:

```
mp4rtphintinfo --trackid 2 my_hinted_file.mp4
```

where 2 is the ID of the hint track.

7.10 Mp4Tag

Mp4tag gives access to iTunes metadata. This application is still in development. More documentation will be added later.

```
usage: mp4tag [options] [commands...] <input> [<output>]
commands:
--help
     print this usage information
--show-tags
     show tags found in the input file
--list-symbols
     list all the builtin symbols
--list-keys
     list all the builtin keys
--set <key>:<encoding>:<value>
     set a tag (if the tag does not already exist, set behaves like add)
--add <key>:<encoding>:<value>
     set/add a tag
     where <encoding> is:
               s if <value> is a direct string
               i if <value> is a decimal integer
               f if <value> is a filename
               z if <value> is a the name of a builtin symbol
--remove <key>
     remove a tag
--extract <key>:<file>
     extract the value of a tag and save it to a file

In all commands with a <key> argument, except for 'add', <key> can be a key
name or name#n where n is the index of the key when there is more than one key
with the same name (ex: multiple images for cover art). The name of a key is
either one of the builtin keys (see --list-keys) or namespace/key, where
namespace is either 'meta' for the default metadata namespace or a user
defined namespace. For the 'meta' namespace, the key name must be a 4
character atom name.
```

8 Copyright and Licensing

8.1 Copyright

8.2 LICENSE.txt

Bento4 is available under two different licenses.

For applications that are entirely distributable under the terms of the GPL, the Bento4 GPL license applies.

For applications that cannot be entirely distributable under the terms of the GPL (either the application, or code modules linked with the application are not compatible with the terms of the GPL licence), a non-GPL commercial license is available from Axiomatic Systems LLC. Contact Gilles Boccon-Gibod (licensing@axiosys.com or bok@bok.net) for more information.

9 References

[Bento4Doxy]	Bento4 SDK Doxygen documentation (http://bento4.sourceforge.net/docs/html/index.html)
[ISMA E&A]	ISMA E&A specification (http://www.isma.tv)
[ISO/IEC 14496]	ISO/IEC 14496-12, 14496-14 and 14496-15 MP4 specification
[OMA 2.0 DCF/PDCF]	OMA 2.0 DCF/PDCF specification (http://www.openmobilealliance.org)