



MP4 Core Parser Library API Specification

Version: 1.0

Date: Mar. 2, 2010

Freescale Semiconductor, Shanghai Software Development Center
UnitA, 20F, 560#, SongTao Road, Pu Dong New District
Shanghai, 201203, P.R of China

Revision History

Version	Date	Revised By	Description of Changes
1.0	03/02/2010	Amanda Lin	Initial version. Based on MP4 core parser library version 6.1

1. Introduction.....	4
1.1. In this document.....	4
1.2. References.....	4
1.3. Symbols and abbreviated terms	4
2. API Functions	4
2.1. Creation and Deletion	4
2.1.1. MP4ParserVersionInfo	4
2.1.2. MP4CreateParser	5
2.1.3. MP4DeleteParser	5
2.2. Movie & Track Properties	5
2.2.1. MP4GetNumTracks	5
2.2.2. MP4GetMovieDuration	6
2.2.3. MP4GetTrackDuration	6
2.2.4. MP4GetTrackType	6
2.2.5. MP4GetDecoderSpecificInfo.....	7
2.2.6. MP4GetMaxSampleSize.....	7
2.2.7. MP4GetBitRate.....	8
2.2.8. MP4GetSampleDuration.....	8
2.2.9. MP4GetLanguage	9
2.2.10. MP4GetVideoFrameWidth	9
2.2.11. MP4GetVideoFrameHeight	10
2.2.12. MP4GetAudioNumChannels	10
2.2.13. MP4GetAudioSampleRate.....	10
2.2.14. MP4GetAudioBitsPerSample	11
2.2.15. MP4GetTextTrackWidth	11
2.2.16. MP4GetTextTrackHeight	12
2.3. User Data	12
2.3.1. MP4GetUserData.....	12
2.4. Normal Playback.....	13
2.4.1. MP4GetNextSample	13
2.5. Seek and Trick Mode	14
2.5.1. MP4IsSeekable	14
2.5.2. MP4Seek	15
2.5.3. MP4GetSyncSample	16
3. Data Types & Constants	17
3.1. Error Codes	17
3.2. Handle of MP4 Parser.....	18
3.3. Other Constants.....	18
3.3.1. Maximum Track Number	18
3.3.2. Media & Codec Types	18
3.3.3. User data ID	19
4. API Calling Sequence	19

1. Introduction

1.1. In this document

Freescale MP4 core parser library provides parsing and data reading functions on local or streaming MP4 files. Both ISO/IEC standard MP4 file format & QuickTime file format are supported.

This document is the API specification for the MP4 core parser library. The calling sequence of the API functions is also explained.

The API is declared in the header file "MP4ParserAPI.h".

1.2. References

- [1] The following parts of ISO/IEC 14496, under the general title Information technology — Coding of audio-visual objects:
- Part 1: Systems
 - Part 12: ISO base media file format
 - Part 14: MP4 file format
 - Part 15: Advanced Video Coding (AVC) file format

- [2] QuickTime File Format Specification (2007) by Apple Inc.

1.3. Symbols and abbreviated terms

AVC ISO/IEC 14496-10, Advanced Video Coding. Same as ITU-T H.264.
NAL Network Abstraction Layer

2. API Functions

2.1. Creation and Deletion

2.1.1. MP4ParserVersionInfo

```
const char * MP4ParserVersionInfo()
```

Description:

Function to get the MP4 core parser version.

Return value:

ASCII Version string like "MP4_PARSER_xx.xx.xx".

2.1.2. MP4CreateParser

```
int32 MP4CreateParser(file_stream_t * stream,
                     MP4MemoryOps * memOps,
                     void * context,
                     MP4ParserHandle * parserHandle);
```

Description:

Function to create the MP4 core parser. This is the 1st necessary API to call.

Arguments:

stream [in] Source stream of the MP4 file.
It implements functions to open, close, tell, read and seek a file.

memOps [in] Memory operation callback table.
It implements the functions to malloc, calloc, realloc and free memory.

context [in] Wrapper context for the callback functions. It will be used as an arguments of the
callback functions. MP4 core parser will never modify its content.

parserHandle [out] Handle of the MP4 core parser if the core parser is created successfully.
NULL for failure.

Return value:

PARSER_SUCCESS - Success
Other error codes - Failure

2.1.3. MP4DeleteParser

```
int32 MP4DeleteParser(MP4ParserHandle parserHandle)
```

Description:

function to delete the MP4 core parser. This is the last API to call.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

Return value:

PARSER_SUCCESS - Success
Other error codes - Failure

2.2. Movie & Track Properties

2.2.1. MP4GetNumTracks

```
int32 MP4GetNumTracks (MP4ParserHandle parserHandle,
                      bool * seekable)
```

Description:

Function to tell how many tracks in the movie.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

numTracks [out] Number of tracks.

Return value:

PARSER_SUCCESS - Success
Other error codes - Failure

2.2.2. MP4GetMovieDuration

```
int32 MP4GetTheMovieDuration(MP4ParserHandle parserHandle,  
                             uint64 * usDuration)
```

Description:

Function to tell the movie duration.
The tracks may have different durations. And the movie's duration is usually the video track's duration (maybe not the longest one).

Arguments:

parserHandle [in] Handle of the MP4 core parser.
usDuration [out] Duration in us.

Return value:

PARSER_SUCCESS - Success
Other error codes – Failure

2.2.3. MP4GetTrackDuration

```
int32 MP4GetTheTrackDuration(MP4ParserHandle parserHandle,  
                             uint32 trackNum,  
                             uint64 * usDuration)
```

Description:

Function to tell a track's duration.
The tracks may have different durations. And the movie's duration is usually the video track's duration (maybe not the longest one).

If a track's duration is 0, this track is empty! Application can read nothing from an empty track.

Arguments:

parserHandle [in] Handle of the MP4 core parser.
trackNum [in] Number of the track, 0-based.
usDuration [out] Duration in us.

Return value:

PARSER_SUCCESS - Success
Other error codes – Failure

2.2.4. MP4GetTrackType

```
int32 MP4GetTrackType(MP4ParserHandle parserHandle,  
                      uint32 trackNum,  
                      uint32 * mediaType,  
                      uint32 * decoderType,  
                      uint32 * decoderSubtype);
```

Description:

Function to tell the type of a track, includes the media type, decoder type and decoder subtype if available.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based.

mediaType [out] Media type of the track. (video, audio, subtitle...)
 "MEDIA_TYPE_UNKNOWN" means the media type is unknown.

decoderType [out] Decoder type of the track if available. (eg. MPEG-4, H264, AAC, MP3, AMR ...)
 "UNKNOWN_CODEC_TYPE" means the decoder type is unknown.

decoderSubtype [out] Decoder Subtype type of the track if available. (eg. AMR-NB, AMR-WB ...)
 "UNKNOWN_CODEC_SUBTYPE" means the decoder subtype is unknown.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.2.5. MP4GetDecoderSpecificInfo

```
int32 MP4GetDecoderSpecificInfo (MP4ParserHandle parserHandle,
                                uint32 trackNum,
                                uint8 ** data,
                                uint32 * size)
```

Description:

Function to tell the decoder specific information of a track.

Now only MPEG-4/H.264 video and AAC audio tracks have decoder specific information.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based.

data [out] Buffer holding the codec specific information. The user shall never free this buffer.

size [out] Size of the codec specific information, in bytes.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.2.6. MP4GetMaxSampleSize

```
int32 MP4GetMaxSampleSize( MP4ParserHandle parserHandle,
                           uint32 trackNum,
                           uint32 * size)
```

Description:

Optional API. Function to tell the maximum sample size of a track.

MP4 parser read A/V tracks sample by sample. The max sample size can help the user to prepare a big enough buffer.

Warning:

[1]The "max sample size" is not available if the index table is invalid.

And for MP4 file, if the index table is invalid, playback will be affected.

[2]Core parser need to scan the whole index table to get the max sample size. So this API is time-consuming for long movies that has big index tables. For application that requests a quick movie loading, it's better not to call this API. Anyway, the parser can output a sample piece by piece if output buffer is not big enough.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based.

size [out] Max sample size of the track. Warning! It can be zero if index table does not exist.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.2.7. MP4GetBitRate

```
int32 MP4GetBitRate( MP4ParserHandle parserHandle,
                    uint32 trackNum,
                    uint32 *bitrate)
```

Description:

Function to tell the average bitrate of a track.

If the average bitrate is not available in the file header, 0 is given.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based.

bitrate [out] Average bitrate, in bits per second.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.2.8. MP4GetSampleDuration

```
int32 MP4GetSampleDuration( MP4ParserHandle parserHandle,
                          uint32 trackNum,
                          uint64 *usDuration)
```

Description:

Function to tell the sample duration in us of a track.

If the sample duration is not a constant (eg. some audio, subtitle), 0 is given.

For a video track, the frame rate can be calculated from this information.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based.

usDuration [out] Sample duration in us. If sample duration is not a constant, this value is 0.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.2.9. MP4GetLanguage

```
int32 MP4GetLanguage(MP4ParserHandle parserHandle,
                     uint32 trackNum,
                     uint8 *threeCharCode)
```

Description:

Function to tell the language of a track used.

This is helpful to select a video/audio/subtitle track or menu pages.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based.

threeCharCode [out] Three or two character language code.

See ISO 639-2/T for the set of three character codes. Eg. 'eng' for English.

Four special case:

mis- "uncoded languages"

mul- "multiple languages"

und- "undetermined language"

zxx- "no linguistic content"

See ISO 639 for the set of two character codes. Eg. 'en' for English and 'zh' for Chinese.

If ISO 639 is used, the 3rd character will be '\0'.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.2.10. MP4GetVideoFrameWidth

```
int32 MP4GetVideoFrameWidth (MP4ParserHandle parserHandle,
                             uint32 trackNum,
                             uint32 *width)
```

Description:

Function to tell the width in pixels of a video track.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based. It must be a video track.

width [out] Width in pixels.

Return value:

PARSER_SUCCESS - Success
Other error codes – Failure

2.2.11. MP4GetVideoFrameHeight

int32 MP4GetVideoFrameHeight (MP4ParserHandle parserHandle,
uint32 trackNum,
uint32 *height)

Description:

Function to tell the height in pixels of a video track.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based. It must be a video track.

height [out] Height in pixels.

Return value:

PARSER_SUCCESS - Success
Other error codes – Failure

2.2.12. MP4GetAudioNumChannels

int32 MP4GetAudioNumChannels(MP4ParserHandle parserHandle,
uint32 trackNum,
uint32 *numchannels)

Description:

Function to tell how many channels in an audio track.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based. It must be an audio track.

numchannels [out] Number of the channels. 1 mono, 2 stereo, or more for multiple channels.

Return value:

PARSER_SUCCESS - Success
Other error codes - Failure

2.2.13. MP4GetAudioSampleRate

int32 MP4GetAudioSampleRate(MP4ParserHandle parserHandle,
uint32 trackNum,
uint32 *sampleRate)

Description:

Function to tell the audio sample rate (sampling frequency) of an audio track.

Warning:

The audio sample rate from the file header may be wrong. And if the audio decoder specific information is available (for AAC), the decoder can double check the audio sample rate from this information.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based. It must be an audio track.

sampleRate [out] Audio integer sample rate (sampling frequency).

Return value:

PARSER_SUCCESS - Success

Other error codes - Failure

2.2.14. MP4GetAudioBitsPerSample

```
int32 MP4GetAudioBitsPerSample (    MP4ParserHandle parserHandle,
                                     uint32 trackNum,
                                     uint32 * bitsPerSample)
```

Description:

Function to tell the bits per sample for an audio track.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based. It must be an audio track.

bitsPerSample [out] Bits per PCM sample.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.2.15. MP4GetTextTrackWidth

```
int32 MP4GetTextTrackWidth(    MP4ParserHandle parserHandle,
                                uint32 trackNum,
                                uint32 * width)
```

Description:

Function to tell the width of a text (subtitle) track.

The text track defines a window to display the subtitles.

This window shall be positioned in the middle of the screen.

And the sample is displayed in the window. How to position the sample within the window is defined by the sample data.

The origin of window is always (0, 0).

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based. It must be a text track.

width [out] Width of the text track, in pixels.

Return value:

PARSER_SUCCESS - Success
Other error codes – Failure

2.2.16. MP4GetTextTrackHeight

```
int32 MP4GetTextTrackHeight ( MP4ParserHandle parserHandle,  
                             uint32 trackNum,  
                             uint32 * height)
```

Description:

Function to tell the height of a text (subtitle) track.
The text track defines a window to display the subtitles.
This window shall be positioned in the middle of the screen.
And the sample is displayed in the window. How to position the sample within the window is defined by the sample data.
The origin of window is always (0, 0).

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based. It must be a text track.

height [out] Height of the text track, in pixels.

Return value:

PARSER_SUCCESS - Success
Other error codes – Failure

2.3. User Data

User data information (such as title, artist etc.) is descriptive and optional information for movies.
They are not necessary for playback.

2.3.1. MP4GetUserData

```
int32 MP4GetUserData( MP4ParserHandle parserHandle,  
                     uint32 userDataId,  
                     uint8 ** buffer,  
                     uint32 *size)
```

Description:

Function to tell the user data information (title, artist, genre etc) of the movie.
The information is usually a null-terminated ASCII string.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track, 0-based.

userDataId [in] User data ID. Type of the user data.

e.g. USER_DATA_TITLE, USER_DATA_GENRE, USER_DATA_ARTIST etc.

buffer [out] Buffer containing the information.
The core parser manages this buffer and the user shall NOT free it.
If no such info is available, this value will be set to NULL.

size [out] Length of the information in bytes.
If no such info is available, this value will be set to 0.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.4. Normal Playback

To support normal playback (rate = 1X), MP4 parser provides one API to read samples from a track sequentially. The sample reading is track-based. Given the track number, the user can choose to read any track in the MP4 movie.

MP4 parser supports partial output of large samples. If the data buffer is not large enough, it can output the sample data piece by piece.

2.4.1. MP4GetNextSample

```
int32 MP4GetNextSample(    MP4ParserHandle parserHandle,
                           uint32 trackNum,
                           uint8 * sampleData,
                           uint32 * dataSize,
                           uint64 * usStartTime,
                           uint64 * usEndTime,
                           uint32 * flag)
```

Description:

Function to read the next sample from a track.

The data reading is track-based. Given the track number, the parser can output any track's samples one by one. It makes the switching easy among multiple audio/subtitles.

It supports partial output of large samples. If the entire sample can not be output by calling this function once, its remaining data can be got by repetitive calling the same function.

BSAC audio track is somewhat special:

Parser does not only read this BSAC track. But it will read one sample from each BSAC track in the proper order and make up a "big sample" for the user. Partial output is still supported and the sample flags describe this "big sample". So the user shall take all BSAC tracks as one track and use any BSAC track number to call this function.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track to read, 0-based.

sampleData [in] Buffer to hold the sample data.
If the buffer is not big enough, only part of sample is output.

dataSize [in/out] Size of the buffer as input, in bytes.

As output:

If an entire sample or part of a sample is output successfully (return value is `PARSER_SUCCESS`), it's the size of the data actually got.

If the sample can not be output at all because buffer is too small (the return value is `PARSER_INSUFFICIENT_MEMORY`), it's the buffer size needed. This is for efficiency consideration of tracks with very small size samples.

usStartTime [out] Start time of the sample in us (timestamp)

usEndTime [out] End time of the sample in us.

flag [out] Flags of this sample, if a sample or part of it is got successfully.

`FLAG_SYNC_SAMPLE`

Whether this sample is a sync sample (video key frame, random access point).

For non-video media, the application can take every sample as sync sample.

`FLAG_SAMPLE_NOT_FINISHED`

Sample data output is not finished because the buffer is not big enough. Need to get the remaining data by repetitive calling this function.

Return value:

`PARSER_SUCCESS` - An entire sample or part of it is got successfully.

`PARSER_EOS` - No sample is got because of end of the track.

`PARSER_INSUFFICIENT_MEMORY` - Buffer is too small to hold the sample
The user can allocate a larger buffer and call this API again.

`PARSER_READ_ERROR` - File reading error. No need for further error concealment.

Others - Other failures.

2.5. Seek and Trick Mode

An MP4 movie is seekable if and only if it has the index table and there are random access points (sync samples) in the index table. And seeking and trick mode (FF/RW) can be performed only on a seekable MP4 movie.

Seeking is also track-based like sample reading.

2.5.1. MP4IsSeekable

```
int32 MP4IsSeekable( MP4ParserHandle parserHandle,
                    bool * seekable)
```

Description:

Function to tell whether the movie is seekable. A seekable MP4 movie must have the index table.

If the file's index table is loaded from file successfully and there are random access points in the index table, the movie is seekable.

Seeking and trick mode can be performed on a seekable file.

If the index table is corrupted, the file is NOT seekable. This function will fail and return value can tell the error type.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

seekable [out] TRUE for seekable movie and FALSE for non-seekable one.

Return value:

PARSER_SUCCESS - Success

Other error codes - Failure

2.5.2. MP4Seek

```
int32 MP4Seek(MP4ParserHandle parserHandle,
              uint32 trackNum,
              uint64 * usTime,
              uint32 flag)
```

Description:

Function to seek a track to a target time. It will seek to a sync sample of the time stamp matching the target time. Due to the scarcity of the video sync samples (key frames), there can be a gap between the target time and the timestamp of the matched sync sample. So this time stamp will be output to as the accurate start time of the following playback segment.

BSAC audio track is somewhat special:

Parser does not only seek this BSAC track. But it will seek all BSAC tracks to the target time and align their reading position. So the user shall take all BSAC tracks as one track and use any BSAC track number to call this function.

NOTE: Seeking to the beginning of the movie (target time is 0 us) does not require the index table.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track to read, 0-based.

usTime [in/out] Target time to seek as input, in us.
Actual seeking time, timestamp of the matched sync sample, as output.

flag [in] Control flags to seek.

SEEK_FLAG_NEAREST

Default flag. The matched time stamp shall be the nearest one to the target time (may be later or earlier)

SEEK_FLAG_NO_LATER

The matched time stamp shall be no later than the target time.

SEEK_FLAG_NO_EARLIER

The matched time stamp shall be no earlier than the target time.

Return value:

PARSER_SUCCESS Seeking succeeds.

PARSER_EOS Can NOT to seek to the target time because of end of stream.

PARSER_ERR_NOT_SEEKABLE

Seeking fails because the movie is not seekable (index not available or no sync samples).

Others Seeking fails for other reason.

2.5.3. MP4GetSyncSample

```
int32 MP4GetNextSyncSample( MP4ParserHandle parserHandle,
                           uint32 trackNum,
                           uint32 direction,
                           uint8 * sampleData,
                           uint32 * dataSize,
                           uint64 * usStartTime,
                           uint64 * usEndTime,
                           uint32 * flag)
```

Description:

Function to get the next or previous sync sample (key frame) from current reading position of a track. For trick mode FF/RW.

It also supports partial output of large samples. If not the entire sample is got, its remaining data can be got by repetitive calling the same function.

Warning: This function does not support BSAC audio tracks because there is dependency between these tracks.

Arguments:

parserHandle [in] Handle of the MP4 core parser.

trackNum [in] Number of the track to read, 0-based.

direction [in] Direction to get the sync sample.
 FLAG_FORWARD
 Read the next sync sample from current reading position.

FLAG_BACKWARD
 Read the previous sync sample from current reading position.

sampleData [in] Buffer to hold the sample data.
 If the buffer is not big enough, only the part of sample is output.

dataSize [in/out] Size of the buffer as input, in bytes.
 As output:
 If a entire sample or part of a sample is output successfully (return value is PARSER_SUCCESS), it's the size of the data actually got.

 If the sample can not be output at all because buffer is too small (the return value is PARSER_INSUFFICIENT_MEMORY), it's the buffer size needed. This is for efficiency consideration of tracks with very small size samples.

usStartTime [out] Start time of the sample in us (timestamp)

usEndTime [out] End time of the sample in us.

flag [out] Flags of this sample, if a sample or part of it is got successfully.

FLAG_SYNC_SAMPLE

Whether this sample is a sync sample (key frame).

For non-video media, the application can take every sample as a sync sample. This flag shall always be SET for this API.

FLAG_SAMPLE_NOT_FINISHED

Sample data output is not finished because the buffer is not big enough. Need to get the remaining data by repetitive calling this function.

Return value:

PARSER_SUCCESS An entire sync sample or part of it is got successfully.

PARSER_ERR_NOT_SEEKABLE No sync sample is got because the movie is not seekable (index not available or no sync samples).

PARSER_EOS Reaching the end of the track, no sync sample is got.

PARSER_BOS Reaching the beginning of the track, no sync sample is got.

PARSER_INSUFFICIENT_MEMORY Buffer is too small to hold the sample.
The user can allocate a larger buffer and call this API again.

PARSER_READ_ERROR File reading error. No need for further error concealment.

Others ... Reading fails for other reason.

3. Data Types & Constants

3.1. Error Codes

Common Error Codes	Comments
PARSER_SUCCESS	success
PARSER_EOS	Reach end of the steam. Not an error.
PARSER_BOS	Reach beginning of the stream. Not an error.
PARSER_ERR_UNKNOWN	Unknown failure, not captured by parser logic.
PARSER_INSUFFICIENT_MEMORY	Insufficient memory, causing allocation or data output failure.
PARSER_NOT_IMPLEMENTED	The API's feature is not implemented yet.
PARSER_ERR_INVALID_PARAMETER	Invalid parameter.
PARSER_INSUFFICIENT_DATA	Data are not, need more data.
PARSER_FILE_OPEN_ERROR	Can not open the movie file.
PARSER_READ_ERROR	Error on file reading. No need for further error concealment
PARSER_WRITE_ERROR	Error on file writing.

PARSER_ERR_INVALID_MEDIA	Invalid or unsupported media format
PARSER_SEEK_ERROR	File system seeking error.
PARSER_ERR_NOT_SEEKABLE	Perform seeking or trick mode on a movie NOT seekable.
PARSER_ERR_CONCEAL_FAIL	Error in bitstream and no sample can be found by error concealment. If the file is seekable, it's better to perform a seeking than further searching the bit stream for the next sample.
DRM_ERR_NOT_AUTHORIZED_USER	Not an authorized user to play a DRM-protected file
DRM_ERR_RENTAL_EXPIRED	The protected rental file is expired (reaching its view limit)
MP4 Specific Error Codes	
MP4NoLargeAtomSupportErr	Large size atoms (64-bit size field) are not supported.
MP4BadDataErr	Data is invalid.
MP4VersionNotSupportedErr	The file format version is not supported.
MP4DataEntryTypeNotSupportedErr	Unknown stream format of a track.
MP4NoQTAtomErr	Not MP4 or QuickTime file format.
MP4_ERR_WRONG_SAMPLE_SIZE	Abnormal large size sample.
MP4_ERR_EMPTY_TRACK	This track has no data and so can not be seeked or read at all.

The common error codes are defined in “fsl_parser.h” and others in “MP4ParserAPI.h”.

3.2. Handle of MP4 Parser

```
typedef void * MP4ParserHandle
```

3.3. Other Constants

3.3.1. Maximum Track Number

MAX_MP4_TRACKS

Maximum media tracks to support for a MP4 movie. Current value is 64.

If there are more tracks, those with a larger track number will be overlooked.

3.3.2. Media & Codec Types

```
typedef enum
{
    MEDIA_TYPE_UNKNOWN = 0,
    MEDIA_VIDEO,
    MEDIA_AUDIO,
    MEDIA_TEXT, /* subtitle text or stand-alone application, string-based or bitmap-based */
    MEDIA_MIDI
}MediaTypes; /* Media types of a track.*/
```

Please see the common header file “fsl_media_types.h” for the media type definitions.

3.3.3. User data ID

```
typedef enum
{
    USER_DATA_TITLE = 0,
    USER_DATA_LANGUAGE, /* user data may tell the language of the movie as a string */
    USER_DATA_GENRE,
    USER_DATA_ARTIST,
    USER_DATA_COPYRIGHT,
    USER_DATA_COMMENTS,
    USER_DATA_CREATION_DATE,
    USER_DATA_RATING
}UserDataID;
```

4. API Calling Sequence

This section describes the API Calling Sequence. Without explicitly explanation, the user shall go ahead only if previous step succeeds.

Please refer to the test application code for more details.

- **Check core parser version (optional)**

MP4ParserVersionInfo()

- **Create the parser**

MP4CreateParser()

As long as the parser is created successfully, it must be deleted as the final step to free the resources.

- **Get properties of the movie and tracks, as well as the user data**

MP4IsSeekable()

MP4GetNumTracks()

...

MP4GetTrackType

...

MP4GetMaxSampleSize() ... (not necessary to save movie loading time)

...

MP4GetUserData()

- **Seek to the beginning of the movie**

Must perform a seeking on all selected tracks, with target time 0 us.

MP4Seek()

Of course, seek to any time within the play duration is reasonable as long as the movie is seekable.

- **Output samples for playback**

(a) For normal playback (rate = 1X), begin to read A/V samples one by one sequentially from the selected tracks. The user chooses which tracks to read.

MP4GetNextSample()

If only part of sample is got due to buffer size restriction, repetitively calling the same function *MP4GetNextSample* until all sample data are got.

(b) For trick mode (FF/RW), can only pick sync sample (video key frames)

MP4GetSyncSample(forward) ... for FAST FORWARD

Or

MP4GetSyncSample(backward) ... for REWIND

If only part of sample is got due to buffer size restriction, repetitively calling the same function *MP4GetSyncSample* until all sample data are got.

(c) A seeking can be performed during the playback. *MP4Seek()*

- **Delete the parser**
MP4DeleteParser()

