



AVI Core Parser Library API Specification

Version: 1.6

Date: Sept. 11, 2009

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	09/28/2009	Amanda Lin	Initial version. Based on AVI core parser library version 2.2
1.1	10/10/2009	Amanda Lin	Support partial output for large samples. New API <code>AviGetSampleRemainingBytes()</code> is added.
1.2	10/15/2009	Amanda Lin	Reviewed. <ol style="list-style-type: none">1. Add API to get a track's duration "AviGetTrackDuration".2. Language parameter removed in API "AviGetUserData".3. Unified the common error codes, media types & constants.
1.3	10/22/2009	Amanda Lin	Added API for text(subtitle) tracks: <code>aviGetTextTrackWidth</code> <code>aviGetTextTrackHeight</code>
1.4	11/02/2009	Amanda Lin	API added to get sample size before reading its data: <code>AviGetNextSampleSize</code> <code>AviGetNextSyncSampleSize</code>
1.5	11/17/2009	Amanda Lin	Add API to get codec specific information: <code>AviGetCodecSpecificInfo</code>
1.6	12/11/2009	Amanda Lin	API changed to get track's type & language: - <code>AviGetTrackType</code> - <code>AviGetLanguage</code>

1. Introduction.....	5
1.1. In this document.....	5
1.2. References.....	5
2. API Functions	5
2.1. Creation and Deletion	5
2.1.1. AviParserVersionInfo	5
2.1.2. AviCreateParser	5
2.1.3. AviDeleteParser	6
2.1.4. AviParseHeader	6
2.2. DRM Features	6
2.2.1. AviIsProtected.....	7
2.2.2. AviQueryContentUsage.....	7
2.2.3. AviQueryOutputProtectionFlag.....	7
2.2.4. AviCommitPlayback.....	8
2.2.5. AviFinalizePlayback	8
2.3. Index Loading	8
2.3.1. AviInitializeIndex	9
2.3.2. AviExportIndex.....	9
2.3.3. AviImportIndex.....	9
2.4. Movie & Track Properties	10
2.4.1. AviGetNumTracks	10
2.4.2. AviGetMovieDuration	10
2.4.3. AviGetTrackDuration	11
2.4.4. AviGetTrackType	11
2.4.5. AviGetCodecSpecificInfo	11
2.4.6. AviGetMaxSampleSize.....	12
2.4.7. AviGetBitRate.....	13
2.4.8. AviGetSampleDuration.....	13
2.4.9. AviGetLanguage	14
2.4.10. AviGetVideoFrameWidth.....	14
2.4.11. AviGetVideoFrameHeight.....	15
2.4.12. AviGetVideoBitsPerPixel	15
2.4.13. AviGetAudioNumChannels.....	15
2.4.14. AviGetAudioSampleRate	16
2.4.15. AviGetAudioBitsPerSample	16
2.4.16. AviGetTextTrackWidth	16
2.4.17. AviGetTextTrackHeight	17
2.5. User Data	17
2.5.1. AviGetUserData.....	17
2.6. Normal Playback.....	18
2.6.1. AviGetNextSample	18
2.6.2. AviGetNextSampleSize	20
2.7. Seek and Trick Mode	20
2.7.1. AviIsSeekable	20
2.7.2. AviSeek.....	21

2.7.3. AviGetSyncSample.....	21
2.7.4. AviGetNextSyncSampleSize	23
3. Data Types & Constants	24
3.1. Error Codes	24
3.2. Handle of AVI Parser.....	25
3.3. Other Constants.....	25
3.3.1. Maximum Track Number	25
3.3.2. Media & Codec Types	25
3.3.3. User data ID	26
3.3.4. Return Value of DRM callback functions	26
4. API Calling Sequence	26

1. Introduction

1.1. In this document

Freescale AVI core parser library provides parsing and data reading functions of local AVI files. Both AVI 1.0 & AVI 2.0 formats are supported.

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

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

1.2. References

[1] AVI File Format by Microsoft (AVI 1.0)

[2] OpenDML AVI File Format Extensions (AVI 2.0)

[3] DivX File Format (*)

* To get full DivX features (playback & DRM), the right packages are needed to work with AVI parser.

2. API Functions

2.1. Creation and Deletion

2.1.1. AviParserVersionInfo

```
const char * AviParserVersionInfo()
```

Description:

Function to get the AVI core parser version.

Return value:

ASCII Version string like "AVI_PARSER_xx.xx.xx".

2.1.2. AviCreateParser

```
AVI_PARSER_ERROR_CODE AviCreateParser(file_stream_t * stream,  
                                       AviMemoryOps * memOps,  
                                       AviDrmOps * drmOps,  
                                       void * context,  
                                       AviParserHandle * parserHandle);
```

Description:

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

Arguments:

- | | |
|--------------------|---|
| stream [in] | Source stream of the AVI 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. |
| drmOps [in] | DRM callback function table.
It implements the functions to read and write local DRM memory. |
| context [in] | Wrapper context for the callback functions. It will be used as an arguments of the callback functions. AVI core parser will never modify its content. |
| parserHandle [out] | Handle of the AVI core parser if the core parser is created successfully.
NULL for failure. |

Return value:

PARSER_SUCCESS - Success
Other error codes - Failure

2.1.3. AviDeleteParser

AVI_PARSER_ERROR_CODE AviDeleteParser(AviParserHandle parserHandle)

Description:

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

Arguments:

- | | |
|-------------------|--------------------------------|
| parserHandle [in] | Handle of the AVI core parser. |
|-------------------|--------------------------------|

Return value:

PARSER_SUCCESS - Success
Other error codes - Failure

2.1.4. AviParseHeader

AVI_PARSER_ERROR_CODE AviParseHeader(AviParserHandle parserHandle)

Description:

Function to parse the AVI file header. It shall be called after AVI parser is created and it will probe whether the movie can be handled by this AVI parser.

Arguments:

- | | |
|-------------------|--------------------------------|
| parserHandle [in] | Handle of the AVI core parser. |
|-------------------|--------------------------------|

Return value:

PARSER_SUCCESS - Success
Other error codes – Failure

2.2. DRM Features

Some AVI movies are protected by DRM. And playback of these movies may need interaction with the user. After the file header is parsed, user shall query the DRM status.

2.2.1. AvilsProtected

AVI_PARSER_ERROR_CODE AvilsProtected(AviParserHandle parserHandle, bool * isProtected)

Description:

DRM interface.function to tell whether movie is protected by DRM.

The wrapper shall call the DRM interface right after the file header is parsed for a quick decision, before doing the time-consuming task such as initialize index.

Only for a DRM-protected file, the following DRM APIs can be called. Otherwise, error "AVI_ERR_DRM_NOT_PROTECTED" will be returned.

Arguments:

parserHandle [in] Handle of the AVI core parser.

isProtected [out] TRUE for protected file. FALSE for a not protected file.

Return value:

PARSER_SUCCESS - Success

Other error codes - Failure

2.2.2. AviQueryContentUsage

AVI_PARSER_ERROR_CODE AviQueryContentUsage(AviParserHandle parserHandle,
 bool * isRental,
 uint32 * viewLimit,
 uint32 * viewCount)

Description:

DRM interface. Function to see whether the file is a rental or purchased movie. This API shall be called once before playing a protected clip.

Arguments:

parserHandle [in] Handle of the avi core parser.

isRental [out] True for a rental file and False for a purchase file. Rental file has a view limit.

viewLimit [out] View limit if a rental file.

viewCount [out] Count of views played already.

Return value:

PARSER_SUCCESS - Success

Other error codes - Failure

2.2.3. AviQueryOutputProtectionFlag

AVI_PARSER_ERROR_CODE AviQueryOutputProtectionFlag(AviParserHandle parserHandle,
 uint8 * cgmsaSignal,
 uint8 * acptbSignal,
 uint8 * digitalProtectionSignal)

Description:

DRM interface. Function to check the video output protection flag.

Arguments:

parserHandle [in] Handle of the AVI core parser.

cgmsaSignal [out] 0, 1, 2, or 3 based on standard CGMSA signaling.

acptbSignal [out] 0, 1, 2, or 3 based on standard trigger bit signaling.
 acptb values:
 0 = Off.
 1 = Auto gain control / pseudo sync pulse.
 2 = Two line color burst.
 3 = Four line color burst.
 digitalProtectionSignal [out] - 0=off, 1=on.

digitalProtectionSignal [out] 0=off, 1=on.

Return value:

PARSER_SUCCESS - Success
 Other error codes - Failure

2.2.4. AviCommitPlayback

AVI_PARSER_ERROR_CODE AviCommitPlayback(AviParserHandle parserHandle)

Description:

DRM interface. Function to commit playing the protected file. The wrapper shall call it before playback is started.

Arguments:

parserHandle [in] Handle of the AVI core parser.

Return value:

PARSER_SUCCESS - Success
 Other error codes - Failure

2.2.5. AviFinalizePlayback

AVI_PARSER_ERROR_CODE AviFinalizePlayback(AviParserHandle parserHandle)

Description:

DRM interface. Function to end playing the protected file. The wrapper shall call it after playback is stopped. Otherwise error "AVI_ERR_DRM_PREV_PLAY_NOT_CLEARED" on next playback.

Arguments:

parserHandle [in] Handle of the AVI core parser.

Return value:

PARSER_SUCCESS - Success
 Other error codes - Failure

2.3. Index Loading

2.3.1. AviInitializeIndex

AVI_PARSER_ERROR_CODE AviInitializeIndex(AviParserHandle parserHandle)

Description:

Function to initialize the index table by scanning the index in the file. For a movie played for the 1st time, the index table has to be loaded from the file. The index table increases with the movie duration. So the longer the movie is, the more time it takes to load the index.

Seeking and trick mode can be performed on a movie only if it has an index. But normal playback does not depend on the index. So even if this function fails, normal playback can still work as long as movie data are right.

Arguments:

parserHandle [in] Handle of the AVI core parser.

Return value:

PARSER_SUCCESS - Success

Other error codes - Failure

2.3.2. AviExportIndex

AVI_PARSER_ERROR_CODE AviExportIndex(AviParserHandle parserHandle, uint32 trackNum, uint8 * buffer, uint32 * size)

Description:

Function to export the index table of a track to outside, after the index table is loaded by scanning the file at 1st play.

This function is usually used on a 1st play movie. This can reduce time to open a 2nd-play movie if its index table has been exported on 1st play.

To save the memory used, import/export a track's index per time.

Arguments:

parserHandle [in] Handle of the AVI core parser.

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

buffer [in] Buffer to export the index data. If this parameter is NULL, just return the size of buffer needed without exporting the index data.

size [in/out] Size of the the buffer as input, in bytes.
Size of the index data in the buffer as output, in bytes.

Return value:

PARSER_SUCCESS - Success

Other error codes - Failure

2.3.3. AviImportIndex

AVI_PARSER_ERROR_CODE AviImportIndex(AviParserHandle parserHandle,
uint32 trackNum,
uint8 * buffer,
uint32 size)

Description:

Function to import the index table of a track from outside, instead of scanning the file.

This can reduce time to open a 2nd-play movie if its index table has been exported on 1st play. To save the memory used, import/export a track's index per time.

Arguments:

parserHandle [in] Handle of the AVI core parser.

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

buffer [in] Buffer containing the index data.

size [in] Size of the index data in the buffer, in bytes.

Return value:

PARSER_SUCCESS - Success
Other error codes - Failure

2.4. Movie & Track Properties

2.4.1. AviGetNumTracks

AVI_PARSER_ERROR_CODE AviGetNumTracks (AviParserHandle parserHandle,
bool * seekable)

Description:

Function to tell how many tracks in the movie.

Arguments:

parserHandle [in] Handle of the AVI core parser.

numTracks [out] Number of tracks.

Return value:

PARSER_SUCCESS - Success
Other error codes - Failure

2.4.2. AviGetMovieDuration

AVI_PARSER_ERROR_CODE AviGetMovieDuration(AviParserHandle parserHandle,
uint64 * usDuration)

Description:

Function to tell the movie duration. Usually it's equal to the video track's duration.

Arguments:

parserHandle [in] Handle of the AVI core parser.

usDuration [out] Duration in us.

Return value:

PARSER_SUCCESS - Success
Other error codes - Failure

2.4.3. AviGetTrackDuration

AVI_PARSER_ERROR_CODE AviGetTrackDuration(AviParserHandle 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).

Warning: For some media track such as subtitle, the duration is ZERO.

Arguments:

parserHandle [in] Handle of the AVI core parser.

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

usDuration [out] Duration in us.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.4.4. AviGetTrackType

AVI_PARSER_ERROR_CODE AviGetTrackType(AviParserHandle 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 AVI 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

2.4.5. AviGetCodecSpecificInfo

AVI_PARSER_ERROR_CODE AviGetCodecSpecificInfo(AviParserHandle parserHandle,

```
uint32 trackNum,
uint8 ** data,
uint32 * size)
```

Description:

Function to tell the codec specific information of a track.

It's the data of stream format atom (strf) of a track.

For video tracks, it's a Windows bitmap header (at least 40 bytes)

For audio tracks, it's a Windows Waveform audio header (if no "wave extra data" is present, it's 16 bytes long; otherwise, it's at least 18 bytes long)

```
typedef struct
```

```
{
    uint32 size;
    uint32 width;
    uint32 height;
    uint16 planes;
    uint16 bitCount; /* bits per pixel */
    uint32 compression;
    uint32 sizeImage;
    uint32 xPelsPerMeter; /* often 0 */
    uint32 yPelsPerMeter;
    uint32 clrUsed;
    uint32 clrImportant;
}BitmapInfo;
```

```
typedef struct
```

```
{
    uint16 formatTag; /* wave format tag */
    uint16 channels; /* 1 for mono, 2 for stereo, more for multi-channels */
    uint32 samplesPerSec;
    uint32 avgBytesPerSec;
    uint16 blockAlign;
    uint16 bitsPerSample;
    uint16 extraSize;
    uint8 extraData[]; /* wave extra data, byte array */
}WaveFormatEx; /* 'strf' for audio stream*/
```

Arguments:

parserHandle [in] Handle of the AVI 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

2.4.6. AviGetMaxSampleSize

```
AVI_PARSER_ERROR_CODE AviGetMaxSampleSize(AviParserHandle parserHandle,
                                           uint32 trackNum,
```

uint32 * size)

Description:

Function to tell the maximum sample size of a track.

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

Warning! The "max sample size" can be zero if the file header information is not complete or index table is not available.

Arguments:

parserHandle [in] Handle of the AVI 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

2.4.7. AviGetBitRate

AVI_PARSER_ERROR_CODE AviGetBitRate(AviParserHandle parserHandle,
 uint32 trackNum,
 uint32 *bitrate)

Description:

Function to tell the bit rate of a track.

For CBR stream, the real bit rate is given.

For VBR stream, 0 is given since the bit rate varies during the playback and AVI parser does not calculate the peak or average bit rate.

Arguments:

parserHandle [in] Handle of the AVI core parser.

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

bitrate [out] Bit rate of the track. For CBR stream, this is the real bit rate.
 For VBR stream, the bit rate is 0 since max bit rate is usually not available.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.4.8. AviGetSampleDuration

AVI_PARSER_ERROR_CODE AviGetSampleDuration(AviParserHandle 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.

Arguments:

parserHandle [in] Handle of the AVI 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

2.4.9. AviGetLanguage

AVI_PARSER_ERROR_CODE AviGetLanguage(AviParserHandle parserHandle,
uint32 trackNum,
uint8 *threeCharCode)

Description:

Function to tell the language of a track used.

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

Arguments:

parserHandle [in] Handle of the AVI 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

2.4.10. AviGetVideoFrameWidth

AVI_PARSER_ERROR_CODE AviGetVideoFrameWidth (AviParserHandle parserHandle,
uint32 trackNum,
uint32 *width)

Description:

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

Arguments:

parserHandle [in] Handle of the AVI 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

2.4.11. AviGetVideoFrameHeight

AVI_PARSER_ERROR_CODE AviGetVideoFrameHeight (AviParserHandle parserHandle,
uint32 trackNum,
uint32 *height)

Description:

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

Arguments:

parserHandle [in] Handle of the AVI 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

2.4.12. AviGetVideoBitsPerPixel

AVI_PARSER_ERROR_CODE AviGetVideoBitsPerPixel (AviParserHandle parserHandle,
uint32 trackNum,
uint32 *bitCount)

Description:

Function to tell the bits per pixel of a video track.

Arguments:

parserHandle [in] Handle of the AVI core parser.

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

bitCount [out] Bits per pixel.

Return value:

PARSER_SUCCESS - Success

Other error codes - Failure

2.4.13. AviGetAudioNumChannels

AVI_PARSER_ERROR_CODE AviGetAudioNumChannels(AviParserHandle parserHandle,
uint32 trackNum,
uint32 *numchannels)

Description:

Function to tell how many channels in an audio track.

Arguments:

parserHandle [in] Handle of the AVI 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

2.4.14. AviGetAudioSampleRate

AVI_PARSER_ERROR_CODE AviGetAudioSampleRate(AviParserHandle parserHandle,
 uint32 trackNum,
 uint32 * sampleRate)

Description:

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

Arguments:

parserHandle [in] Handle of the AVI 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

2.4.15. AviGetAudioBitsPerSample

AVI_PARSER_ERROR_CODE AviGetAudioBitsPerSample (AviParserHandle parserHandle,
 uint32 trackNum,
 uint32 * bitsPerSample)

Description:

Function to tell the bits per sample for a PCM audio track.

Arguments:

parserHandle [in] Handle of the AVI core parser.

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

bitsPerSample [out] Bits per PCM sample.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.4.16. AviGetTextTrackWidth

AVI_PARSER_ERROR_CODE AviGetTextTrackWidth(AviParserHandle 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 AVI 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

2.4.17. AviGetTextTrackHeight

AVI_PARSER_ERROR_CODE AviGetTextTrackHeight (AviParserHandle 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 AVI 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

2.5. User Data

2.5.1. AviGetUserData

AVI_PARSER_ERROR_CODE AviGetUserData(AviParserHandle parserHandle,
 UserDataID id,
 uint8 ** buffer,
 uint32 *size)

Description:

Function to tell the user data information (title, artist, genre etc) of the movie.

Arguments:

parserHandle [in] Handle of the AVI core parser.

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

id [in] Type of the user data.

USER_DATA_TITLE, USER_DATA_GENRE, USER_DATA_ARTIST ...

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.

The information is usually a null-terminated ASCII string.

Return value:

PARSER_SUCCESS - Success

Other error codes – Failure

2.6. Normal Playback

To support normal playback (rate = 1X), AVI 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 AVI movie.

AVI parser supports partial output of large samples, but only for clips not DRM-protected. For DRM-protected clips, it can only output entire samples for decryption need.

2.6.1. AviGetNextSample

```
AVI_PARSER_ERROR_CODE AviGetNextSample( AviParserHandle 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 for clips not protected by DRM. If not the entire sample is got, its remaining data can be got by repetitive calling this function until the whole sample is output.

For A/V tracks, the time stamps of samples are continuous. If a sample is output, its accurate start time and end time are also output.

But for subtitle text tracks, the time stamps of samples are discontinuous and encoded in the sample data. So the parser gives an "estimated" time stamp. The decoder shall decode the accurate time stamp.

Arguments:

parserHandle [in] Handle of the AVI core parser.

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

sampleData [in] Buffer to hold the sample data.
 For DRM-protected clips, it must be large enough to hold the entire sample for decryption need. Otherwise this function will fail.
 For non-protected clips, if the buffer is not big enough, only the 1st 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. Only for DRM-protected files.

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 is got successfully.

FLAG_SYNC_SAMPLE

Whether this sample is a sync sample (key frame).
 For non-video media, the wrapper shall take every sample as sync sample.

FLAG_UNCOMPRESSED_SAMPLE

Whether this sample is a uncompressed one. Uncompressed samples shall bypass the decoder. Warning: Video track may have both compressed & uncompressed samples. But some AVI clips seem to abuse this flag, sync samples are mark as uncompressed, although they are actually compressed ones.

FLAG_SAMPLE_ERR_CONCEALED

There is error in bitstream but a sample is still got by error concealment.

FLAG_SAMPLE_SUGGEST_SEEK

A seeking on ALL tracks is suggested although samples can be got by error concealment. Because there are many corrupts, and A/V sync is likely impacted by simple concealment(scanning bitstream).

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.
 This feature is only for non-protected clips.

Return value:

`PARSER_SUCCESS` - An entire sample or 1st 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.
 Only for DRM-protected files.
 The user can allocate a larger buffer and call this API again.

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

PARSER_ERR_CONCEAL_FAIL - There is error in bitstream, and no sample can be found by error concealment. A seeking is helpful.

Others - Other failures.

2.6.2. AviGetNextSampleSize

```
AVI_PARSER_ERROR_CODE AviGetNextSampleSize( AviParserHandle parserHandle,
                                             uint32 trackNum,
                                             uint32 * sampleSize)
```

Description:

Function to tell the next sample size of a track.

This is an optional API. No matter it's called or not, the parser is able to output the sample data properly.

This function shall be called when current sample output is finished. Otherwise it will give the size of current sample.

Warning:

If the sample size is an ODD number, there is one padding byte will be read by the parser after the sample data. The user shall prepare a buffer big enough to hold the padding byte. If the sample size is EVEN, no such need.

Arguments:

parserHandle [in] Handle of the AVI core parser.

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

sampleSize [out] Size of the next sample, in bytes.

2.7. Seek and Trick Mode

An AVI movie is seekable if and only if it has the index table no matter AVI 1.0 or AVI 2.0. And seeking and trick mode (FF/RW) can be performed only on a seekable AVI movie. Seeking is also track-based like sample reading.

2.7.1. AvilsSeekable

```
AVI_PARSER_ERROR_CODE AvilsSeekable(AviParserHandle parserHandle, bool * seekable)
```

Description:

Function to tell whether the movie is seekable. A seekable AVI movie must have the index table. If the file's index table is loaded from file or imported successfully, it's 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 AVI core parser.

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

Return value:

PARSER_SUCCESS - Success

Other error codes - Failure

2.7.2. AviSeek

AVI_PARSER_ERROR_CODE AviSeek(AviParserHandle 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.

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 AVI 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_ERR_NOT_SEEKABLE

Seeking fails because the movie is not seekable (index not available).

Others Seeking fails for other reason.

2.7.3. AviGetSyncSample

AVI_PARSER_ERROR_CODE AviGetNextSyncSample(AviParserHandle 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 supports partial output of large samples for clips not protected by DRM. If not the entire sample is got, its remaining data can be got by repetitive calling the same function.

Arguments:

parserHandle [in] Handle of the AVI 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.
 For DRM-protected clips, it must be large enough to hold the entire sample for decryption need. Otherwise this function will fail.
 For non-protected clips, if the buffer is not big enough, only the 1st 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 PARSE_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 PARSE_INSUFFICIENT_MEMORY), it's the buffer size needed. Only for DRM-protected files.

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 is got successfully.

FLAG_SYNC_SAMPLE
 Whether this sample is a sync sample (key frame).
 For non-video media, the wrapper shall take every sample as sync sample.
 This flag shall always be SET for this API.

FLAG_UNCOMPRESSED_SAMPLE
 Whether this sample is a uncompressed one. Uncompressed samples shall bypass the decoder. Warning: Video track may have both compressed & uncompressed samples. But some AVI clips seem to abuse this flag, sync samples are mark as uncompressed, although they are actually compressed ones.

FLAG_SAMPLE_ERR_CONCEALED
 There is error in bitstream but a sample is still got by error concealment.

FLAG_SAMPLE_SUGGEST_SEEK

A seeking on ALL tracks is suggested although samples can be got by error concealment. Because there are many corrupts, and A/V sync is likely impacted by simple concealment(scanning bitstream).

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.
This feature is only for non-protected clips.

Return value:

PARSER_SUCCESS An entire sync sample or 1st part of it is got successfully.

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

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.
Only for DRM-protected files.

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

PARSER_ERR_CONCEAL_FAIL There is error in bit stream, and no sample can be found by error concealment. A seeking is helpful.

Others ... Reading fails for other reason.

2.7.4. AviGetNextSyncSampleSize

AVI_PARSER_ERROR_CODE AviGetNextSyncSampleSize(AviParserHandle parserHandle,
uint32 trackNum,
uint32 direction,
uint32 * sampleSize)

Description:

Function to tell the size of next or previous sync sample of a track

This is an optional API. No matter it's called or not, the parser is able to output the sample data properly.

This function shall be called when current sample output is finished. Otherwise it will give the size of current sample.

Warning:

If the sample size is an ODD number, there is one padding byte will be read by the parser after the sample data. The user shall prepare a buffer big enough to hold the padding byte. If the sample size is EVEN, no such need.

Arguments:

parserHandle [in] Handle of the AVI core parser.

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

direction [in] Direction to get the sync sample.

FLAG_FORWARD To get the next sync sample from current reading position.

FLAG_BACKWARD To get the previous sync sample from current reading position.

sampleSize [out] Size of the next sync sample, in bytes.

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.
PARSER_INSUFFICIENT_MEMORY	Insufficient memory.
PARSER_NOT_IMPLEMENTED	The feature is not implemented yet.
PARSER_ERR_INVALID_PARAMETER	Invalid parameter.
PARSER_INSUFFICIENT_DATA	Data are not enough for header paring or index loading.
PARSER_FILE_OPEN_ERROR	Can not open the movie file.
PARSER_READ_ERROR	Error on file reading. Need for further error concealment
PARSER_WRITE_ERROR	Error on file writing.
PARSER_SEEK_ERROR	Error on file seeking.
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.
PARSER_ERR_CUR_SAMPLE_NOT_FINISHED	Current sample output is not finished yet. Can not get the next one.
PARSER_ERR_CUR_SAMPLE_FINISHED	Current sample is finished and no remaining data can be output
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)
AVI Specific Error Codes	
AVI_ERR_NOT_AVI_FILE	This is not an AVI file. No RIFF AVI header found.
AVI_ERR_WRONG_FILE_SIZE	File size is smaller than it shall be.
AVI_ERR_UNKNOWN_STREAM_FORMAT	Unknown stream format of a track.
AVI_ERR_WRONG_MEDIA_TYPE	An API is called on a track of wrong media type. Eg. To get video width on an audio track.
AVI_ERR_WRONG_AUDIO_FORMAT	Unknown audio codec type.

AVI_ERR_WRONG_VIDEO_FORMAT	Unknown video codec type.
AVI_ERR_INVALID_ATOM	Invalid AVI atom found in parsing phase.
AVI_ERR_CORRUPTED_INDEX	Index table is corrupted.
AVI_ERR_WRONG_INDEX_SAMPLE_SIZE	The index give wrong sample size
AVI_ERR_WRONG_INDEX_SAMPLE_OFFSET	The index give wrong sample offset
AVI_ERR_WRONG_IDX1_LIST_SIZE	Size of AVI 1.0 index table is wrong.
AVI_ERR_WRONG_AVI2_INDEX_SIZE	Size of AVI 2.0 index table is wrong.
AVI_ERR_WRONG_AVI2_INDEX_ENTRY_SIZE	Entry size of AVI 2.0 index is wrong.
AVI_ERR_INDEX_TYPE_NOT_SUPPORTED	The AVI 2.0 index type is not supported. Filed index is not supported now.
AVI_ERR_SUPER_INDEX_ENTRY_NOT_FOUND	The super index entry is not found.
AVI_ERR_EMPTY_INDEX	The AVI 2.0 index table has no entries.
AVI_ERR_DRM_NOT_PROTECTED	Calling DRM APIs for a not protected clip
AVI_ERR_DRM_INVALID_CONTEXT	DRM context is invalid
AVI_ERR_DRM_PREV_PLAY_NOT_CLEARED	Not reset DRM status for the 2nd playback.
AVI_ERR_DRM_INVALID_CALLBACK	Invalid DRM callback.
AVI_ERR_DRM_OTHERS	Reserved for other DRM errors.
AVI_ERR_WRONG_DRM_INFO_SIZE	Size of DRM info is wrong.
AVI_ERR_ZERO_STREAM_RATE	The rate of a track is ZERO. Can not calculate time stamp.
AVI_ERR_NO_INDEX	There is no index, but not affect normal playback
AVI_ERR_INDEX_ALREADY_LOADED	The index table is already loaded or imported
AVI_ERR_NO_USER_DATA	The user data is not available
AVI_ERR_WRONG_CHUNK	Invalid chunk ID.
AVI_ERR_WRONG_CHUNK_SIZE	Invalid chunk size, greater than the total movie size or maximum sample size of the track.
AVI_ERR_WRONG_TRACK_NUM	Invalid chunk number.
AVI_ERR_WRONG_MOVIE_LIST_SIZE	Invalid movie data size.

3.2. Handle of AVI Parser

```
typedef void * AviParserHandle
```

3.3. Other Constants

3.3.1. Maximum Track Number

MAX_AVI_TRACKS

Maximum media tracks to support for an AVI movie. Current value is 24.
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

}MediaType; /* 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;
```

3.3.4. Return Value of DRM callback functions

```
#define AVI_DRM_LOCAL_SUCCESS    0
#define AVI_DRM_ERROR_READING_MEMORY  12
#define AVI_DRM_ERROR_WRITING_MEMORY  13
```

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)**

AviParserVersionInfo()

- **Create the parser**

AviCreateParser()

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

- **Probe the file by parsing its header**

AviParseHeader()

- **Check DRM information**

- (1) Whether this movie file is protected by DRM? *AvilsProtected()*
- (2) If protected, is it a rental file or a purchased file? *AviQueryContentUsage()*
- (3) Whether protection is required for digital video output signals?
AviQueryOutputProtectionFlag()
- (4) If the user wants to play the protected clip, confirm playback.
AviCommitPlayback()
Otherwise, delete the parser object and exit.

- **Initialize index table**

- (a) For the 1st play, initialize the index table by loading it from the movie file. It can be time consuming for a long movie.
AvilInitializeIndex()

And then export the index table to the external database, for fast opening it next time (optional)
AviExportIndex()

- (b) For a movie not playing for the 1st time, if its index has be exported before, import the index table from the external database,
AvilImportIndex()
It's much faster than loading again from the file. Of course, you call still choose to use *AvilInitializeIndex()*.

NOTE: If the index table loading fails, normal playback will not be affected. The user can still seek to the beginning of the movie and start reading A/V samples.

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

AvilsSeekable()
AviGetNumTracks()
...
AviGetTrackType
...
AviGetMaxSampleSize()
AviGetUserData()

NOTE: *AviGetMaxSampleSize()* call tell you how big a buffer is enough to hold the largest sample of a track. But this value can be ZERO if the information is neither in the file header nor index table.

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

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

Of course, seek to any time within the play duration is reasonable.

- **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.

AviGetNextSampleSize() ... optional

AviGetNextSample()

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

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

AviGetNextSyncSampleSize(forward) ... optional

AviGetSyncSample(forward) ... for FAST FORWARD

Or

AviGetNextSyncSampleSize(backward) ... optional

AviGetSyncSample(backward) ... for REWIND

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

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

- **Reset DRM state when playback ends (optional)**

This step is only required for a DRM-protected clip: *AviFinalizePlayback()*

- **Delete the parser**

AviDeleteParser()

