# Creating Custom FFmpeg IO-Context

**Philipp Sch**, 6 Nov 2012

★★★★★ 4.89 (8 votes) Rate this:

How to use an IStream-Interface with FFmpeg

## Introduction

In this short Article I will explain how to use a custom IO-Context with FFmpeg. Although I used an IStream Object the code can be used for other Streams like `std::istream`.

FFmpeg can only read from files or named pipes easily, but if you want to read directly from memory, from sockets or IStreams you have to provide a custom IO-Context. I could not find any resource in the internet which profferes a complete and working solution with the current version of FFmpeg explaining how to deal correctly with an IO-Context. After some hours of experimentation I finally managed to get this working without getting access-violations in FFmpeg functions.

## Creating the IO-Context

FFmpeg uses a custom IO-Context, when you allocate the `AVFormatContext`-structure yourself and provide your own version of `AVIOContext` but there are several other things to consider. At first we will create the AVIOContext and the `AVFormatContext` structures. The size of the internal buffer is up to you, I decided to provide 32kb for internal buffering. The two functions `ReadFunc` and `SeekFunc` are shown later.

Hide Copy Code

```
// IStream-Interface that was already created elsewhere:
IStream* pInStream;

// Create internal Buffer for FFmpeg:
const int iBufSize = 32 * 1024;
BYTE* pBuffer = new BYTE[iBufSize];

// Allocate the AVIOContext:
// The fourth parameter (pStream) is a user parameter which will be passed to our
callback functions
AVIOContext* pIOCtx = avio_alloc_context(pBuffer, iBufSize,  // internal Buffer and its
size
                                         0,                  // bWriteable
(1=true,0=false)
                                         pInStream,          // user data ; will be
passed to our callback functions
                                         ReadFunc,
                                         0,                  // Write callback function
(not used in this example)
```

```
                                                SeekFunc);

// Allocate the AVFormatContext:
AVFormatContext* pCtx = avformat_alloc_context();

// Set the IOContext:
pCtx->pb = pIOCtx;
```

Note: As you can see, the custom IO-Context can also be used for writing, but this is not explained here.

Now you have to tell FFmpeg, which input format it has to use. For a custom IO-Context this is necessary! FFmpeg will otherwise read *about* 5Mb data from the stream by default to determine the input format. By doing this, FFmpeg will crash because of a buffer overrun. I have not tested whether it will work, if the internal buffer is large enough to hold the 5Mb data because it was easier for me to determine the input format on my own.

Hide Copy Code

```
// Determining the input format:
ULONG ulReadBytes = 0;
if(FAILED(pInStream->Read(pBuffer, iBufSize, &ulReadBytes)))
    // Error Handling...

// Don't forget to reset the data pointer back to the beginning!
if(FAILED(pInStream->Seek(0, SEEK_SET)))
    // Error Handling...

// Now we set the ProbeData-structure for av_probe_input_format:
AVProbeData probeData;
probeData.buf = pBuffer;
probeData.buf_size = ulReadBytes;
probeData.filename = "";

// Determine the input-format:
pCtx->iformat = av_probe_input_format(&probeData, 1);
```

The last thing to do is to set the flags of the `AVFormatContext`. This is not directly mentioned in the documentation and although FFmpeg realizes that you have set your own `AVIOContext` you have to set the `AVFMT_FLAG_CUSTOM_IO`-flag on your own.

Hide Copy Code

```
pCtx->flags = AVFMT_FLAG_CUSTOM_IO;
```

Now we use the avformat_open_input function to tell FFmpeg that it can start to read from the stream.

Hide Copy Code

```
if(avformat_open_input(&pCtx, "", 0, 0)) != 0)
    // Error Handling
```

The second parameter of avformat_open_input is the filename. This is not used, because we want to use the custom IO-Context. Older versions of FFmpeg will crash if you pass 0 as filename instead of "". This issue was fixed in newer versions of FFmpeg.

The callback functions `ReadFunc` and `SeekFunc` are easily implemented:

Hide Copy Code

```
int ReadFunc(void* ptr, uint8_t* buf, int buf_size)
{
    IStream* pStream = reinterpret_cast<IStream*>(ptr);
    ULONG bytesRead = 0;
    HRESULT hr = pStream->Read(buf, buf_size, &bytesRead);
    if(hr == S_FALSE)
        return AVERROR_EOF;  // Let FFmpeg know that we have reached eof
    if(FAILED(hr))
        return -1;
    return bytesRead;
}
// whence: SEEK_SET, SEEK_CUR, SEEK_END (like fseek) and AVSEEK_SIZE
int64_t SeekFunc(void* ptr, int64_t pos, int whence)
```

```
{
    // Quelle Abfragen:
    IStream* pStream = reinterpret_cast<IStream*>(ptr);

    // Seek:
    LARGE_INTEGER in = { pos };
    ULARGE_INTEGER out = { 0 };
    if(FAILED(pStream->Seek(in, whence, &out)))
         return -1;

    // Return the new position:
    return out.QuadPart;
}
```

The whence-parameter has one more option than fseek: `AVSEEK_SIZE`. When this option is passed to the seek function it should return the file size (if possible). If its not possible, the function may return and do nothing -1. In my implementation `pStream->Seek(...)` will fail with `AVSEEK_SIZE` and `SeekFunc` will return -1.

## Freeing resources

One last comment on which functions are to use to release all the allocated resources:

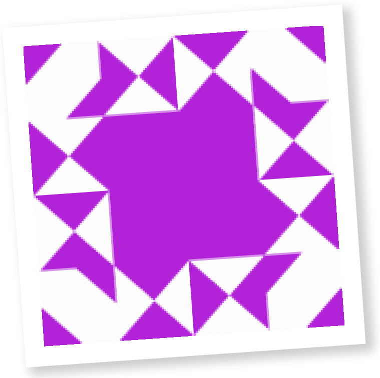
Hide Copy Code

```
avformat_close_input(pCtx);  // AVFormatContext is released by avformat_close_input
av_free(pIOCtx);             // AVIOContext is released by av_free
delete[] pBuffer;
```

## License

## Share

## About the Author



**Philipp Sch**

Germany

No Biography provided

## You may also be interested in...

A Solution Blueprint for DevOps

Solution Build Timer for VS 2005/2013/2015/2017/2019

FFmpeg Tutorial

Dynamic Treeview with Drag and Drop by Kendo

Invoke FFmpeg to export frames

Customized Android ListView with Image and Text

## Comments and Discussions

**You must Sign In to use this message board.**

Search Comments

First Prev Next

**Custom ffmpeg io context in android(with java)**
**Member 12840631 7-Apr-18 9:37**

**seek_func for AVIOContext**
**Iday 6-Apr-15 19:54**

Re: seek_func for AVIOContext
**Philipp Sch** 13-Jul-15 7:53

**Question regarding buffer size**
**Member 10650165 15-Mar-14 16:38**

**My vote of 5**
**CallMeZh 10-Mar-13 21:32**

Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.