

The AppleEvent Stream Library

Apple Events

The Æ Stream Library

Version 1.0

Jens Peter Alfke

13 August 1991

The AppleEvent Builder/Printer

The AppleEvent Stream Library
User Programming Group
© Apple Computer, Inc. 1991

The AppleEvent Builder/Printer

The AppleEvent Stream Library

The AppleEvent Builder/Printer

Contents

Contents.....	iii
Introduction.....	1
OK, What Is It?	1
What's In It For Me?	1
Just How Stable Is It, Anyway?	2
What Are All Those Files?	2
Disclaimer	2
AStream Functions.....	3
Opening and Closing a Stream	3
Writing Descriptors	3
Writing Lists	5
Writing Records	5
Writing Key Descriptors For Records	6
The Header Files.....	7
AStream.h	7
AStream_CPlus.h	8

Introduction

OK, What Is It?

The Apple Event Manager routines that assemble descriptors (I speak here of `AECreatDesc`, `AECreatList`, `AEPutDesc`, *et al*) provide very flexible random access at the expense of significant overhead in speed and memory. Constructing nested structures involves duplicating and copying a whole lot of data. In many cases, programs create descriptors in a more or less linear fashion, without many common subexpressions. In such situations you could use a stream-like protocol and speed things up quite a bit.

That's what the AESTream library does for you. It doesn't use the Apple Event Manager at all. Instead, it builds up a single descriptor from beginning to end. The descriptor data stays all in one block and grows in discrete increments, so there will be far fewer Memory Manager calls.

What's In It For Me?

The benefits of using this library are threefold:

- ✱ Sheer, raw speed. It's about three times as fast as using the Apple Event Manager routines.
- ✱ Code that creates nested descriptors in a linear fashion will be simpler and clearer.
- ✱ Simpler cleanup; there's only one stream object, not many sub-descriptors, to be disposed.

Just How Stable Is It, Anyway?

I've tested this code out, and verified it runs reliably on a moderately complex expression, produces exactly the same descriptors as do the Apple Event

The AppleEvent Stream Library

Manager routines, and has no memory leakage. I did most of this testing by converting my `AEBuild` function to use the `AESStream` library. `AEBuild` now runs just as reliably and much faster.

My inside knowledge of the structure of compound Apple Event descriptors comes from Mr. Apple Event Manager himself, Ed Lai.

This library has not, however, been extensively tested. (See the disclaimer below.)

What Are All Those Files?

Here's what you get:

Release Notes	Notes on the latest release.
AESStream.doc	This document.
AESStream.o	The library itself, in MPW format.
AESStream.h	C header file
AESStream_CPlus.h	C++ header file. C++ users can <code>#include</code> this file or <code>AESStream.h</code> ; either way, this file will be read.

How About Access From Pascal?

As far as I know, there's no reason why you couldn't call this library from Pascal, whether the MPW or THINK variety. All you need to do is write a Pascal header equivalent to `AESStream.h`. (Keep in mind that all the functions have use C calling conventions.) I haven't provided you such a header because my Pascal expertise is pretty rusty and I'm not sure I'd get the syntax right. If you'd like to send me your Pascal header for inclusion in the next release, feel free...

Disclaimer

THIS SOFTWARE HAS NOT BEEN PAINSTAKINGLY TESTED BY APPLE'S RUTHLESSLY EFFICIENT QUALITY ENGINEERS. NEITHER APPLE COMPUTER, INCORPORATED, NOR THE AUTHOR OF THIS SOFTWARE MAKE ANY LEGALLY BINDING CLAIM THAT THIS SOFTWARE WILL DO ANYTHING IN PARTICULAR BESIDES USE UP VALUABLE SPACE ON A CD OR HARD DISK. IN THE EVENT THAT YOUR USE OF OR INABILITY TO USE THIS SOFTWARE RESULTS IN A VISITATION FROM MACSBUG, DAMAGE TO OTHER SOFTWARE OR HARDWARE, THE EXPLOSION OF YOUR MACINTOSH IN A SHOWER OF SPARKS (AS SEEN ON STAR TREK®) OR INDEED THE END OF WESTERN CIVILIZATION AS WE KNOW IT, YOUR ATTEMPTS TO ATTACH BLAME ONTO APPLE COMPUTER, INCORPORATED OR THE AUTHOR OF THIS SOFTWARE WILL BE EXPENSIVE AND UNSUCCESSFUL. HAVE A NICE DAY.

AESStream Functions

Opening and Closing a Stream

```
OSErr AESStream_Open( AESStreamRef s )
```

`AESStream_Open` opens the stream structure pointed to by `s` as a new, empty stream. You must do this before calling any other `AESStream` functions. For example:

```
AESStream myStream;
err= AESStream_Open(&myStream);
// more stream calls...
```

```
OSErr AESStream_Close( AESStreamRef s, AEDesc *desc )
```

`AESStream_Close` closes the stream. If `desc` is `NIL`, the data in the stream will be disposed with no questions asked. (Use this if you need to abort due to an error.) Otherwise, the finished descriptor will be copied into the `AEDesc` pointed to by `desc`. If the descriptor is not finished, the error code `errAESStream_BadNesting` will be returned.

After closing a stream, you can re-open it with `AESStream_Open` and re-use it.

Writing Simple Descriptors

△ **Warning** Do not use these routines to write list or record descriptors! You write those by using routines described in following sections. △

```
OSErr AESStream_WriteDesc( AESStreamRef s, DescType type, void *data, Size length );
```

`AESStream_WriteDesc` appends an arbitrary block of data to the stream as a descriptor, much like `AEPutPtr`.

The AppleEvent Stream Library

```
OSErr AESTream_WriteAEDesc( AESTreamRef s, AEDesc *desc );
```

`AESTream_WriteAEDesc` appends a prepackaged Apple Event descriptor to the stream `s`, much like `AEPutDesc`. The descriptor `desc` is not disposed by the call; if you don't need it anymore, dispose it yourself by calling `AEDisposeDesc`.

```
OSErr AESTream_OpenDesc( AESTreamRef s, DescType type, AESTreamMarkRef mark );
```

```
OSErr AESTream_WriteData( AESTreamRef s, void *data, Size length );
```

```
OSErr AESTream_CloseDesc( AESTreamRef s, AESTreamMarkRef mark );
```

Use these three routines if you want to write a descriptor piece by piece. First call `AESTream_OpenDesc`, then call `AESTream_WriteData` zero or more times to write zero or more bytes of data, then call `AESTream_CloseDesc` to end the descriptor.

Here's the funny part: `AESTream_OpenDesc` will hand you a small data structure called a *mark*. (You pass the address of an `AESTreamMark` and `AESTream_OpenDesc` writes the data to it.) You must give this *same* mark data to the matching `AESTream_CloseDesc` call. In between, while you're holding onto the mark, you may not change it yourself or give it to any `AESTream` routines.

Usually you use a local variable for the mark, like so:

```
AESTreamMark mark;
err= AESTream_OpenDesc(s, type, &mark);
err= AESTream_WriteData(s, ... );
// ...
err= AESTream_CloseDesc(s, &mark);
```

- ◆ The astute reader will have recognized that this mark scheme is `AESTream`'s way of getting the caller to maintain a stack for it. Pieces of state (pointers to the start of the data) need to be kept around while writing a descriptor, and since descriptors can be nested, so can the state. The stream library pushes the old state information to you as a mark when a descriptor begins, and pops it back from you when it ends. ◆

Writing Lists

```
OSErr AESTream_OpenList( AESTreamRef s, AESTreamMarkRef mark );
```

```
OSErr AESTream_CloseList( AESTreamRef s, AESTreamMarkRef mark );
```

To write a list, call `AESTream_OpenList`, write zero or more descriptors, then call `AESTream_CloseList`. The descriptors can be simple descriptors (see above), lists or records (see below).

- ◆ These functions, like `AESTream_OpenDesc` and `AESTream_CloseDesc`, ask you to hold onto a mark for them. See the previous section for a discussion of proper mark etiquette. ◆

Writing Records

```
OSErr AESTream_OpenRecord( AESTreamRef s, DescType type, AESTreamMarkRef mark );
```

```
OSErr AESTream_CloseRecord( AESTreamRef s, AESTreamMarkRef mark );
```

To write a record, call `AESTream_OpenRecord`, write zero or more *key descriptors* (see below), then call `AESTream_CloseRecord`. `AESTream_OpenRecord` lets you specify the type to which the record should be coerced; use the constant `typeAERecord` if you want an uncoerced record.

- ◆ These functions, like `AESTream_OpenDesc` and `AESTream_CloseDesc`, ask you to hold onto a mark for them. See above for a discussion of proper mark etiquette. ◆

Writing Key Descriptors For Records

- △ **Warning** Write key descriptors only *within* records, and write *only* key descriptors within records, or you will end up with a bogus descriptor that will crash the Apple Event Manager! △

The AppleEvent Stream Library

```
OSErr AESTream_WriteKeyDesc( AESTreamRef s, DescType key,
                             DescType type, void *data, Size length );
```

AESTream_WriteKeyDesc writes a complete key descriptor. It's identical to AESTream_WriteDesc except for the key parameter which specifies the keyword to use for this descriptor.

```
OSErr AESTream_OpenKeyDesc( AESTreamRef s, DescType key, DescType type,
                             AESTreamMarkRef mark );
```

AESTream_OpenKeyDesc is identical to AESTream_OpenDesc except for the key parameter which specifies the keyword to use for this descriptor. Use AESTream_CloseDesc to end the descriptor.

```
OSErr AESTream_WriteKey( AESTreamRef s, DescType key );
```

AESTream_WriteKey writes the keyword to be used by the immediately following descriptor, list or record.

△ **Warning** The next AESTream call after AESTream_WriteKey *must* begin a descriptor — it must be _WriteDesc, _OpenDesc, _OpenList or _OpenRecord — or you will end up with a bogus descriptor that will crash the Apple Event Manager! △

AESTream_WriteKey is the only way to add a list or record as a field of a record. It's also useful if you are writing a record and want to call a subroutine that writes a descriptor:

```
err= AESTream_OpenRecord(s, typeAERecord, &mark);
err= AESTream_WriteKey(s, kMyKey);           // this is the key
writeMyDescriptor(s);                       // for this descriptor
err= AESTream_CloseRecord(s, &mark);
```

The Header Files

Here for your convenience are printouts of the header files as of 12 August 1991.

AESTream.h

```
////
////  AESTream.h          A (write-only) stream for creating AE Descriptors.
////                      This header file automatically uses AESTream_CPlus in C++.
////
////  By Jens Alfke       ©1991 Apple Computer, Inc. All rights reserved.
////

#ifdef __cplusplus
    #include "AESTream_CPlus.h"      /* C++ programs use C++ header instead */
#else

// NOTE: In case of disagreement between this header and the C++ one (AESTream_CPlus.h),
//       the C++ header is correct and this header needs to be fixed.

#ifndef __AESTREAM__
    #define __AESTREAM__

#define errAESTream_BadNesting  13579 /* Bad descriptor/array nesting error */

typedef struct {                // Mark descriptor
    Size  sizeIndex;
    Size  countIndex;
} AESTreamMark, *AESTreamMarkRef;

typedef struct {                // A (write-only) stream on an AE descriptor
    Handle data;                // The data
    Size  index;                // Current index (into data handle) to write to
    AESTreamMark mark;          // Current mark: Index to size/length field
                                // of open desc/array/record
}
```

The AppleEvent Stream Library

```
    Size    size;           // Current size of handle
} AESTream, *AESTreamRef;
```

The AppleEvent Builder/Printer

The AppleEvent Stream Library

```
OSErr AESTream_Open    ( AESTreamRef ),
    AESTream_Close     ( AESTreamRef, AEDesc *desc ),

    AESTream_OpenDesc   ( AESTreamRef, DescType type, AESTreamMarkRef mark ),
    AESTream_WriteData  ( AESTreamRef, void *data, Size length ),
    AESTream_CloseDesc  ( AESTreamRef,                      AESTreamMarkRef mark ),

    AESTream_WriteDesc  ( AESTreamRef, DescType type, void *data, Size length ),
    AESTream_WriteAEDesc( AESTreamRef, AEDesc *desc ),

    AESTream_OpenList   ( AESTreamRef, AESTreamMarkRef mark ),
    AESTream_CloseList  ( AESTreamRef, AESTreamMarkRef mark ),

    AESTream_OpenRecord ( AESTreamRef, DescType type, AESTreamMarkRef mark ),
    AESTream_CloseRecord( AESTreamRef,                      AESTreamMarkRef mark ),

    AESTream_WriteKeyDesc( AESTreamRef, DescType key,
                            DescType type, void *data, Size length ),
    AESTream_OpenKeyDesc( AESTreamRef, DescType key,
                            DescType type, AESTreamMarkRef mark ),
    AESTream_WriteKey    ( AESTreamRef, DescType key );

#endif

#endif
```

AESTream_CPlus.h

```
////
////  AESTream_CPlus.h    A (write-only) stream for creating AE Descriptors.
////                      This header file for use with C++. Use AESTream.h with C.
////
////  By Jens Alfke       ©1991 Apple Computer, Inc. All rights reserved.
////

// NOTE: This header file is for C++ programs only. If you #include "AESTream.h" in a C++
//        program, you'll get this header anyway.

// NOTE: In case of disagreement between this header and the C one (AESTream.h),
//        this header is correct and the C header needs to be fixed.
```

The AppleEvent Builder/Printer

The AppleEvent Stream Library

```
#ifndef __AESTREAM__
#define __AESTREAM__

#ifndef __MEMORY__
#include <Memory.h>
#endif
#ifndef __APPLEEVENTS__
#include <AppleEvents.h>
#endif

const errAESTream_BadNesting = 13579; // Bad descriptor/array nesting error

// Here are the C-style definitions, which are the actual functions implemented:

struct AESTream;
struct AESTreamMark;

extern "C" {
    OSErr
        AESTream_Open      ( AESTream& ),
        AESTream_Close      ( AESTream&, AEDesc *desc ),

        AESTream_WriteDesc ( AESTream&, DescType type, const void *data, Size length ),
        AESTream_WriteAEDesc( AESTream&, const AEDesc &desc ),

        AESTream_OpenDesc   ( AESTream&, DescType type, AESTreamMark &mark ),
        AESTream_WriteData   ( AESTream&, const void *data, Size length ),
        AESTream_CloseDesc   ( AESTream&, const AESTreamMark &mark ),

        AESTream_OpenList    ( AESTream&, AESTreamMark &mark ),
        AESTream_CloseList    ( AESTream&, const AESTreamMark &mark ),

        AESTream_OpenRecord ( AESTream&, DescType type, AESTreamMark &mark ),
        AESTream_CloseRecord( AESTream&, const AESTreamMark &mark ),

        AESTream_WriteKeyDesc( AESTream&, DescType key,
                                DescType type, const void *data, Size length ),
        AESTream_OpenKeyDesc( AESTream&, DescType key,
                                DescType type, AESTreamMark &mark ),
        AESTream_WriteKey    ( AESTream&, DescType key );
}

// Here are the data structures, complete with fancy C++ inline methods
// to call the above fns:

struct AESTreamMark { // Mark descriptor
```

The AppleEvent Builder/Printer

The AppleEvent Stream Library

```
Size    sizeIndex;
Size    countIndex;
};

struct AESTream {          // A (write-only) stream on an AE descriptor
    Handle data;           // The data
    Size    index;         // Current index (into data handle) to write to
    AESTreamMark mark;     // Current mark: Index to size/length field
                           //      of open desc/array/record
    Size    size;          // Current size of handle

    AESTream();
    ~AESTream();

    inline OSErr
        Close      ( AEDesc *desc )
                        {return AESTream_Close(*this,desc);}

        OpenDesc   ( DescType type, AESTreamMark &mark )
                        {return AESTream_OpenDesc(*this,type,mark);}
        WriteData  ( const void *data, Size length )
                        {return AESTream_WriteData(*this,data,length);}
        CloseDesc  ( const AESTreamMark &mark )
                        {return AESTream_CloseDesc(*this,mark);}

        WriteDesc  ( DescType type, const void *data, Size length )
                        {return AESTream_WriteDesc(*this,type,data,length);}
        WriteDesc  ( const AEDesc &desc )
                        {return AESTream_WriteAEDesc(*this,desc);}

        OpenList   (      AESTreamMark &mark )
                        {return AESTream_OpenList(*this,mark);}
        CloseList  ( const AESTreamMark &mark )
                        {return AESTream_CloseList(*this,mark);}

        OpenRecord ( DescType type, AESTreamMark &mark )
                        {return AESTream_OpenRecord(*this,type,mark);}
        CloseRecord (      const AESTreamMark &mark )
                        {return AESTream_CloseRecord(*this,mark);}

        WriteKeyDesc( DescType key, DescType type, void *data, Size length )
                        {return AESTream_WriteKeyDesc(*this,key,type,data,length);}
        OpenKeyDesc ( DescType key, DescType type, AESTreamMark &mark )
                        {return AESTream_OpenKeyDesc(*this,key,type,mark);}
        WriteKey   ( DescType key )
                        {return AESTream_WriteKey(*this,key);}
};
```

The AppleEvent Builder/Printer

The AppleEvent Stream Library

The AppleEvent Builder/Printer