Legacy ATSUI Reference

(Legacy)

Carbon > Text & Fonts

2006-07-24

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Legacy ATSUI Reference (Legacy)

Overview

Important: The Legacy ATSUI Reference documents functions, data types, and constants that are no longer supported. The reference is provided to help developers who already have code that uses these functions. Unless otherwise stated, see *Inside Mac OS X: ATSUI Reference* for descriptions of the result codes used by the functions in this reference.

Functions

ATSUCreateMemorySetting

Specifies how memory is allocated in your application.

Unsupported

```
OSStatus ATSUCreateMemorySetting (
    ATSUHeapSpec iHeapSpec,
    ATSUMemoryCallbacks *iMemoryCallbacks,
    ATSUMemorySetting *oMemorySetting
);
```

Parameters

iHeapSpec

A value that indicates whether ATSUI or your application controls memory allocation operations. If you pass the `kATSUUseSpecificHeap` constant, you must pass a pointer to a union that contains the correctly-prepared heap in the `heapToUse` field in the `iMemoryCallbacks` parameter. If you pass the `kATSUUseCallbacks` constant, you must pass a pointer to a `ATSUMemoryCallbacks` union that contains pointers to your application-defined functions in the `iMemoryCallbacks` parameter. If you pass the `kATSUUseCurrentHeap` or `kATSUUseAppHeap` constant, you should pass a `NULL` pointer in the `iMemoryCallbacks` parameter.

iMemoryCallbacks

A pointer to a memory callback union that contains either pointers to your memory allocation callback functions or the heap that you want ATSUI to use when allocating memory.

oMemorySetting

The `ATSUMemorySetting` type is used to store the results from a `ATSUGetCurrentMemorySetting` call. It can also be used to change the current memory setting by passing it into the `ATSUSetCurrentMemorySetting` call. On return, a pointer to a reference to a new memory allocation setting. To make this setting current, you must pass it to the function `ATSUSetCurrentMemorySetting` (page 7).

Return Value
A result code.

Discussion
The `ATSUCreateMemorySetting` function enables you to specify whether you wish to perform memory allocations yourself or have ATSUI do so. If you want to control memory allocation in ATSUI, pass `kATSUUseCallbacks` in the `iHeapSpec` parameter and a pointer to an `ATSUMemoryCallbacks` union that contains pointers to your callback functions in the `iMemoryCallbacks` parameter.

After creating a memory setting, you must pass it to the function `ATSUSetCurrentMemorySetting` (page 7) to ensure that it is used in subsequent Memory Manager calls.

You might want to create different memory settings for different memory allocation operations. For example, you might create two different settings designating different heaps to use for allocating the memory associated with style and text layout object creation. Before creating a style or text layout object, you would then make the appropriate setting current by calling `ATSUSetCurrentMemorySetting`.

Availability
Not supported in Carbon. Not available in CarbonLib. Not available in Mac OS X.

Carbon Porting Notes

Control of memory allocation is not available in CarbonLib and is not necessary in Mac OS X.

ATSUDisposeMemorySetting

Disposes of the current memory setting.

Unsupported

```
OSStatus ATSUDisposeMemorySetting (
    ATSUMemorySetting iMemorySetting
);
```

Parameters

iMemorySetting

A reference to the memory setting to dispose. If you dispose of the current memory setting, ATSUI uses the current heap and its own callback functions to perform memory allocation operations.

Return Value
A result code.

Discussion
Before you dispose of a memory setting, you should dispose of the memory associated with style and text layout objects that were allocated using that memory setting. For example, if you want to dispose of a memory setting that uses your application-defined callback functions to allocate memory, you should dispose of any memory that ATSUI allocated as a result of these callbacks before disposing of the setting.

Availability
Not supported in Carbon. Not available in CarbonLib. Not available in Mac OS X.

Carbon Porting Notes

Control of memory allocation not available in CarbonLib and not necessary in Mac OS X.

ATSUGetCurrentMemorySetting

Obtains the current memory setting.

Unsupported

```
ATSUMemorySetting ATSUGetCurrentMemorySetting ();
```

Return Value
A reference to the current memory allocation setting. If there is no current memory setting, `ATSUGetCurrentMemorySetting` returns `NULL`. See the description of the `ATSUMemorySetting` data type.

Availability
Not supported in Carbon. Not available in CarbonLib. Not available in Mac OS X.

Carbon Porting Notes

Control of memory allocation is not available in CarbonLib and is not necessary in Mac OS X.

ATSUSetCurrentMemorySetting

Sets the current memory setting to be used in Memory Manager calls.

Unsupported

```
OSStatus ATSUSetCurrentMemorySetting (
    ATSUMemorySetting iMemorySetting
);
```

Parameters

iMemorySetting

A reference to the memory setting to make current. This setting is used in subsequent Memory Manager calls made within ATSUI until you call `ATSUSetCurrentMemorySetting` with a new setting.

Return Value
A result code.

Discussion
After you create a memory setting by calling the function `ATSUCreateMemorySetting` (page 5), you must pass it to the `ATSUSetCurrentMemorySetting` function to ensure that it is used in subsequent Memory Manager calls.

Availability
Not supported in Carbon. Not available in CarbonLib. Not available in Mac OS X.

Carbon Porting Notes

Control of memory allocation is not available in CarbonLib and is not necessary in Mac OS X.

Callbacks

ATSUCustomAllocFunc

Defines a pointer to a callback function that handles memory allocation. Your callback function manages memory allocation operations typically handled by ATSUI.

```
typedef void(*ATSUCustomAllocFunc) (
    void *refCon,
    ByteCount howMuch
);
```

If you name your function `MyATSUCustomAllocCallback`, you would declare it like this:

```
void *MyATSUCustomAllocCallback (
    void *refCon,
    ByteCount howMuch
);
```

Parameters

refCon

On input, ATSUI passes your `MyATSUCustomAllocCallback` function a pointer to arbitrary data previously supplied by your application in the `memoryRefCon` field of the `ATSUMemoryCallbacks` (page 10) union.

howMuch

On input, ATSUI passes the amount of memory (in bytes) that you need to allocate.

Return Value

Your callback function returns an untyped pointer to the beginning of the block of memory allocated by your callback function.

Discussion

You can register your callback function by calling the function `ATSUCreateMemorySetting` (page 5) and passing the constant `kATSUUseCallbacks` in `iHeapSpec` and a pointer to the `ATSUMemoryCallbacks` (page 10) union in `iMemoryCallbacks`. You then supply a pointer of type `ATSUCustomAllocFunc` in the `Alloc` field of the `callbacks` structure of the `ATSUMemoryCallbacks` union.

Note that your `MyATSUCustomAllocCallback` function is expected to return a pointer to the start of the allocated memory, unless it terminates in an application.

Carbon Porting Notes

Control of memory allocation is not available in CarbonLib and is not necessary in Mac OS X.

ATSUCustomFreeFunc

Defines a pointer to a callback function that handles memory deallocation. Your callback function manages memory deallocation operations typically handled by ATSUI.

```
typedef void (*ATSUCustomFreeFunc) (
    void *refCon,
    void *doomedBlock
);
```

If you name your function `MyATSUCustomFreeCallback`, you would declare it like this:

```
void MyATSUCustomFreeCallback (
    void *refCon,
    void *doomedBlock
);
```

Parameters

refCon

On input, ATSUI passes your `MyATSUCustomFreeCallback` function a pointer to arbitrary data previously supplied by your application in the `memoryRefCon` field of the `ATSUMemoryCallbacks` (page 10) union.

doomedBlock

On input, ATSUI passes your `MyATSUCustomFreeCallback` function a pointer to the beginning of the block of memory to deallocate.

Return Value

Your callback function returns the address of the block of deallocated memory.

Discussion

You can register your callback function by calling the function `ATSUCreateMemorySetting` (page 5) and passing the constant `kATSUUseCallbacks` in `iHeapSpec` and a pointer to the `ATSUMemoryCallbacks` (page 10) union in `iMemoryCallbacks`. You then supply a pointer of type `ATSUCustomFreeFunc` in the `Free` field of the `callbacks` structure of the `ATSUMemoryCallbacks` union.

Carbon Porting Notes

Control of memory allocation is not available in CarbonLib and is not necessary in Mac OS X.

ATSUCustomGrowFunc

Defines a pointer to a callback function that handles memory reallocation. Your callback function manages memory reallocation operations typically handled by ATSUI.

```
typedef void(*ATSUCustomGrowFunc) (
    void *refCon,
    void *oldBlock,
    ByteCount oldSize,
    ByteCount newSize
);
```

If you name your function `MyATSUCustromGrowCallback`, you would declare it like this:

```
void *MyATSUCustromGrowCallback (
    void *refCon,
    void *oldBlock,
    ByteCount oldSize,
    ByteCount newSize
);
```

Parameters

refCon

On input, ATSUI passes your `MyATSUCustomGrowCallback` function a pointer to arbitrary data previously supplied by your application in the `memoryRefCon` field of the `ATSUMemoryCallbacks` (page 10) union.

oldBlock

On input, ATSUI passes your `MyATSUCustomFreeCallback` function a pointer to the beginning of the block of memory to grow. ATSUI passes this value to your application.

oldSize

ATSUI passes your `MyATSUCustomFreeCallback` function the size (in bytes) of the memory prior to growing. Your callback function can use this to determine the number of bytes of memory to copy if you need to allocate memory for the grown block.

newSize

ATSUI passes your `MyATSUCustomFreeCallback` function the size (in bytes) of the memory after growing.

Return Value

Your callback function returns an untyped pointer to the beginning address of the reallocated block of memory.

Discussion

You can register your callback function by calling the function `ATSUCreateMemorySetting` (page 5) and passing the constant `kATSUUseCallbacks` in `iHeapSpec` and a pointer to the `ATSUMemoryCallbacks` (page 10) union in `iMemoryCallbacks`. You then supply a pointer of type `ATSUCustomGrowFunc` in the `Grow` field of the `callbacks` structure of the `ATSUMemoryCallbacks` union.

Note that your `MyATSUCustomGrowCallback` function is expected to return a pointer to the start of the allocated memory, unless it terminates in an application.

Carbon Porting Notes

Control of memory allocation is not available in CarbonLib and is not necessary in Mac OS X.

Data Types

ATSUMemoryCallbacks

A union that contains either pointers to your application-defined memory allocation functions or the heap that you want ATSUI to use when allocating memory.

```
union ATSUMemoryCallbacks {
    union {
           Alloc;
       Free;
       Grow;
       *memoryRefCon;
    } callbacks;
    THz heapToUse;
};
```

Fields

`callbacks`

A pointer of type `ATSUCustomAllocFunc` (page 8) to your memory allocation callback function.

`heapToUse`

A pointer of type `ATSUCustomFreeFunc` (page 8) to your memory deallocation callback function.

Discussion

Note that control of memory allocation is not available in CarbonLib and is not necessary in Mac OS X.

`ATSUMemoryCallbacks` is a union struct that allows the ATSUI client to specify a specific heap for ATSUI use or allocation callbacks of which ATSUI is to use each time ATSUI performs a memory operation (alloc, grow, free).

If you want to control memory allocation in ATSUI, you should supply pointers to your memory allocation callback functions in the `callbacks` structure field of the union. If you want ATSUI to control memory allocation, you should supply the memory heap for ATSUI to use in the `heapToUse` field.

The `ATSUMemoryCallbacks` union is passed back by the function `ATSUCreateMemorySetting` (page 5) to represent the newly-created memory setting.

ATSUMemorySetting

Represents a reference to a private structure containing information about the current memory setting.

```
typedef struct OpaqueATSUMemorySetting* ATSUMemorySetting;
```

Discussion

Note that control of memory allocation is not available in CarbonLib and is not necessary in Mac OS X.

`ATSUMemorySetting` is used to store the results from a `ATSUSetMemoryAlloc` or a `ATSUGetCurrentMemorySetting` call. It can also be used to change the current `ATSUMemorySetting` by passing it into the `ATSUSetCurrentMemorySetting` call.

You pass a reference of type `ATSUMemorySetting` reference to the functions `ATSUDisposeMemorySetting` (page 6), `ATSUDisposeMemorySetting` (page 6) and to either dispose of a memory setting or make one current. The function `ATSUGetCurrentMemorySetting` (page 7) passes back a reference of this type to indicate the current memory setting.

Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared In

`ATSUnicodeTypes.h`

ATSUHeapSpec

Represents a preference for which heap to use.

```
typedef UInt16 ATSUHeapSpec;
```

Discussion
Note that control of memory allocation is not available in CarbonLib and is not necessary in Mac OS X.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared In
`ATSUnicodeTypes.h`

Constants

Heap Usage Options

Specify the heap from which ATSUI should allocate its dynamic memory.

```
typedef UInt16 ATSUHeapSpec;
enum {
    kATSUUseCurrentHeap = 0,
    kATSUUseAppHeap = 1,
    kATSUUseSpecificHeap = 2,
    kATSUUseCallbacks = 3
};
```

Constants

`kATSUUseCurrentHeap`
Specifies to use the current heap.

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in `ATSUnicodeTypes.h`.

`kATSUUseAppHeap`
Specifies to use the application heap.

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in `ATSUnicodeTypes.h`.

`kATSUUseSpecificHeap`
Specifies that ATSUI should perform memory allocation operations. When you use this constant you must also supply the correctly prepared heap in the `heapToUse` field of the `ATSUMemoryCallbacks` union.

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in `ATSUnicodeTypes.h`.

`kATSUUseCallbacks`

Specifies that ATSUI should use your own application-defined functions to control memory allocation. When you use this constant you must supply pointers to your application in the `callback` structure of the `ATSUMemoryCallbacks` union.

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in `ATSUnicodeTypes.h`.

Discussion
The functions that use Heap Usage Options are no longer supported.

Version Notes
Control of memory allocation is not available in CarbonLib and is not necessary in Mac OS X.

Document Revision History

This table describes the changes to *Legacy ATSUI Reference*.

Date	Notes
2006-07-24	Moved deprecated functions to ATSUI Reference.
2002-09-10	First release of this document. Released in conjunction with the *Inside Mac OS X: ATSUI Reference* that is updated for ATSUI version 2.4.
	Legacy ATSUI Reference contains deprecated functions, callbacks, and data types, whereas *Inside Mac OS X: ATSUI Reference* contains documentation for supported functions, callbacks, and data types.

Document Revision History

Index