Apple Type Services for Fonts Reference

For ATS for Fonts version 1.4
Carbon > Text & Fonts

2007-12-11

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

Simultaneously published in the United States and Canada.

Contents

Tables

Apple Type Services for Fonts Reference 9

Apple Type Services for Fonts Reference

Framework:	ApplicationServices/ApplicationServices.h
Declared in	ATSFont.h ATSTypes.h

Overview

Apple Type Services for Fonts is a collection of functions and data types that you can use to access and manage font data in Mac OS X. It is designed to handle a wide range of font technologies and data formats. The programming interface is designed with performance, scalability, and consistency in mind, and is available to Cocoa and Carbon applications through the Apple Type Services (ATS) and QuickDraw frameworks in Mac OS X.

Carbon supports the Apple Type Services for Fonts.

Functions by Task

Activating and Deactivating Fonts

`ATSFontActivateFromFileReference` (page 13)
Activates one or more fonts from a file reference.

`ATSFontActivateFromMemory` (page 14)
Activates one or more fonts at the specified location in memory.

`ATSFontDeactivate` (page 16)
Deactivates one or more fonts.

`ATSGetGeneration` (page 43)
Obtains the generation of the font database.

`ATSFontSetGlobalAutoActivationSetting` (page 43)
Sets the user's global auto-activation setting.

`ATSFontGetGlobalAutoActivationSetting` (page 30)
Gets the user's global auto-activation setting.

`ATSFontSetAutoActivationSettingForApplication` (page 42)
Sets the auto-activation setting for the specified application bundle.

`ATSFontGetAutoActivationSettingForApplication` (page 26)
Gets the activation setting for the specified application.

`ATSFontActivateFromFileSpecification` (page 83) Deprecated in Mac OS X v10.5
Activates one or more fonts from a file specification. (Deprecated. Instead use `ATSFontActivateFromFileReference` (page 13).)

Working With Font Families

`ATSFontFamilyApplyFunction` (page 17)
Applies your callback to a font family iteration.

`ATSFontFamilyIteratorCreate` (page 20)
Creates a font family iterator that your application can use to access font family objects.

`ATSFontFamilyIteratorRelease` (page 23)
Releases the memory associated with a font family iterator.

`ATSFontFamilyIteratorReset` (page 23)
Resets a font family iterator to the beginning of the iteration.

`ATSFontFamilyIteratorNext` (page 22)
Obtains the next font family reference.

`ATSFontFamilyFindFromName` (page 17)
Returns the font family reference associated with a font family name.

`ATSFontFamilyFindFromQuickDrawName` (page 18)
Returns the font family reference associated with a standard QuickDraw font name.

`ATSFontFamilyGetGeneration` (page 19)
Returns the generation count of a font family.

`ATSFontFamilyGetName` (page 19)
Obtains the font family name associated with a font family reference.

`ATSFontFamilyGetQuickDrawName` (page 20)
Obtains the standard QuickDraw font name associated with a font family reference.

`ATSFontFamilyGetEncoding` (page 18)
Returns the text encoding used by a font family.

Working With Fonts

`ATSFontApplyFunction` (page 15)
Applies your callback to a font iteration.

`ATSFontIteratorCreate` (page 35)
Creates a font iterator.

`ATSFontIteratorRelease` (page 38)
Releases a font iterator.

`ATSFontIteratorReset` (page 38)
Resets a font iterator to the beginning of the iteration.

`ATSFontIteratorNext` (page 37)
Obtains the next font reference.

`ATSFontFindFromName` (page 25)
Returns the font reference associated with a font name.

`ATSFontFindFromPostScriptName` (page 26)
Returns the font reference associated with a PostScript font name.

`ATSFontFindFromContainer` (page 24)
Obtains the font references contained in a font container.

`ATSFontGetGeneration` (page 29)
Returns the generation count for a font.

`ATSFontGetContainerFromFileReference` (page 27)
Gets the font container reference associated with an activated file reference.

`ATSFontGetContainer` (page 27)
Gets the font container reference for a font.

`ATSFontGetName` (page 31)
Obtains the name of a font associated with a font reference.

`ATSFontGetPostScriptName` (page 31)
Obtains the PostScript name from a font reference.

`ATSFontGetTableDirectory` (page 33)
Obtains the table directory for a font.

`ATSFontGetTable` (page 32)
Obtains a font table.

`ATSFontGetHorizontalMetrics` (page 30)
Obtains the horizontal metrics for a font.

`ATSFontGetVerticalMetrics` (page 34)
Obtains the vertical metrics for a font.

`ATSFontGetFileReference` (page 28)
Obtains the file reference for a font.

`ATSFontGetFontFamilyResource` (page 28)
Obtains the font family resource for a font.

`ATSFontSetEnabled` (page 42)
Sets a font state to enabled or disabled.

`ATSFontIsEnabled` (page 35)
Returns `true` if the font is enabled.

`ATSFontGetFileSpecification` (page 84) Deprecated in Mac OS X v10.5
Obtains the file specification for a font. (Deprecated. Instead use `ATSFontGetFileReference` (page 28).)

Setting Up Notifications and Queries

`ATSFontNotify` (page 41)
Notifies Apple Type Services of an action taken by your application.

`ATSFontNotificationSubscribe` (page 39)
Signs up your application to receive notification of changes to fonts and font directories.

`ATSFontNotificationUnsubscribe` (page 40)
Unsubscribes your application from receiving notifications of changes to fonts and font directories.

`ATSCreateFontQueryRunLoopSource` (page 12)
Sets up your application to handle font queries.

Creating, Calling, and Deleting Universal Procedure Pointers

`NewFMFontCallbackFilterUPP` (page 45)
Creates a new universal procedure pointer (UPP) to a filter callback function that uses your criteria for filtering fonts.

`DisposeFMFontCallbackFilterUPP` (page 43)
Disposes of a universal procedure pointer to a customized filter function used for fonts.

`InvokeFMFontCallbackFilterUPP` (page 44)
Calls a customized filter function used for fonts.

`NewFMFontFamilyCallbackFilterUPP` (page 45)
Creates a new universal procedure pointer (UPP) to a filter callback function that uses your criteria for filtering font families.

`DisposeFMFontFamilyCallbackFilterUPP` (page 44)
Disposes of a universal procedure pointer to a customized filter function used for font families.

`InvokeFMFontFamilyCallbackFilterUPP` (page 45)
Calls a customized filter function used for font families.

Functions

ATSCreateFontQueryRunLoopSource

Sets up your application to handle font queries.

```
CFRunLoopSourceRef ATSCreateFontQueryRunLoopSource (
   CFIndex queryOrder,
   CFIndex sourceOrder,
   ATSFontQueryCallback callout,
   const ATSFontQuerySourceContext *context
);
```

Parameters

queryOrder
A `CFIndex` value that specifies the priority of the query relative to other queries. ATS sends font queries to each run loop in priority order, from highest to lowest, with normal priority equal to `0`.

sourceOrder
A `CFIndex` value that specifies the order of the run loop source.

callout
A pointer to your callback for processing a font query. See `ATSFontQueryCallback` (page 48) for more information on the callback you can supply.

context
A pointer to font query source context that ATS passes to your callback function. You can pass `NULL` if your callback function does not need any data passed to it.

Return Value
A `CFRunLoopSourceRef`. When you want to stop receiving queries, you must invalidate this reference.

Discussion
When an application needs a font, ATS sends a query to those font utility applications who have signed up to handle queries by calling the function `ATSCreateFontQueryRunLoopSource`. When a font utility application receives a query, it iterates through its available fonts to look for the requested font. If the font utility application finds the font, it obtains the file specification for the font and sends the file specification to ATS. ATS activates the font and sends notification of the activation to each application who subscribes to notifications.

The function `ATSCreateFontQueryRunLoopSource` creates a Core Foundation run loop source reference (`CFRunLoopSourceRef`) to convey font queries from ATS to your font utility application. If your application does not have a `CFRunLoop` (for example, a faceless server application), you must explicitly set up a `CFRunLoop` before you can receive queries.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.2 and later.

Declared In
`ATSFont.h`

ATSFontActivateFromFileReference

Activates one or more fonts from a file reference.

```
OSStatus ATSFontActivateFromFileReference (
   const FSRef *iFile,
   ATSFontContext iContext,
   ATSFontFormat iFormat,
   void *iRefCon,
   ATSOptionFlags iOptions,
   ATSFontContainerRef *oContainer
);
```

Parameters

iFile

A pointer to the file reference that specifies the name and location of a file or directory that contains the font data you want to activate.

iContext

A value that specifies the context of the activated font. If you want the activated font to be accessible only from your application use the `kATSFontContextLocal` constant. If you want the activated font to be accessible to all applications use the constant `kATSFontContextGlobal`. See "Context Options" (page 68) for more information.

iFormat

A value that represents the format identifier of the font. Pass `kATSFontFormatUnspecified` as the system automatically determines the format of the font. For more information on this constant, see "Font Formats" (page 71).

iRefCon

This parameter is currently reserved for future use, so you should pass `NULL`.

iOptions

An options flag. Pass `kATSOptionFlagsDefault` unless the font's data fork contains resource-fork information, you need to activate a directory of font directories, or you plan to call this function a number of time. If the font's data fork contains resource-fork information, pass the option `kATSOptionFlagsUseDataForkAsResourceFork`. If the you want to activate a font directory that contains font directories, you must pass the option `kATSOptionFlagsProcessSubdirectories`. If you plan to call this function a number of times, you can set the `iOptions` parameter to `kATSOptionFlagsDoNotNotify` set. When you are done activating fonts you can call the function `ATSFontNotify` (page 41) with the `action` parameter set to `kATSFontNotifyActionFontsChanged`. Then ATS notifies all applications who subscribe to notifications of the changes you made.

oContainer

On output, a reference to the font container that is activated from the file reference. You need this reference when you deactivate the font by calling the function `ATSFontDeactivate` (page 16).

Return Value

If activated successfully, `noErr`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ATSFont.h`

ATSFontActivateFromMemory

Activates one or more fonts at the specified location in memory.

```
OSStatus ATSFontActivateFromMemory (
   LogicalAddress iData,
   ByteCount iLength,
   ATSFontContext iContext,
   ATSFontFormat iFormat,
   void *iReserved,
   ATSOptionFlags iOptions,
   ATSFontContainerRef *oContainer
);
```

Parameters

iData

The logical address of the font you want to activate.

iLength

The length (in bytes) of the font data.

iContext

A value that specifies the context of the activated font. If you want the activated font to be accessible only from your application use the `kATSFontContextLocal` constant. If you want the activated font to be accessible to all applications use the constant `kATSFontContextGlobal`. See "Context Options" (page 68) for more information.

iFormat

A value that represents the format identifier of the font. There is only one font format constant available for you to pass—`kATSFontFormatUnspecified`. This constant specifies the default behavior, which is to handle the data as raw TrueType font data. This is equivalent to the contents of an `'sfnt'` font resource or the data fork of a Windows TrueType `.ttf` or `.ttc` file. You can also activate the contents of an OpenType TrueType `.OTF` file. See "Font Formats" (page 71) for more information.

iReserved

An arbitrary 32-bit value. This parameter is currently reserved for future use, so you should pass `NULL`.

iOptions

An `ATSOptionFlags` value. This parameter is currently reserved for future use, so you should pass `kATSOptionFlagsDefault`.

oContainer

On output, a pointer to a font container reference that refers to the file that contains the activated font data.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

You use this function to activate a streamed font, such as a font contained in a PDF file. Your application must first map the streamed font data to memory and then pass the address of the font data in memory to the function `ATSFontActivateFromMemory`.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In

`ATSFont.h`

ATSFontApplyFunction

Applies your callback to a font iteration.

```
OSStatus ATSFontApplyFunction (
   ATSFontApplierFunction iFunction,
   void *iRefCon
);
```

Parameters

iFunction

The callback function you want applied to a font iteration. See `ATSFontApplierFunction` (page 46) for more information on the callback you need to supply.

iRefCon

An arbitrary 32-bit value specified by your application. This is passed to your callback.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

The function `ATSFontApplyFunction` iterates through the default fonts, which include globally activated fonts and fonts activated locally to your application. Calling this function is similar to creating an iterator that operates on a local context with an unrestricted scope.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
ATSFont.h

ATSFontDeactivate

Deactivates one or more fonts.

```
OSStatus ATSFontDeactivate (
   ATSFontContainerRef iContainer,
   void *iRefCon,
   ATSOptionFlags iOptions
);
```

Parameters

iContainer

A font container reference that refers to the file containing the activated font data. You obtain a font container reference when you activate a font by calling the functions ATSFontActivateFromFileSpecification (page 83) or ATSFontActivateFromMemory (page 14).

iRefCon

An arbitrary 32-bit value specified. This parameter is currently reserved for future use, so you should pass NULL.

iOptions

An ATSOptionFlags value. You should pass kATSOptionFlagsDefault unless to plan to call this function a number of times to deactivate many fonts. If you plan to call this function a number of times, you can set the iOptions parameter to kATSOptionFlagsDoNotNotify set. When you are done deactivating fonts you can call the function ATSFontNotify (page 41) with the action parameter set to kATSFontNotifyActionFontsChanged. ATS notifies all applications who subscribe to notifications of the changes you made.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
When you deactivate a font, you must supply the font container reference you obtained when you activated the font. You can't deactivate a font that you did not activate by calling the functions ATSFontActivateFromFileSpecification (page 83) or ATSFontActivateFromMemory (page 14).

You should use caution if you deactivate a font that is available globally, as its deactivation impacts any application that uses that font.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
ATSFont.h

ATSFontFamilyApplyFunction

Applies your callback to a font family iteration.

```
OSStatus ATSFontFamilyApplyFunction (
   ATSFontFamilyApplierFunction iFunction,
   void *iRefCon
);
```

Parameters

iFunction

The callback function you want applied to a font family iteration. See `ATSFontApplierFunction` (page 46) for more information on the callback you need to supply.

iRefCon

An arbitrary 32-bit value specified by your application. This value is passed to your callback.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

The function `ATSFontFamilyApplyFunction` iterates through the default font families, which include globally activated font families and font families activated locally to your application. Calling this function is similar to creating an iterator that operates on a local context with an unrestricted scope.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In

`ATSFont.h`

ATSFontFamilyFindFromName

Returns the font family reference associated with a font family name.

```
ATSFontFamilyRef ATSFontFamilyFindFromName (
   CFStringRef iName,
   ATSOptionFlags iOptions
);
```

Parameters

iName

A reference to a font family name, formatted as a `CFString`.

iOptions

An `ATSOptionFlags` value. This parameter is currently reserved for future use, so you should pass `kATSOptionFlagsDefault`.

Return Value

A reference to the font family specified by the `iName` parameter. See the description of the `ATSFontFamilyRef` data type.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In
ATSFont.h

ATSFontFamilyFindFromQuickDrawName

Returns the font family reference associated with a standard QuickDraw font name.

```
ATSFontFamilyRef ATSFontFamilyFindFromQuickDrawName (
   ConstStr255Param iName
);
```

Parameters

iName
A QuickDraw font name.

Return Value
A reference to the font family associated with the font name specified by the `iName` parameter. See the description of the `ATSFontFamilyRef` data type.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
ATSFont.h

ATSFontFamilyGetEncoding

Returns the text encoding used by a font family.

```
TextEncoding ATSFontFamilyGetEncoding (
   ATSFontFamilyRef iFamily
);
```

Parameters

iFamily
A font family reference.

Return Value
On output, a pointer to the text encoding used by the font family associated with the font family reference. See the Text Encoding Conversion Manager documentation for a description of the `TextEncoding` data type.

Discussion
Once you have obtained the text encoding, you can use Text Encoding Converter Manager function `RevertTextEncodingToScriptInfo` to extract the script as follows:

```
status = ATSFontFamilyGetEncoding (myFontFamily, &myTextEncoding)

status = RevertTextEncodingToScriptInfo (myTextEncoding, &myScriptCode);
```

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontFamilyGetGeneration

Returns the generation count of a font family.

```
ATSGeneration ATSFontFamilyGetGeneration (
   ATSFontFamilyRef iFamily
);
```

Parameters

iFamily
A font family reference.

Return Value
On output, the generation count for the font family associated with the font family reference. See the description of the `ATSGeneration` data type.

Discussion
The generation of a font family changes any time part of a font family is removed or added.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontFamilyGetName

Obtains the font family name associated with a font family reference.

```
OSStatus ATSFontFamilyGetName (
   ATSFontFamilyRef iFamily,
   ATSOptionFlags iOptions,
   CFStringRef *oName
);
```

Parameters

iFamily
A font family reference.

iOptions
An `ATSOptionFlags` value. This parameter is currently reserved for future use, so you should pass `kATSOptionFlagsDefault`.

oName
On output, a reference to the name associated with the font family reference, formatted as a `CFString`. You are responsible for releasing the `CFStringRef`.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontFamilyGetQuickDrawName

Obtains the standard QuickDraw font name associated with a font family reference.

```
OSStatus ATSFontFamilyGetQuickDrawName (
   ATSFontFamilyRef iFamily,
   Str255 oName
);
```

Parameters

iName

A reference to the font family name whose QuickDraw name you want to obtain.

oName

On input, a `Str255` value allocated by your application. On output, the QuickDraw name associated with the font family reference.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
All font families are assigned a QuickDraw name by the system. The QuickDraw name is almost identical to the font family name.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontFamilyIteratorCreate

Creates a font family iterator that your application can use to access font family objects.

```
OSStatus ATSFontFamilyIteratorCreate (
   ATSFontContext iContext,
   const ATSFontFilter *iFilter,
   void *iRefCon,
   ATSOptionFlags iOptions,
   ATSFontFamilyIterator *ioIterator
);
```

Parameters

iContext

A value that specifies the context of the iterator. If you want to apply the font family iterator only to the fonts accessible from your application use the `kATSFontContextLocal` constant. If you want the to apply the font family iterator to all fonts registered with the system use the constant `kATSFontContextGlobal`. See "Context Options" (page 68) for more information on the constants you can supply. See the Discussion for information on the interaction between the `iContext` and `iOptions` parameters.

iFilter

A pointer to a filter specification. Pass `NULL` if you do not want to apply a filter to this iteration. Otherwise, you can use this parameter to restrict the iteration to the font families that match a generation count or criteria you specify in a custom filter function. Pass the filter selector constant `kATSFontFilterSelectorGeneration` to select a generation filter or the constant `kATSFontFilterSelectorFontApplierFunction` to select a custom filter. See "Font Filter Selectors" (page 70) for more information on the constants you can supply.

iRefCon

An arbitrary 32-bit value specified by your application. If you are using a custom filter function, you can use this parameter to pass data to the custom filter function. Otherwise, pass `NULL`.

iOptions

A value that specifies the scope of the iterator. If you want to iterate through font families that can be used only by your application, pass the constant `kATSOptionFlagsRestrictedScope`. If you want to iterate through font families that can be used by all applications pass the constant `kATSOptionFlagsUnRestrictedScope`. See "Scoping Options" (page 75) for more information on the constants you can supply. See the Discussion for information on the interaction between the `iContext` and `iOptions` parameters.

ioIterator

A pointer to a font family iterator. On output, points to an opaque font family iterator ready for you to use. When you no longer need the font family iterator, you should call the function `ATSFontFamilyIteratorRelease` (page 23) to release the auxiliary data and memory allocated by the system.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

Your application can use a font family iterator to access font family objects. A font family iterator is an opaque data structure used by ATS for Fonts to keep track of an iteration over currently active font families. When the font family iterator is initialized, it does not yet reference a font family.

The context and scope you specify for the font family iterator interact as shown in Table 1 (page 22).

Table 1 The interaction of context and scope in a font family enumeration

	Local context	Global context
Restricted scope	Font families activated locally to your application	Only globally activated font families
Unrestricted scope	Defaults font families, which include globally activated font families and font families activated locally to your application	All font families, which include globally activated font families and all other font families activated locally for an application.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontFamilyIteratorNext

Obtains the next font family reference.

```
OSStatus ATSFontFamilyIteratorNext (
   ATSFontFamilyIterator iIterator,
   ATSFontFamilyRef *oFamily
);
```

Parameters

iIterator

A pointer to a font family iterator you created with the function `ATSFontFamilyIteratorCreate` (page 20).

oFamily

A pointer to a font family reference. On output, points to the font family reference obtained by the iterator. You are responsible for allocating memory for the font family reference.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
If any changes are made to the font database while you are using the font family iterator, the iterator is invalidated and the function `ATSFontFamilyIteratorNext` returns the error `kATSIterationScopeModified`. To remedy this error, your application must either restart or cancel the enumeration by calling the `ATSFontFamilyIteratorReset` (page 23) or the `ATSFontFamilyIteratorRelease` (page 23) functions.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontFamilyIteratorRelease

Releases the memory associated with a font family iterator.

```
OSStatus ATSFontFamilyIteratorRelease (
   ATSFontFamilyIterator *ioIterator
);
```

Parameters

ioIterator

A pointer to a font family iterator you created with the function `ATSFontFamilyIteratorCreate` (page 20). If you try to use the font family iterator after disposing of its contents through this function, ATS for Fonts returns an error code to your application.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

If you plan to use the font family iterator again, you should consider calling the function `ATSFontFamilyIteratorReset` rather than releasing the font family iterator and then creating it again.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In

`ATSFont.h`

ATSFontFamilyIteratorReset

Resets a font family iterator to the beginning of the iteration.

```
OSStatus ATSFontFamilyIteratorReset (
   ATSFontContext iContext,
   const ATSFontFilter *iFilter,
   void *iRefCon,
   ATSOptionFlags iOptions,
   ATSFontFamilyIterator *ioIterator
);
```

Parameters

iContext

A value that specifies the context of the iterator. If you want to apply the font family iterator only to the fonts accessible from your application use the `kATSFontContextLocal` constant. If you want the to apply the font family iterator to all fonts registered with the system use the constant `kATSFontContextGlobal`. See "Context Options" (page 68) for more information.

iFilter

A pointer to a filter specification. Pass `NULL` if you do not want to apply a filter to this iteration. Otherwise, you can use this parameter to restrict the iteration to the font families that match a generation count or criteria you specify in a custom filter function. Pass the filter selector constant `kATSFontFilterSelectorGeneration` to select a generation filter or the constant `kATSFontFilterSelectorFontApplierFunction` to select a custom filter. See "Font Filter Selectors" (page 70) for more information on these constants.

iRefCon

An arbitrary 32-bit value specified by your application. If you are using a custom filter function, you can use this parameter to pass data to the custom filter function. If you are not using a custom filter function, pass `NULL`.

iOptions

An value that specifies the scope of the iterator. If you want to iterate through font families that can be used only by your application, pass the constant `kATSOptionFlagsRestrictedScope`. If you want to iterate through font families that can be used by all applications pass the constant `kATSOptionFlagsUnRestrictedScope`.

ioIterator

A pointer to a font family iterator you created with the function `ATSFontFamilyIteratorCreate` (page 20). On output, the font family iterator is reset to the beginning of the iteration.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

Once you have created a font family iterator, you can reuse it by calling the function `ATSFontFamilyIteratorReset`. This function sets the parameters to the new values you specify, and repositions the iterator so it is ready to get the first font family reference when you call the function `ATSFontFamilyIteratorNext` (page 22).

During an iteration, if you obtain the result code `kATSIterationScopeModified` from the function `ATSFontFamilyIteratorNext` (page 22), you can reset the iteration by calling the function `ATSFontFamilyIteratorReset`. This assures that you obtain the most up-to-date information from the iteration.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In

`ATSFont.h`

ATSFontFindFromContainer

Obtains the font references contained in a font container.

```
OSStatus ATSFontFindFromContainer (
   ATSFontContainerRef iContainer,
   ATSOptionFlags iOptions,
   ItemCount iCount,
   ATSFontRef ioArray[],
   ItemCount *oCount
);
```

Parameters

iContainer

A reference to the font container whose fonts you want to obtain. You obtain a font container reference when you activate a font by calling the functions `ATSFontActivateFromFileSpecification` (page 83) or `ATSFontActivateFromMemory` (page 14).

iOptions

An `ATSOptionFlags` value. This parameter is currently reserved for future use, so you should pass `kATSOptionFlagsDefault`.

iCount

The number of items in the `ioArray` array. If you are uncertain of how many items are in this array, see the Discussion.

ioArray

A pointer to memory you have allocated for an array of font references. On return, the array contains the font references in the font container specified by the `iContainer` parameter. If you are uncertain of how much memory to allocate for this array, see the Discussion.

oCount

A pointer to an `ItemCount` value. On output, the value specifies the actual number of `ATSFontRef` values in the font container.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
The function `ATSFontFindFromContainer` operates on font containers that reference font files. It does not work on font containers that reference font directories.

Typically you use the function `ATSFindFontFromContainer` by calling it twice, as follows:

1. Pass a reference to the font container to examine in the `iContainer` parameter, a valid pointer to an `ItemCount` value in the *oCount* parameter, `NULL` for the `ioArray` parameter, and 0 for the *iCount* parameter. `ATSFindFontFromContainer` returns the size of the array in the `oCount` parameter.

2. Allocate enough space for an array of the returned size, then call the `ATSFindFontFromContainer` function again, passing a valid pointer in the `ioArray` parameter and the number of items in the array in the *iCount* parameter. On return, the pointer refers to an array of the font references contained in the font container.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontFindFromName

Returns the font reference associated with a font name.

```
ATSFontRef ATSFontFindFromName (
   CFStringRef iName,
   ATSOptionFlags iOptions
);
```

Parameters

iName

A reference to a font name formatted as a `CFString`.

iOptions

An `ATSOptionFlags` value. This parameter is currently reserved for future use, so you should pass `kATSOptionFlagsDefault`.

Return Value
A reference to the font specified by the `iName` parameter. See the description of the `ATSFontRef` data type.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontFindFromPostScriptName

Returns the font reference associated with a PostScript font name.

```
ATSFontRef ATSFontFindFromPostScriptName (
   CFStringRef iName,
   ATSOptionFlags iOptions
);
```

Parameters

iName

A reference to the PostScript name for a font, formatted as a `CFString`.

iOptions

An `ATSOptionFlags` value. This parameter is currently reserved for future use, so you should pass `kATSOptionFlagsDefault`.

Return Value
A reference to the font specified by the `iName` parameter. See the description of the `ATSFontRef` data type.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontGetAutoActivationSettingForApplication

Gets the activation setting for the specified application.

```
ATSFontAutoActivationSetting ATSFontGetAutoActivationSettingForApplication (
   CFURLRef iApplicationFileURL
);
```

Parameters

iApplicationFileURL

A valid file URL for an application. Pass `NULL` to specify the current process.

Return Value
The activation setting for the specified application.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`ATSFont.h`

ATSFontGetContainer

Gets the font container reference for a font.

```
OSStatus ATSFontGetContainer (
   ATSFontRef iFont,
   ATSOptionFlags iOptions,
   ATSFontContainerRef *oContainer
);
```

Parameters

iFont

The font reference.

iOptions

An options flag.

oContainer

On output, a reference to the font container that was used to activate the font reference. On error ATS sets this to `kATSFontContainerRefUnspecified`.

Return Value
If successful, `noErr`; if the container is invalid, `kATSInvalidFontContainerAccess`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`ATSFont.h`

ATSFontGetContainerFromFileReference

Gets the font container reference associated with an activated file reference.

```
OSStatus ATSFontGetContainerFromFileReference (
   const FSRef *iFile,
   ATSFontContext iContext,
   ATSOptionFlags iOptions,
   ATSFontContainerRef *oContainer
);
```

Parameters

iFile

A pointer to the valid file reference that specifies the activated font file for which to get the container.

iContext

The context that the font file is accessible to. If you want the activated font to be accessible only from your application pass `kATSFontContextDefault` or `kATSFontContextLocal`. If you want the activated font to be accessible to all applications use the constant `kATSFontContextGlobal`. See "Context Options" (page 68) for more information.

iOptions

An options flag.

oContainer

On output, a reference to the font container representing the file reference activated in the specified context. On error or for a file that is not activated, ATS sets this to `kATSFontContainerRefUnspecified`.

Return Value

`noErr` or `paramErr` if one or more parameters are invalid.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ATSFont.h`

ATSFontGetFileReference

Obtains the file reference for a font.

```
OSStatus ATSFontGetFileReference (
   ATSFontRef iFont,
   FSRef *oFile
);
```

Parameters

iFont

A reference to the font whose file reference you want to obtain.

oFile

On output, points to the file reference that specifies the name and location of a file or directory that contains the font data specified by the `iFont` parameter.

Return Value

If successful, `noErr`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ATSFont.h`

ATSFontGetFontFamilyResource

Obtains the font family resource for a font.

```
OSStatus ATSFontGetFontFamilyResource (
   ATSFontRef iFont,
   ByteCount iBufferSize,
   void *ioBuffer,
   ByteCount *oSize
);
```

Parameters

iFont

A font reference.

iBufferSize

The size of the buffer pointed to by the `ioBuffer` parameter. See the Discussion if you are unsure of the size of this buffer.

ioBuffer

On input, a pointer to memory you allocated for the font family resource. On output, points to the `FOND` resource for the font. Note that the `FOND` resource data is in big endian format, regardless of the native endian format of the Macintosh computer on which you make the function call. If you are uncertain of how much memory to allocate for this array, see the Discussion.

oSize

On output, the actual size of the buffer.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

The function `ATSFontGetFontFamilyResource` provides a compatibility path for font families that use resources. Beginning with Mac OS X version 10.2, ATS for Fonts synthesizes `FOND` resources for OpenType fonts.

Typically you use the function `ATSFontGetFontFamilyResource` by calling it twice, as follows:

1. Pass a reference to the font to examine in the `iFont` parameter, a valid pointer in the *oSize* parameter, `NULL` for the `ioBuffer` parameter, and 0 for the *iBufferSize* parameter. `ATSFontGetFontFamilyResource` returns the size of the buffer in the `oSize` parameter.

2. Allocate enough space for an array of the returned size, then call the `ATSFontGetFontFamilyResource` function again, passing a valid pointer in the `ioBuffer` parameter, the size of the buffer in the *iBufferSize* parameter, and the appropriate values in the other parameters. On return, the pointer refers to an array of the font references contained in the font container.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In

`ATSFont.h`

ATSFontGetGeneration

Returns the generation count for a font.

```
ATSGeneration ATSFontGetGeneration (
   ATSFontRef iFont
);
```

Parameters

iFont

A font reference.

Return Value
A generation count. See the description of the `ATSGeneration` data type.

Discussion
ATS for Fonts increments the generation count for any changes to a font, including when the system synthesizes data for the font.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontGetGlobalAutoActivationSetting

Gets the user's global auto-activation setting.

```
ATSFontAutoActivationSetting ATSFontGetGlobalAutoActivationSetting (
   void
);
```

Return Value
The user's global auto-activation setting.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`ATSFont.h`

ATSFontGetHorizontalMetrics

Obtains the horizontal metrics for a font.

```
OSStatus ATSFontGetHorizontalMetrics (
   ATSFontRef iFont,
   ATSOptionFlags iOptions,
   ATSFontMetrics *oMetrics
);
```

Parameters

iFont

A reference to the font whose horizontal metrics you want to obtain.

iOptions

An options flag. This parameter is currently reserved for future use, so you should pass `kATSOptionFlagsDefault`.

oMetrics

On input, a valid pointer to an `ATSFontMetrics` (page 54) data structure. On output, the structure contains the font's horizontal metrics. If one or more measurements are not available for a font, then the appropriate fields in the `ATSFontMetrics` data structure are set to `0`.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontGetName

Obtains the name of a font associated with a font reference.

```
OSStatus ATSFontGetName (
   ATSFontRef iFont,
   ATSOptionFlags iOptions,
   CFStringRef *oName
);
```

Parameters

iFont

A font reference.

iOptions

An `ATSOptionFlags` value. This parameter is currently reserved for future use, so you should pass `kATSOptionFlagsDefault`.

oName

On output, a reference to the font name associated with the specified font reference, formatted as a `CFString`. You are responsible for releasing the `CFStringRef`.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontGetPostScriptName

Obtains the PostScript name from a font reference.

```
OSStatus ATSFontGetPostScriptName (
   ATSFontRef iFont,
   ATSOptionFlags iOptions,
   CFStringRef *oName
);
```

Parameters

iFont

A font reference.

iOptions

An `ATSOptionFlags` value. This parameter is currently reserved for future use, so you should pass `kATSOptionFlagsDefault`.

oName

On output, a reference to the PostScript name for the font, formatted as a `CFString`. You are responsible for releasing the `CFStringRef`.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

The system automatically detects whether or not the font is composed PostScript. If the font is, ATS for Fonts appends the CMAP name.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In

`ATSFont.h`

ATSFontGetTable

Obtains a font table.

```
OSStatus ATSFontGetTable (
   ATSFontRef iFont,
   FourCharCode iTag,
   ByteOffset iOffset,
   ByteCount iBufferSize,
   void *ioBuffer,
   ByteCount *oSize
);
```

Parameters

iFont

A reference to the font whose table you want to obtain.

iTag

A four-character code that specifies the font table you want to obtain.

iOffset

The offset to a font table. If you want to obtain all the font tables associated with a font, pass `0`.

iBufferSize

The size of the buffer pointed to by the ioBuffer parameter. The size should be the actual size of the buffer (oSize) minus the offset to the font table (iOffset) you want to obtain. See the Discussion if you are unsure of value to supply.

ioBuffer

On input, a valid pointer. On output, a pointer to the font table. Note that the data returned in the font table is in big endian format, regardless of the native endian format of the Macintosh computer on which you make the function call. See the Discussion for information on allocating this buffer.

oSize

On output, the actual size of the buffer returned in the ioBuffer parameter.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
Synthetic font tables (entries with offset of 0) can only be accessed by calling the function ATSFontGetTable.

Typically you use the function ATSFontGetTable by calling it twice, as follows:

1. Pass a reference to the font whose table you want obtain in the iFont parameter, a four-character code that specifies the font table you want to obtain in the iTag parameter, the appropriate offset to the font table in the *iOffset* parameter, 0 for the *iBufferSize* parameter, NULL for the ioBuffer parameter, and a valid pointer to a ByteCount value in the *oSize* parameter. ATSUFontGetTable returns the size of the table in the oSize parameter.

2. Allocate enough space for a buffer of the returned size, then call the ATSFontGetTable function again, passing a valid pointer in the ioBuffer parameter, the size of the buffer in the *iBufferSize* parameter, and the appropriate values in the other parameters. On return, the pointer refers to the table for the font specified by the iFont parameter and the table specified by the iTag parameter.

You should use the function ATSFontGetTable when you need to obtain an entire font table. For performance reasons, avoid using the function to check a single value in the table.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
ATSFont.h

ATSFontGetTableDirectory

Obtains the table directory for a font.

```
OSStatus ATSFontGetTableDirectory (
   ATSFontRef iFont,
   ByteCount iBufferSize,
   void *ioBuffer,
   ByteCount *oSize
);
```

Parameters

iFont

The font reference whose table directory you want to obtain.

iBufferSize

The size of the buffer pointed to by the `ioBuffer` parameter. See the Discussion if you are unsure of the size of this buffer.

ioBuffer

On input, a valid pointer. On output, points to the table directory for the font specified by the `iFont` parameter. Note that the data returned in the table directory is in big endian format, regardless of the native endian format of the Macintosh computer on which you make the function call. See the Discussion for information on allocating this buffer.

oSize

On output, the actual size of the buffer returned in the `ioBuffer` parameter.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

If necessary, ATS for Fonts synthesizes font tables or data, replacing existing tables or data. ATS for Fonts synthesizes data on an as needed basis; if data is synthesized, the generation count of the font increases.

Typically you use the function `ATSFontGetTableDirectory` by calling it twice, as follows:

1. Pass a reference to the font whose table directory you want obtain in the `iFont` parameter, `0` for the *iBufferSize* parameter, `NULL` for the `ioBuffer` parameter, and a valid pointer to a `ByteCount` value in the *oSize* parameter. `ATSFontGetTableDirectory` returns the size of the table directory in the `oSize` parameter.

2. Allocate enough space for a buffer of the returned size, then call the `ATSFontGetTableDirectory` function again, passing a valid pointer in the `ioBuffer` parameter, and the size of the buffer in the *iBufferSize* parameter. On return, the pointer refers to the table directory for the font specified by the `iFont` parameter.

If you want to obtain a font table, call the function `ATSFontGetTable` (page 32).

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In

`ATSFont.h`

ATSFontGetVerticalMetrics

Obtains the vertical metrics for a font.

```
OSStatus ATSFontGetVerticalMetrics (
   ATSFontRef iFont,
   ATSOptionFlags iOptions,
   ATSFontMetrics *oMetrics
);
```

Parameters

iFont

A reference to the font whose vertical metrics you want to obtain.

iOptions

An options flag. This parameter is currently reserved for future use, so you should pass `kATSOptionFlagsDefault`.

oMetrics

On input, a valid pointer to an `ATSFontMetrics` (page 54) data structure. On output, the structure contains the font's vertical metrics. If one or more measurements are not available for a font, then the appropriate fields in the `ATSFontMetrics` data structure are set to `0`.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In

`ATSFont.h`

ATSFontIsEnabled

Returns `true` if the font is enabled.

```
Boolean ATSFontIsEnabled (
   ATSFontRef iFont
);
```

Parameters

iFont

The font reference.

Return Value

`true` if the font is enabled.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ATSFont.h`

ATSFontIteratorCreate

Creates a font iterator.

```
OSStatus ATSFontIteratorCreate (
   ATSFontContext iContext,
   const ATSFontFilter *iFilter,
   void *iRefCon,
   ATSOptionFlags iOptions,
   ATSFontIterator *ioIterator
);
```

Parameters

iContext

A value that specifies the context of the iterator. If you want to apply the font iterator only to the fonts accessible from your application use the `kATSFontContextLocal` constant. If you want the to apply the font iterator to all fonts registered with the system use the constant `kATSFontContextGlobal`. See "Context Options" (page 68) for more information on the constants you can supply. See the Discussion for information on the interaction between the `iContext` and `iOptions` parameters.

iFilter

A pointer to a filter specification. Pass `NULL` if you do not want to apply a filter to this iteration. Otherwise, you can use this parameter to restrict the iteration to the fonts that match a generation count or criteria you specify in a custom filter function. Pass the filter selector constant `kATSFontFilterSelectorGeneration` to select a generation filter or the constant `kATSFontFilterSelectorFontApplierFunction` to select a custom filter. See "Font Filter Selectors" (page 70) for more information on these constants.

iRefCon

An arbitrary 32-bit value specified by your application. If you are using a custom filter function, you can use this parameter to pass data to the custom filter function. Otherwise, pass `NULL`.

iOptions

A value that specifies the scope of the iterator. If you want to iterate through fonts that can be used only by your application, pass the constant `kATSOptionFlagsRestrictedScope`. If you want to iterate through fonts that can be used by all applications pass the constant `kATSOptionFlagsUnRestrictedScope`. See "Scoping Options" (page 75) for more information on the constants you can supply. See the Discussion for information on the interaction between the `iContext` and `iOptions` parameters.

ioIterator

A pointer to a font iterator. On input, pass a pointer to an uninitialized iterator. On output, the iterator's contents may have been changed and may include references to data structures allocated by the system to maintain the iterator's state. When you no longer need the font iterator, you should call the function `ATSFontIteratorRelease` (page 38) to release the auxiliary data and memory allocated by the system.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

Your application can use a font iterator to access font data. A font iterator is an opaque data structure used by ATS for Fonts to keep track of an iteration over currently active fonts. When the font iterator is initialized, it does not yet reference a font.

The context and scope you specify for the font iterator interact as shown in Table 2 (page 37).

Table 2 The interaction of context and scope in a font enumeration

	Local context	Global context
Restricted scope	Fonts activated locally to your application	Only globally activated fonts
Unrestricted scope	Defaults fonts, which include globally activated fonts and fonts activated locally to your application	All fonts, which include globally activated fonts and all other fonts activated locally for an application.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontIteratorNext

Obtains the next font reference.

```
OSStatus ATSFontIteratorNext (
   ATSFontIterator iIterator,
   ATSFontRef *oFont
);
```

Parameters

iIterator

A pointer to a font iterator you created with the function `ATSFontIteratorCreate` (page 35). If you try to use the font iterator after disposing of its contents through this function, ATS for Fonts returns an error code to your application.

oFont

A pointer to a font reference. On output, points to the font reference obtained by the iterator. You are responsible for allocating memory for the font reference.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
If any changes are made to the font database while you are using the font iterator, the iterator is invalidated and the function `ATSFontFamilyIteratorNext` returns the error `kATSIterationScopeModified`. To remedy this error, your application must either restart or cancel the enumeration by calling the `ATSFontFamilyIteratorReset` (page 23) or the `ATSFontIteratorRelease` (page 38) functions.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
`ATSFont.h`

ATSFontIteratorRelease

Releases a font iterator.

```
OSStatus ATSFontIteratorRelease (
   ATSFontIterator *ioIterator
);
```

Parameters

ioIterator

A pointer to a font iterator you created with the function `ATSFontIteratorCreate` (page 35). If you try to use the font iterator after disposing of its contents through this function, ATS for Fonts returns an error code to your application.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

If you plan to use the font iterator again, you should consider calling the function `ATSFontIteratorReset` rather than releasing the font iterator and then creating it again.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In

`ATSFont.h`

ATSFontIteratorReset

Resets a font iterator to the beginning of the iteration.

```
OSStatus ATSFontIteratorReset (
   ATSFontContext iContext,
   const ATSFontFilter *iFilter,
   void *iRefCon,
   ATSOptionFlags iOptions,
   ATSFontIterator *ioIterator
);
```

Parameters

iContext

A value that specifies the context of the iterator. If you want to apply the font iterator only to the fonts accessible from your application use the `kATSFontContextLocal` constant. If you want the to apply the font iterator to all fonts registered with the system use the constant `kATSFontContextGlobal`. See "Context Options" (page 68) for more information.

iFilter

A pointer to a filter specification. Pass `NULL` if you do not want to apply a filter to this iteration. Otherwise, you can use this parameter to restrict the iteration to the fonts that match a generation count or criteria you specify in a custom filter function. Pass the filter selector constant `kATSFontFilterSelectorGeneration` to select a generation filter or the constant `kATSFontFilterSelectorFontApplierFunction` to select a custom filter. See "Font Filter Selectors" (page 70) for more information on these constants.

iRefCon

An arbitrary 32-bit value specified by your application. If you are using a custom filter function, you can use this parameter to pass data to the custom filter function. Otherwise, pass `NULL`.

iOptions

A value that specifies the scope of the iterator. If you want to iterate through fonts that can be used only by your application, pass the constant `kATSOptionFlagsRestrictedScope`. If you want to iterate through fonts that can be used by all applications pass the constant `kATSOptionFlagsUnRestrictedScope`.

ioIterator

A pointer to a font iterator you created with the function `ATSFontIteratorCreate` (page 35). If you try to use the font iterator after disposing of its contents through this function, ATS for Fonts returns an error code to your application.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

Once you have created a font iterator, you can reuse it by calling the function `ATSFontIteratorReset`. This function sets the parameters to the new values you specify, and repositions the iterator so it is ready to get the first font reference when you call the function `ATSFontIteratorNext` (page 37).

During an iteration, if you obtain the result code `kATSIterationScopeModified` from the function `ATSFontIteratorNext` (page 37), you can reset the iteration by calling the function `ATSFontIteratorReset`. This assures that you obtain the most up-to-date information from the iteration.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.0 and later.

Declared In

`ATSFont.h`

ATSFontNotificationSubscribe

Signs up your application to receive notification of changes to fonts and font directories.

```
OSStatus ATSFontNotificationSubscribe (
   ATSNotificationCallback callback,
   ATSFontNotifyOption options,
   void *iRefcon,
   ATSFontNotificationRef *oNotificationRef
);
```

Parameters

callback

The callback function you want ATS to invoke whenever the notification action specified in the `options` parameter occurs. See `ATSNotificationCallback` (page 49) for more information on the callback you can supply.

options

A notification option that specifies when you want ATS to respond to notification actions. If you want to receive notifications when your application is in the foreground, pass the constant `kATSFontNotifyOptionDefault`. If your application is a server process or a tool that performs font management functions and requires immediate notification when fonts change, pass the constant `kATSFontNotifyOptionReceiveWhileSuspended`. See "Notification Options" (page 74) for more information.

iRefCon

An arbitrary 32-bit value specified by your application and which you want passed to your callback function. You can pass `NULL` if your callback does not need any data.

oNotificationRef

On output, a notification reference. You need this reference when you call the function `ATSFontNotificationUnsubscribe` (page 40). You can pass `NULL` if you do not want to obtain the reference.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80). This function returns `paramErr` the `callback` parameter is `NULL` and `memFullErr` if the function cannot allocate enough memory for internal data structures.

Discussion

If your application uses Carbon events or the Application Kit, you can call the function `ATSFontNotificationSubscribe` to receive notifications of changes to fonts and font directories. However, if your application is of a type that does not use a `CFRunLoop`, it can't receive notifications unless you explicitly set up a run loop. For more information on run loops, see *Overview of Programming Topic: Run Loops* on the Cocoa Developer Documentation website.

If you want to stop receiving notifications, call the function `ATSFontNotificationUnsubscribe` (page 40).

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X 10.2 and later.

Declared In

`ATSFont.h`

ATSFontNotificationUnsubscribe

Unsubscribes your application from receiving notifications of changes to fonts and font directories.

```
OSStatus ATSFontNotificationUnsubscribe (
   ATSFontNotificationRef notificationRef
);
```

Parameters

oNotificationRef

On input, the notification reference you obtained when you called the function `ATSFontNotificationSubscribe` (page 39). On output, `NULL`.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80). Returns `paramErr` if you pass a `NULL` or invalid notification reference in the `oNotificationRef` parameter.

Discussion
The function `ATSFontNotificationUnsubscribe` unsubscribes your application from receiving the notification associated with the notification reference you pass to the function. You must call `ATSFontNotificationUnsubscribe` for each notification for which you want to unsubscribe.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.2 and later.

Declared In
`ATSFont.h`

ATSFontNotify

Notifies Apple Type Services of an action taken by your application.

```
OSStatus ATSFontNotify (
   ATSFontNotifyAction action,
   void *info
);
```

Parameters

action

A notification action that specifies the action taken by your application. If your application activates or deactivates fonts, you should pass `kATSFontNotifyActionFontsChanged`. If your application makes changes to any of the font directories (System, local, user, or the Classic System folder), you should pass the constant `kATSFontNotifyActionDirectoriesChanged`. See "Notification Actions" (page 73) for more information on the constants you can supply.

info

A pointer to the data you want ATS for Fonts to pass to the clients who subscribe to notifications. You can pass `NULL` if there is no data associated with this action.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80). Returns `paramErr` if you pass an invalid notification action in the `action` parameter.

Discussion
A notification is a mechanism by which your application can inform ATS for Fonts that you have changed a font or font directory. Other applications can sign up to receive notifications of these changes by calling the function `ATSFontNotificationSubscribe` (page 39). When you call the function `ATSFontNotify`, the system passes the notification along with any data you provide to every client who is signed up to receive notifications.

You can call the function `ATSFontNotify` after your application makes a batch of changes. For example, if your application calls the functions `ATSFontActivateFromFileSpecification` (page 83) or `ATSFontDeactivate` (page 16) multiple times to activate and deactivate fonts, you can set the `iOptions` parameter in these functions to `kATSOptionFlagsDoNotNotify` set. When you are done activating and deactivating fonts you can call the function `ATSFontNotify` with the `action` parameter set to `kATSFontNotifyActionFontsChanged`. Then ATS notifies all applications who subscribe to notifications of the changes you made.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.2 and later.

Declared In
`ATSFont.h`

ATSFontSetAutoActivationSettingForApplication

Sets the auto-activation setting for the specified application bundle.

```
OSStatus ATSFontSetAutoActivationSettingForApplication (
   ATSFontAutoActivationSetting iSetting,
   CFURLRef iApplicationFileURL
);
```

Parameters

iSetting

A font auto-activation setting. See "Automatic Activation Settings" (page 67).

iApplicationFileURL

A valid file URL for an application. Pass `NULL` to specify the current process.

Return Value
Returns `noErr` on success, and `paramErr` for any invalid input. May return `memFullErr` if unable to allocate temporary structures.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`ATSFont.h`

ATSFontSetEnabled

Sets a font state to enabled or disabled.

```
OSStatus ATSFontSetEnabled (
   ATSFontRef iFont,
   ATSOptionFlags iOptions,
   Boolean iEnabled
);
```

Parameters

iFont

The font reference.

iOptions

An options flag.

iEnabled

The state to set the font to. `True` for enabled, `false` for disabled.

Return Value
`kATSInvalidFontAccess` if the font reference is invalid in the current application context.

Availability
Available in Mac OS X v10.5 and later.

Declared In
ATSFont.h

ATSFontSetGlobalAutoActivationSetting

Sets the user's global auto-activation setting.

```
OSStatus ATSFontSetGlobalAutoActivationSetting (
   ATSFontAutoActivationSetting iSetting
);
```

Parameters

iSetting

A font auto-activation setting. See "Automatic Activation Settings" (page 67).

Return Value
If successful, noErr; if invalid input, paramErr.

Availability
Available in Mac OS X v10.5 and later.

Declared In
ATSFont.h

ATSGetGeneration

Obtains the generation of the font database.

```
ATSGeneration ATSGetGeneration (
   void
);
```

Return Value
A value that specifies the generation count of the font database. See the description of the ATSGeneration data type.

Discussion
Any operation that adds, deletes, or modifies one or more font families or fonts triggers an update of the font database generation count. If you want to obtain the generation of a font family, call the function ATSFontFamilyGetGeneration (page 19). If you want to obtain the generation of a font, call the function ATSFontGetGeneration (page 29).

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.

Declared In
ATSFont.h

DisposeFMFontCallbackFilterUPP

Disposes of a universal procedure pointer to a customized filter function used for fonts.

```
void DisposeFMFontCallbackFilterUPP (
   FMFontCallbackFilterUPP userUPP
);
```

Discussion
See the callback `FMFontCallbackFilterProcPtr` (page 49) for more information.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`ATSTypes.h`

DisposeFMFontFamilyCallbackFilterUPP

Disposes of a universal procedure pointer to a customized filter function used for font families.

```
void DisposeFMFontFamilyCallbackFilterUPP (
   FMFontFamilyCallbackFilterUPP userUPP
);
```

Discussion
See the callback `FMFontFamilyCallbackFilterProcPtr` (page 50) for more information.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`ATSTypes.h`

InvokeFMFontCallbackFilterUPP

Calls a customized filter function used for fonts.

```
OSStatus InvokeFMFontCallbackFilterUPP (
   FMFont iFont,
   void *iRefCon,
   FMFontCallbackFilterUPP userUPP
);
```

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
This function is not recommended nor needed, as the system invokes your filter for you.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`ATSTypes.h`

InvokeFMFontFamilyCallbackFilterUPP

Calls a customized filter function used for font families.

```
OSStatus InvokeFMFontFamilyCallbackFilterUPP (
   FMFontFamily iFontFamily,
   void *iRefCon,
   FMFontFamilyCallbackFilterUPP userUPP
);
```

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
This function is not recommended nor needed, as the system invokes your filter for you.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`ATSTypes.h`

NewFMFontCallbackFilterUPP

Creates a new universal procedure pointer (UPP) to a filter callback function that uses your criteria for filtering fonts.

```
FMFontCallbackFilterUPP NewFMFontCallbackFilterUPP (
   FMFontCallbackFilterProcPtr userRoutine
);
```

Return Value
See the description of the `FMFontCallbackFilterUPP` data type.

Discussion
See the callback `FMFontCallbackFilterProcPtr` (page 49) for more information.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`ATSTypes.h`

NewFMFontFamilyCallbackFilterUPP

Creates a new universal procedure pointer (UPP) to a filter callback function that uses your criteria for filtering font families.

```
FMFontFamilyCallbackFilterUPP NewFMFontFamilyCallbackFilterUPP (
   FMFontFamilyCallbackFilterProcPtr userRoutine
);
```

Return Value
See the description of the `FMFontFamilyCallbackFilterUPP` data type.

Discussion
See the callback `FMFontFamilyCallbackFilterProcPtr` (page 50) for more information.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`ATSTypes.h`

Callbacks by Task

ATS Callbacks

The callbacks in this section are used by ATS for Fonts.

`ATSFontApplierFunction` (page 46)
Defines a pointer to a customized function to be applied to a font iteration.

`ATSFontFamilyApplierFunction` (page 47)
Defines a pointer to a customized function to be applied to a font family iteration.

`ATSFontQueryCallback` (page 48)
Defines a pointer to a customized function that handles font queries.

`ATSNotificationCallback` (page 49)
Defines a pointer to a customized function that handles notifications.

FM Callbacks

The callbacks in this section are used by the Font Manager.

`FMFontCallbackFilterProcPtr` (page 49)
Defines a pointer to a customized filter function to be used with a font iterator.

`FMFontFamilyCallbackFilterProcPtr` (page 50)
Defines a pointer to a customized filter function to be used with a font family iterator.

Callbacks

ATSFontApplierFunction

Defines a pointer to a customized function to be applied to a font iteration.

```
typedef OSStatus (*ATSFontApplierFunction) (
    ATSFontRef iFont,
    void * iRefCon
);
```

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

```
OSStatus MyATSFontApplierFunction (
    ATSFontRef iFont,
    void * iRefCon
);
```

Parameters

iFont

A font reference. This is the font on which your callback operates.

iRefCon

An arbitrary 32-bit value specified by your application and that is passed to your callback.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

You provide a pointer to an `ATSFontApplierFunction` callback as a parameter to the function `ATSFontApplyFunction` (page 15). You can also provide a pointer to an `ATSFontApplierFunction` callback in the `ATSFontFilter` (page 52) data structure. This structure can be passed as a parameter to the functions `ATSFontIteratorCreate` (page 35) and `ATSFontIteratorReset` (page 38).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ATSFont.h`

ATSFontFamilyApplierFunction

Defines a pointer to a customized function to be applied to a font family iteration.

```
typedef OSStatus (*ATSFontFamilyApplierFunction) (
    ATSFontFamilyRef iFamily,
    void * iRefCon
);
```

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

```
OSStatus MyATSFontFamilyApplierFunction (
    ATSFontFamilyRef iFamily,
    void * iRefCon
);
```

Parameters

iFamily

A font family reference. This is the font family on which your callback operates.

iRefCon

An arbitrary 32-bit value specified by your application and that is passed to your callback.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
You provide a pointer to an `ATSFontFamilyApplierFunction` callback as a parameter to the function `ATSFontFamilyApplyFunction` (page 17). You can also provide a pointer to an `ATSFontApplierFunction` callback in the `ATSFontFilter` (page 52) data structure. This structure can be passed as a parameter to the functions `ATSFontFamilyIteratorCreate` (page 20) and `ATSFontFamilyIteratorReset` (page 23).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSFont.h`

ATSFontQueryCallback

Defines a pointer to a customized function that handles font queries.

```
typedef CFPropertyListRef (*ATSFontQueryCallback) (
    ATSFontQueryMessageID msgid,
     CFPropertyListRef data
       void * iRefCon
);
```

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

```
CFPropertyListRef MyATSFontQueryCallback (
    ATSFontQueryMessageID msgid,
     CFPropertyListRef data
       void * iRefCon
);
```

Parameters

msgid

An `ATSFontQueryMessageID` value that identifies the message type your application receives from ATS. See "Font Query Message ID" (page 73) for the constants you can supply.

data

A `CFPropertyListRef` that represents the font query. The content of the `CFPropertyList` is specific to the message type. The property list should contain data that specifies the font for which the query is sent.

iRefCon

An arbitrary 32-bit value specified by your application and that is passed to your callback.

Return Value
A `CFPropertyListRef` that represents your application's response to the query. The content of the `CFPropertyList` is specific to the message type, and it may be `NULL`.

Discussion
ATS for Fonts calls your customized function each time ATS receives a font query from another application. You provide a pointer to an `ATSFontQueryCallback` as a parameter to the function `ATSCreateFontQueryRunLoopSource` (page 12).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ATSFont.h`

ATSNotificationCallback

Defines a pointer to a customized function that handles notifications.

```
typedef void (*ATSNotificationCallback) (
   ATSFontNotificationInfoRef info,
      void * iRefCon
);
```

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

```
void MyATSNotificationCallback (
   ATSFontNotificationInfoRef info,
      void * iRefCon
);
```

Parameters

info

Reserved for future use. Currently, your callback is passed `NULL`.

iRefCon

An arbitrary 32-bit value specified by your application and that is passed to your callback.

Discussion
ATS for Fonts calls your customized function each time ATS receives a font notification from another application. You provide a pointer to an `ATSNotificationCallback` callback function as a parameter to the function `ATSFontNotificationSubscribe` (page 39).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ATSFont.h`

FMFontCallbackFilterProcPtr

Defines a pointer to a customized filter function to be used with a font iterator.

```
typedef OSStatus (*FMFontCallbackFilterProcPtr) (
    FMFont iFont,
    void * iRefCon
);
```

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

```
OSStatus MyFMFontCallbackFilterProcPtr (
    FMFont iFont,
    void * iRefCon
);
```

Parameters

iFont

A font reference. This is the font on which your callback operates.

iRefCon

A pointer to arbitrary data that defines your custom filter.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

The Font Manager calls your customized function each time it obtains a font in a font iteration. You can use a custom filter function in any Font Manager function that has a parameter of type `FMFilter`. You provide a universal procedure pointer to your filter callback function in the `FMFilter` data structure. First, you must use the `NewFMFontCallbackFilterUPP` (page 45) function to create a universal procedure pointer (UPP) of type `FMFontCallbackFilterUPP`. You can do so with code similar to the following:

```
FMFontCallbackFilterUPP MyFMFontFilterUPPMyFMFontFilterUPP =
NewFMFontCallbackFilterUPP (&MyFMFontCallbackFilterCallback)
```

Your application must specify the result code that should be returned by the Font Manager. Any value other than `noErr` will cause the iterator to ignore a font.

When you are finished with your filter callback function, you should use the `DisposeFMFontCallbackFilterUPP` (page 43) function to dispose of the UPP associated with it. However, if you plan to use the same filter callback function in subsequent calls, you can reuse the same UPP, rather than dispose of it and later create a new UPP.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ATSTypes.h`

FMFontFamilyCallbackFilterProcPtr

Defines a pointer to a customized filter function to be used with a font family iterator.

```
typedef OSStatus (*FMFontFamilyCallbackFilterProcPtr) (
    FMFontFamily iFontFamily,
    void * iRefCon
);
```

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

```
OSStatus MyFMFontFamilyCallbackFilterProcPtr (
    FMFontFamily iFontFamily,
    void * iRefCon
);
```

Parameters

iFontFamily

A font family reference. This is the font family on which your callback operates.

iRefCon

A pointer to arbitrary data that defines your custom filter.

Return Value

A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion

The Font Manager calls your customized function each time it obtains a font family in a font family iteration. You can use a custom filter function in any Font Manager function that has a parameter of type `FMFilter`. You provide a universal procedure pointer to your filter callback function in the `FMFilter` data structure. First, you must use the function `NewFMFontFamilyCallbackFilterUPP` (page 45) to create a universal procedure pointer (UPP) of type `FMFontFamilyCallbackFilterUPP`. You can do so with code similar to the following:

```
FMFontFamilyCallbackFilterUPP MyFMFontFamilyFilterUPPMyFMFontFamilyFilterUPP =
NewFMFontFamilyCallbackFilterUPP (&MyFMFontFamilyCallbackFilterCallback)
```

Your application must specify the result code that should be returned by the Font Manager. Any value other than `noErr` will cause the iterator to ignore a font family.

When you are finished with your filter callback function, you should use the `DisposeFMFontFamilyCallbackFilterUPP` (page 44) function to dispose of the UPP associated with it. However, if you plan to use the same filter callback function in subsequent calls, you can reuse the same UPP, rather than dispose of it and later create a new UPP.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ATSTypes.h`

Data Types

ATS Data Types

The data types in this section are used by ATS for Fonts.

ATSFontContainerRef

An opaque data type that represents a reference to a font file or folder.

```
typedef UInt32 ATSFontContainerRef;
```

Discussion
A font container reference is an opaque type used as a parameter in the functions `ATSFontActivateFromFileSpecification` (page 83) and `ATSFontActivateFromMemory` (page 14).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSFontFamilyIterator

An opaque data type that represents a font family iterator.

```
typedef struct ATSFontFamilyIterator_ * ATSFontFamilyIterator;
```

Discussion
You initialize a structure of type `ATSFontFamilyIterator` by calling the function `ATSFontFamilyIteratorCreate` (page 20). You should not attempt to modify the contents of a font family iterator.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSFont.h`

ATSFontFamilyRef

An opaque data type that represents a font family reference.

```
typedef UInt32 ATSFontFamilyRef;
```

Discussion
Unlike font family and font names which are part of a font's data, data types, such as `ATSFontFamily` represent values that are arbitrarily assigned by ATS at system startup. These values can change when the system is restarted.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSFontFilter

Contains font filter information.

```
struct ATSFontFilter {
    UInt32 version
    ATSFontFilterSelector filterSelector
    union {
        ATSGeneration generationFilter;
        ATSFontFamilyRef fontFamilyFilter;
        ATSFontFamilyApplierFunction fontFamilyApplierFunctionFilter;
        ATSFontApplierFunction fontApplierFunctionFilter;
    } filter;
};
typedef struct ATSFontFilter ATSFontFilter;
```

Fields

`version`
The version of the filter.

`filterSelector`
A font filter selector. See "Font Filter Selectors" (page 70)for a list of the filter selectors you can specify.

`filter`
A union whose contents are specified by the `filterSelector` field.

`generationFilter`
An `ATSGeneration` value that specifies the generation to which you want to restrict an operation.

`fontFamilyFilter`
A font family reference that specifies the font family to which you want to restrict an operation.

`fontFamilyApplierFunctionFilter`
A pointer the callback you want applied to a font family iteration. See `ATSFontFamilyApplierFunction` (page 47) for more information on the callback you can supply.

`fontApplierFunctionFilter`
A pointer the callback you want applied to a font iteration. See `ATSFontApplierFunction` (page 46) for more information on the callback you can supply.

Discussion

You can pass an `ATSFontFilter` structure to the functions `ATSFontFamilyIteratorCreate` (page 20), `ATSFontFamilyIteratorReset` (page 23), `ATSFontIteratorCreate` (page 35), and `ATSFontIteratorReset` (page 38).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ATSFont.h`

ATSFontIterator

An opaque data type that represents a font iterator.

```
typedef struct ATSFontIterator_ * ATSFontIterator;
```

Discussion

You initialize a structure of type `ATSFontIterator` by calling the function `ATSFontIteratorCreate` (page 35). You should not attempt to modify the contents of a font iterator.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSFont.h`

ATSFontMetrics

Contains metrics for a font.

```
struct ATSFontMetrics {
    UInt32 version;
    Float32 ascent;
    Float32 descent;
    Float32 leading;
    Float32 avgAdvanceWidth;
    Float32 maxAdvanceWidth;
    Float32 minLeftSideBearing;
    Float32 minRightSideBearing;
    Float32 stemWidth;
    Float32 stemHeight;
    Float32 capHeight;
    Float32 xHeight;
    Float32 italicAngle;
    Float32 underlinePosition;
    Float32 underlineThickness;
};
typedef struct ATSFontMetrics ATSFontMetrics;
```

Fields

`version`
: The version of the font metrics structure.

`ascent`
: The maximum height from the baseline to the ascent line of the glyphs in the font. For vertical text, the maximum distance from the center line to the ascent line of the glyphs in the font.

`descent`
: The maximum distance from the baseline to the descent line of the the glyphs in the font. For vertical text, the maximum distance from center line to the descent line of the glyphs in the font.

`leading`
: The spacing from the descent line to the ascent line below it. This defines the spacing between lines of text

`avgAdvanceWidth`
: The average advance width of the glyph in the font.

`maxAdvanceWidth`
: The maximum advance width of the glyphs in the font.

`minLeftSideBearing`
: The minimum left-side bearing value of the glyphs in the font. For vertical text, the minimum top-side bearing value of the glyphs in the font.

`minRightSideBearing`
: The minimum right-side bearing value of the glyphs in the font. For vertical text, the minimum bottom side bearing of a glyphs in the font.

`stemWidth`
: The width of the dominant vertical stems of the glyphs in the font.

`stemHeight`
: The vertical width of the dominant horizontal stems of glyphs in the font.

`capHeight`
: The height of a capital letter in the font from the baseline to the top of the letter.

`xHeight`
: The height of lowercase characters in the font, specifically the letter x, excluding ascenders and descenders.

`italicAngle`
: The angle (in degrees counterclockwise) at which glyphs in the font slant when italicized.

`underlinePosition`
: The position at which an underline stroke should be placed for the font.

`underlineThickness`
: The thickness, in pixels, of the underscore character used to underline the glyphs in the font.

Discussion
This structure is passed as a parameter to the functions `ATSFontGetHorizontalMetrics` (page 30) and `ATSFontGetVerticalMetrics` (page 34).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSFontNotificationInfoRef

An opaque data type that represents a font notification information structure.

```
typedef struct ATSFontNotificationInfoRef_ * ATSFontNotificationInfoRef;
```

Discussion
This data type is used in the `ATSNotificationCallback` (page 49) callback function.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ATSFont.h`

ATSFontNotificationRef

An opaque data type that represents a font notification structure.

```
typedef struct ATSFontNotificationRef_ * ATSFontNotificationRef;
```

Discussion
The `ATSFontNotificationRef` data type is returned by the function `ATSFontNotificationSubscribe` (page 39) and passed as a parameter to the function `ATSFontNotificationUnsubscribe` (page 40).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ATSFont.h`

ATSFontQuerySourceContext

Contains font query information that is passed back to a font query callback.

```
struct ATSFontQuerySourceContext {
    UInt32 version;
    void * refCon;
    CFAllocatorRetainCallBack retain;
    CFAllocatorReleaseCallBack release;
};
typedef struct ATSFontQuerySourceContext ATSFontQuerySourceContext;
```

Fields

`version`
A 32-bit unsigned integer that indicates the version of this data structure. You can set this value to `0`.

`refCon`
An arbitrary 32-bit value specified in your font query callback function.

`retain`
A callback you supply for increasing the retention count associated with the `refCon` value. The `CFAllocatorRetainCallBack` is defined in the header file `CFBase.h`. For more information on Core Foundation allocators, see the *Core Foundation Base Services Reference*.

`release`
A callback you supply for decreasing the retention count associated with the `refCon` value. The `CFAllocatorReleaseCallBack` is defined in the header file `CFBase.h`.

Discussion
You pass a `ATSFontQuerySourceContext` data structure to the function `ATSCreateFontQueryRunLoopSource` (page 12).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ATSFont.h`

ATSFontRef

An opaque data type that represents a font reference.

```
typedef UInt32 ATSFontRef;
```

Discussion
Unlike font names which are part of a font's data, data types, such as `ATSFontRef` represent values that are arbitrarily assigned by ATS at system startup. These values can change when the system is restarted.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSFontSize

Represents a font size.

```
typedef Float32 ATSFontSize;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSGeneration

Represents a generation count.

```
typedef UInt32 ATSGeneration;
```

Discussion
The generation count data type is used by ATS for Fonts to keep track of the generation of the font database, each font family, and each font. You can obtain a generation count from the functions `ATSGetGeneration` (page 43), `ATSFontFamilyGetGeneration` (page 19), and `ATSFontGetGeneration` (page 29).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSOptionFlags

Represents options you can pass to various ATS functions.

```
typedef OptionBits ATSOptionFlags;
```

Discussion
There are a variety of options associated with this data type. See "Assorted Options" (page 66), "Scoping Options" (page 75), and "Iteration Precedence Options" (page 73).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

FM Data types

The data types in this section are used by the Font Manager.

FMFilter

Contains a filter format, a selector and filter information.

```
struct FMFilter {
    UInt32 format
    FMFilterSelector selector
    union {
        FourCharCode fontTechnologyFilter;
        FSSpec fontContainerFilter;
        FMGeneration generationFilter;
        FMFontFamilyCallbackFilterUPP fontFamilyCallbackFilter;
        FMFontCallbackFilterUPP fontCallbackFilter;
        FMFontDirectoryFilter fontDirectoryFilter;
    } filter;
};
typedef struct FMFilter FMFilter;
```

Fields

`format`
: A filter format. For possible values, see FM Filter Format (page 76).

`selector`
: A filter selector. The selector indicates the data contained in the union. For possible values, see "FM Filter Selectors" (page 76).

`filter`
: The filter you want to use to restrict an operation. The filter must correspond to the `selector` parameter. If you are using a custom filter, you should provide a universal procedure pointer that is either of type `FMFontFamilyCallbackFilterUPP` or `FMFontCallBackFilterUPP`.

`fontTechnologyFilter`
: A `FourCharCode` value that specifies the font technology to which you want to restrict an operation. See "FM Font Technologies" (page 77) for constants you can supply.

`fontContainerFilter`
: A pointer to the file specification that specifies the name and location of a file or directory to which you want to restrict an operation.

`generationFilter`
: The generation count to which you want to restrict an operation.

`fontFamilyCallbackFilter`
: The font family callback that you want to use to restrict an operation.

`fontCallbackFilter`
: The font callback that you want to use to restrict an operation.

`fontDirectoryFilter`
: The font directory filter that you want to use to restrict an operation.

Discussion

You use the `FMFilter` (page 58) data structure when you want to restrict the enumeration and activation functions to the criteria specified by a filter.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

FMFont

An opaque data type that specifies a font registered with the font database.

```
typedef UInt32 FMFont;
```

Discussion
You should not modify this value.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

FMFontCallbackFilterUPP

Defines a universal procedure pointer to a font filter callback.

```
typedef FMFontCallbackFilterProcPtr FMFontCallbackFilter;
```

Discussion
For more information, see the description of the `FMFontCallbackFilterProcPtr` (page 49) callback function.

FMFontDirectoryFilter

Contains font directory information used to restrict a font iteration.

```
struct FMFontDirectoryFilter {
    SInt16 fontFolderDomain;
    UInt32 reserved[2];
};
typedef struct FMFontDirectoryFilter FMFontDirectoryFilter;
```

Fields

`fontFolderDomain`
A signed 16-bit integer that specifies the directory to which you want to restrict the font iteration.

`reserved`
Reserved for future use.

Discussion
You supply the `FMFontDirectoryFilter` data structure as part of the `FMFilter` (page 58) data structure when you want to restrict a font iteration to a font directory.

Availability
Available in Mac OS X v10.0 and later.

Declared In
ATSTypes.h

FMFontFamily

A reference to a collection of fonts with the same design characteristics.

```
typedef SInt16 FMFontFamily;
```

Discussion
The font family reference replaces the QuickDraw font ID and can be used with all QuickDraw functions including GetFontName and TextFont. Unlike the QuickDraw font identifier, the font family reference cannot be passed to the Resource Manager to access information from a 'FOND' resource. A font family reference does not imply a script system, nor is the character encoding of a font family determined by an arithmetic mapping of the font family reference.

The fonts associated with a font family consist of individual outline fonts that may be used with the font access functions of the Font Manager and ATSUI.

Availability
Available in Mac OS X v10.0 and later.

Declared In
ATSTypes.h

FMFontFamilyCallbackFilterUPP

Defines a universal procedure pointer to a font family filter callback.

```
typedef FMFontFamilyCallbackFilterProcPtr FMFontFamilyCallbackFilter;
```

Discussion
For more information, see the description of the FMFontFamilyCallbackFilterProcPtr (page 50) callback function.

FMFontFamilyInstance

Contains a font family reference and a QuickDraw style.

```
struct FMFontFamilyInstance {
    FMFontFamily fontFamily;
    FMFontStyle fontStyle;
};
typedef struct FMFontFamilyInstance FMFontFamilyInstance;
```

Fields
fontFamily
A font family reference.
fontStyle
A QuickDraw font style.

Discussion
Each font object can map to one or more font family instance. This mapping is equivalent to the information stored in the font association table of the `'FOND'` resource, except the font family instance does not contain a point size descriptor. Since a font object represents the entire array of point sizes for a given font, only the font family reference and style are required to specify fully any given font object.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

FMFontFamilyInstanceIterator

An opaque structure used to enumerate font family instances.

```
struct FMFontFamilyInstanceIterator {
    UInt32 reserved[16];
};
typedef struct FMFontFamilyInstanceIterator FMFontFamilyInstanceIterator;
```

Fields
`reserved`
Reserved for Apple's use.

Discussion
You initialize a structure of type `FMFontFamilyInstanceIterator` by calling the function `FMCreateFontFamilyInstanceIterator`. You should not attempt to modify the contents of a font family instance iterator.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

FMFontFamilyIterator

An opaque structure used to enumerate font families.

```
struct FMFontFamilyIterator {
    UInt32 reserved[16];
};
typedef struct FMFontFamilyIterator FMFontFamilyIterator;
```

Fields
`reserved`
Reserved for Apple's use.

Discussion
You initialize a structure of type `FMFontFamilyIterator` by calling the function `FMCreateFontFamilyIterator`. You should not attempt to modify the contents of a font family iterator.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

FMFontIterator

An opaque structure used to enumerate fonts.

```
struct FMFontIterator {
    UInt32 reserved[16];
};
typedef struct FMFontIterator FMFontIterator;
```

Fields
`reserved`
Reserved for Apple's use.

Discussion
You initialize a structure of type `FMFontIterator` by calling the function `FMCreateFontIterator`. You should not attempt to modify the contents of a font iterator.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

FMFontSize

Represents a font size.

```
typedef SInt16 FMFontSize;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

FMFontStyle

Represents a font style.

```
typedef SInt16 FMFontStyle;
```

Discussion
The low 8 bits of a Font Manager font style correspond to a QuickDraw style.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

FMGeneration

Keeps track of any operation that adds, deletes, or modifies one or more fonts or font family objects.

```
typedef UInt32 FMGeneration;
```

Discussion
Any operation that adds, deletes, or modifies one or more fonts or font family objects triggers an update of a global generation seed value. Each font and font family modified during a transaction is tagged with a copy of the generation seed.

You can use the function `FMGetGeneration` to get the current value of the generation seed. Then you can use this information in conjunction with the functions `FMGetFontGeneration` and `FMGetFontFamilyGeneration` to identify any changes in the font database.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSUI Data Types

The data types in this section are used by Apple Type Services for Unicode Imaging (ATSUI).

ATSGlyph

Represents a glyph code.

```
typedef UInt16 ATSGlyph;
```

ATSGlyphIdealMetrics

Contains ideal (resolution-independent) metrics for a glyph.

```
struct ATSGlyphIdealMetrics {
    Float32Point advance;
    Float32Point sideBearing;
    Float32Point otherSideBearing;
};
typedef struct ATSGlyphIdealMetrics ATSGlyphIdealMetrics;
```

Fields

`advance`
The amount by which the pen is advanced after drawing the glyph.

`sideBearing`
The offset from the glyph origin to the beginning of the glyph image.

`otherSideBearing`
The offset from the end of the glyph image to the end of the glyph advance.

Discussion
This data structure is passed as a parameter to the ATSUI function `ATSUGlyphGetIdealMetrics`. For more information, see *Inside Mac OS X: ATSUI Reference*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSGlyphRef

Represents a glyph reference.

```
typedef UInt16 ATSGlyphRef;
```

Discussion
This data type is used in the ATSUI data structure `ATSLayoutRecord`. For information, see *Inside Mac OS X: ATSUI Reference*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSGlyphScreenMetrics

Contains device-adjusted font metric information for glyphs in a font.

```
struct ATSGlyphScreenMetrics {
    Float32Point deviceAdvance;
    Float32Point topLeft;
    UInt32 height;
    UInt32 width;
    Float32Point sideBearing;
    Float32Point otherSideBearing;
};
typedef struct ATSGlyphScreenMetrics ATSGlyphScreenMetrics;
```

Fields

`deviceAdvance`
: The number of pixels of the advance for the glyph as actually drawn on the screen.

`topLeft`
: The top-left point of the glyph in device coordinates.

`height`
: The height of the glyph, in pixels. The glyph specified by this value may overlap with other glyphs when drawn.

`width`
: The width of the glyph, in pixels. The glyph specified by this value may overlap with other glyphs when drawn.

`sideBearing`
The origin-side bearing, in pixels.

`otherSideBearinge`
The trailing-side bearing, in pixels.

Discussion
The `ATSGlyphScreenMetrics` data structure contains metrics for where glyphs should be drawn on the screen. The metrics include any adjustments needed to display the glyphs properly on the current screen. The structure is returned by the ATSUI function `ATSUGlyphGetScreenMetrics`. Many of the metrics in this structure are `Float32Point` data types so the metrics can integrate with Quartz functions, which all require `Float32Point` data types.

For information on the ATSUI function `ATSUGlyphGetScreenMetrics`, see *Inside Mac OS X: ATSUI Reference*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSUCurvePath

Contains curve information for a glyph path.

```
struct ATSUCurvePath {
    UInt32 vectors;
    UInt32 controlBits[1];
    Float32Point vector[1];
};
typedef struct ATSUCurvePath ATSUCurvePath;
```

Fields
`vectors`
The number of values in each of the `controlBits` and `vector` arrays.

`controlBits`
An array of control bit values that, together with the values in the `vector` array, define one cubic curve in a glyph.

`vector`
An array of vector values that, together with the values in the `controlBits` array, define one cubic curve in a glyph.

Discussion
This data structure is used in the `ATSUCurvePaths` (page 66) data structure. The `ATSUCurvePaths` data structure is passed as a parameter to the ATSUI function `ATSUGlyphGetCurvePaths`. For more information, see *Inside Mac OS X: ATSUI Reference*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`ATSTypes.h`

ATSUCurvePaths

Contains curve information for an array of glyph paths.

```
struct ATSUCurvePaths {
    UInt32 contours;
    ATSUCurvePath contour[1];
};
typedef struct ATSUCurvePaths ATSUCurvePaths;
```

Fields

`contours`

The number of cubic curves contained in the `contour` array.

`contour`

An array of cubic curves that define the outline of a glyph.

Discussion

The `ATSUCurvePaths` data structure is passed as a parameter to the ATSUI function `ATSUGlyphGetCurvePaths`. For more information, see *Inside Mac OS X: ATSUI Reference*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ATSTypes.h`

GlyphID

Represents a reference to a glyph.

```
typedef ATSGlyphRef GlyphID;
```

Discussion

The `GlyphID` data type is used by ATSUI. For more information, see *Inside Mac OS X: ATSUI Reference*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`ATSTypes.h`

Constants

ATS Constants

Assorted Options

Specify assorted options.

```
enum {
    kATSOptionFlagsDefault = kNilOptions,
    kATSOptionFlagsComposeFontPostScriptName = 1 << 0,
    kATSOptionFlagsUseDataForkAsResourceFork = 1 << 8,
    kATSOptionFlagsUseResourceFork = 2 << 8,
    kATSOptionFlagsUseDataFork = 3 << 8
};
```

Constants

`kATSOptionFlagsDefault`
Specifies to use the default setting.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

`kATSOptionFlagsComposeFontPostScriptName`
Specifies the composed PostScript name of a font.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

`kATSOptionFlagsUseDataForkAsResourceFork`
Specifies to use the data fork of a font as a resource fork. You can pass this option in the `iOptions` parameter for the function `ATSFontActivateFromFileSpecification` (page 83).

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

`kATSOptionFlagsUseResourceFork`
Specifies to use the resource fork of a font.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

`kATSOptionFlagsUseDataFork`
Specifies to use the data fork of a font.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

Automatic Activation Settings

Values for automatic activation settings.

```
enum {
  kATSFontAutoActivationDefault = 0,
  kATSFontAutoActivationDisabled = 1,
  kATSFontAutoActivationEnabled = 2,
  kATSFontAutoActivationAsk     = 4
}
typedef UInt32  ATSFontAutoActivationSetting;
```

Constants

`kATSFontAutoActivationDefault`

Resets the setting the the default state. For application settings this clears the setting. For the global setting, it reverts to the initial system setting, `kATSFontAutoActivationEnabled`.

Available in Mac OS X v10.5 and later.

Declared in `ATSFont.h`.

`kATSFontAutoActivationAsk`

Asks the user before automatically activating fonts requested by the application.

Available in Mac OS X v10.5 and later.

Declared in `ATSFont.h`.

`kATSFontAutoActivationEnabled`

Enables automatic activation of fonts.

Available in Mac OS X v10.5 and later.

Declared in `ATSFont.h`.

`kATSFontAutoActivationDisabled`

Disables automatic activation of fonts.

Available in Mac OS X v10.5 and later.

Declared in `ATSFont.h`.

Declared In

`ATSFont.h`

Context Options

Specify a context to use when enumerating, activating, or deactivating fonts and font families.

```
typedef UInt32 ATSFontContext;
enum {
    kATSFontContextUnspecified = 0,
    kATSFontContextGlobal = 1,
    kATSFontContextLocal = 2
};
```

Constants

`kATSFontContextUnspecified`

Indicates a context is not specified. This option has the same result as providing the option `kATSFontContextLocal`.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

`kATSFontContextGlobal`
Specifies to use a global context. Fonts with a global context are available to all applications on the system.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

`kATSFontContextLocal`
Specifies to use a local context. Fonts with a local context are available to your application.

Available in Mac OS X v10.1 and later.

Declared in `ATSFont.h`.

Discussion
Context refers to the font's availability and can be local or global. You provide a context as an option to such functions as `ATSFontActivateFromFileSpecification` (page 83), `ATSFontActivateFromMemory` (page 14), `ATSFontFamilyIteratorCreate` (page 20), `ATSFontFamilyIteratorReset` (page 23). `ATSFontIteratorCreate` (page 35), and `ATSFontIteratorReset` (page 38).

Data Not Specified Constants

Indicate data that is not specified.

```
enum {
    kATSGenerationUnspecified = 0,
    kATSFontContainerRefUnspecified = 0,
    kATSFontFamilyRefUnspecified = 0,
    kATSFontRefUnspecified = 0
};
```

Constants
`kATSGenerationUnspecified`
Indicates the generation is not specified.

Available in Mac OS X v10.0 and later.

Declared in `ATSTypes.h`.

`kATSFontContainerRefUnspecified`
Indicates the font container reference is not specified.

Available in Mac OS X v10.0 and later.

Declared in `ATSTypes.h`.

`kATSFontFamilyRefUnspecified`
Indicates the font family reference is not specified.

Available in Mac OS X v10.0 and later.

Declared in `ATSTypes.h`.

`kATSFontRefUnspecified`
Indicates the font reference is not specified.

Available in Mac OS X v10.0 and later.

Declared in `ATSTypes.h`.

Discussion
You can pass these constants to functions when you either don't know the appropriate value or do not care to obtain the associated information. These constants can also be returned to you to indicate an error.

Font Filter Selectors

Specify the type of criteria to use when limiting an iteration.

```
enum ATSFontFilterSelector {
    kATSFontFilterSelectorUnspecified = 0,
    kATSFontFilterSelectorGeneration = 3,
    kATSFontFilterSelectorFontFamily = 7,
    kATSFontFilterSelectorFontFamilyApplierFunction = 8,
    kATSFontFilterSelectorFontApplierFunction = 9
};
typedef enum ATSFontFilterSelector ATSFontFilterSelector;
```

Constants

`kATSFontFilterSelectorUnspecified`
Specifies to limit an iteration based on unspecified criteria. In this case, the default is used, which is to iterate using a local context with an unrestricted scope.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

`kATSFontFilterSelectorGeneration`
Specifies to limit an iteration based on generation criteria.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

`kATSFontFilterSelectorFontFamily`
Specifies to limit an iteration based on font family criteria.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

`kATSFontFilterSelectorFontFamilyApplierFunction`
Specifies to limit an iteration based on criteria defined by a font family applier function.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

`kATSFontFilterSelectorFontApplierFunction`
Specifies to limit an iteration based on criteria defined by a font applier function.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

Discussion

You use these constants in the data structure `ATSFontFilter` (page 52) to specify the type of data in the `filter` union.

Font Filter Versions

Specify the version of a font filter.

```
typedef UInt32 ATSFontFormat;
enum {
    kATSFontFilterCurrentVersion = 0
};
```

Constants

`kATSFontFilterCurrentVersion`

Specifies to use the current version of a font filter.

Available in Mac OS X v10.0 and later.

Declared in `ATSFont.h`.

Discussion

There is currently only one constant in this enumeration. You can assign this constant to the `version` field in the `ATSFontFilter` (page 52) data structure.

Font Formats

Specify a font format.

```
enum {
    kATSFontFormatUnspecified = 0
};
```

Constants

`kATSFontFormatUnspecified`

Indicates the font format is not specified. You can pass this in the `iFormat` parameter of the function `ATSFontActivateFromFileSpecification` (page 83).

Available in Mac OS X v10.0 and later.

Declared in `ATSTypes.h`.

Discussion

There are no other font formats currently defined for this enumeration.

Font Request Query Keys

Represent keys in a font request query dictionary.

```
#define kATSQueryClientPID                                              CFSTR("ATS
 client
pid")
#define kATSQueryQDFamilyName               CFSTR("font family
name")
#define kATSQueryFontName                 CFSTR("font name")
#define kATSQueryFontPostScriptName       CFSTR("font PS name")
#define kATSQueryFontNameTableEntries     CFSTR("font name table
entries")
#define kATSFontNameTableCode             CFSTR("font name code")
#define kATSFontNameTablePlatform         CFSTR("font platform
code")
#define kATSFontNameTableScript           CFSTR("font script
code")
#define kATSFontNameTableLanguage         CFSTR("font language
code")
#define kATSFontNameTableBytes
CFSTR("font
name table bytes")
```

Constants

`kATSQueryClientPID`

Specifies a process ID. The value associated with this key is a `CFNumberRef` value that refers to a the process ID (`pid_t`) of the application making the query.

`kATSQueryQDFamilyName`

Specifies a QuickDraw family name. The value associated with this key is a `CFStringRef` value that refers to the QuickDraw family name of the requested font. For example, the name passed to the function `GetFNum`.

`kATSQueryFontName`

Specifies a font name. The value associated with this key is a `CFStringRef` value that refers to the full name of the requested font. You can use this font name as an argument to the function `ATSFontFindFromName` (page 25).

`kATSQueryFontPostScriptName`

Specifies the PostScript name of a font. The value associated with this key is a `CFStringRef` value that refers to either the PostScript name derived from the font's `FOND` resource or from the font's `'sfnt'` name table, with preference given to the `FOND` PostScript name. You can use this font name as an argument to the function `ATSFontFindFromPostScriptName` (page 26).

`kATSQueryFontNameTableEntries`

Specifies the descriptor for `'sfnt'` name table entries. The value associated with this key an array (`CFArrayRef`) of `CFDictionaryRef` values that describe entries in a name table. A font must have all of the specified entries to be considered a match.

`kATSFontNameTableCode`

Specifies the font name's name code. The value associated with this key is a `CFNumberRef`. If no value is specified, the value `kFontNoNameCode` is used.

`kATSFontNameTablePlatform`

Specifies the font name's platform code. The value associated with this key is a `CFNumberRef`. If no value is specified, the value `kFontNoPlatformCode` is used.

`kATSFontNameTableScript`

Specifies the font name's script code. The value associated with this key is a `CFNumberRef`. If no value is specified, the value `kFontNoScriptCode` is used.

`kATSFontNameTableLanguage`
: Specifies the font name's language code. The value associated with this key is a `CFNumberRef`. If no value is specified, the value `kFontNoLanguageCode` is used.

`kATSFontNameTableBytes`
: Specifies the raw bytes of the font name. The value associated with this key is a `CFDataRef` value that refers to the raw name bytes for the font.

Discussion
Font request query keys appear in the dictionary passed to, and returned by, your `ATSFontQueryCallback` (page 48) callback function. The keys comprise a property list (`CFPropertyList`) that defines the query sent to your callback. On return, you supply a property list that specifies your response to the query.

Font Query Message ID

Specifies a message ID for a font request query.

```
enum ATSFontQueryMessageID {
    kATSQueryActivateFontMessage = 'atsa'
};
typedef enum ATSFontQueryMessageID ATSFontQueryMessageID;
```

Constants
`kATSQueryActivateFontMessage`
: Specifies to activate a font message. The data associated with this message ID is a flattened `CFDictionaryRef`. The `CFDictionary` contains on or more of the keys described in "Font Request Query Keys" (page 71).

Available in Mac OS X v10.2 and later.

Declared in `ATSFont.h`.

Discussion
There is currently only one constant in this enumeration. You use a constant of this type when you create an `ATSFontQueryCallback` (page 48) callback function.

Iteration Precedence Options

Specify the order of an iteration.

```
enum {
    kATSOptionFlagsIterateByPrecedenceMask = 0x00000001 <<  5
};
```

Constants
`kATSOptionFlagsIterateByPrecedenceMask`
: Specifies to iterate fonts in the order dictated by a precedence mask.

Available in Mac OS X v10.1 and later.

Declared in `ATSFont.h`.

Notification Actions

Specify a notification action.

```
enum ATSFontNotifyAction {
    kATSFontNotifyActionFontsChanged = 1,
    kATSFontNotifyActionDirectoriesChanged = 2
};
typedef enum ATSFontNotifyAction ATSFontNotifyAction;
```

Constants

`kATSFontNotifyActionFontsChanged`

Specifies that your application has activated or deactivated fonts. Typically you call the functions `ATSFontActivateFromFileSpecification` (page 83) or `ATSFontDeactivate` (page 16) multiple times to activate and deactivate fonts. In each call, you set the `iOptions` parameter to `kATSOptionFlagsDoNotNotify` set. When you are done activating and deactivating fonts you can call the function `ATSFontNotify` (page 41) with the `action` parameter set to `kATSFontNotifyActionFontsChanged`. Then ATS notifies all applications who subscribe to notifications of the changes you made.

Available in Mac OS X v10.2 and later.

Declared in `ATSFont.h`.

`kATSFontNotifyActionDirectoriesChanged`

Specifies that your application has made changes to one or more of the font directories. When you are making changes to font directories, you can call the function `ATSFontNotify` (page 41) with the `action` parameter set to `kATSFontNotifyActionDirectoriesChanged`. Then ATS scans these directories and notifies all applications who subscribe to notifications of the changes you made.

Available in Mac OS X v10.2 and later.

Declared in `ATSFont.h`.

Discussion

You can use these options with the function `ATSFontNotify` (page 41).

Notification Options

Specify when ATS should notify your application of changes in the font database.

```
enum ATSFontNotifyOption {
    kATSFontNotifyOptionDefault = 0,
    kATSFontNotifyOptionReceiveWhileSuspended = 1L << 0
};
typedef enum ATSFontNotifyOption ATSFontNotifyOption;
```

Constants

`kATSFontNotifyOptionDefault`

Specifies to use the default behavior of the function `ATSFontNotificationSubscribe` (page 39).

Available in Mac OS X v10.2 and later.

Declared in `ATSFont.h`.

`kATSFontNotifyOptionReceiveWhileSuspended`

Specifies to receive notifications even if the application is in the background. Setting this option can degrade performance; you should set this option if your application is a faceless process or a tool that performs font management functions and requires immediate notification when fonts change.

Available in Mac OS X v10.2 and later.

Declared in `ATSFont.h`.

Discussion
You use notification options when you call the function `ATSFontNotificationSubscribe` (page 39). The default behavior is for applications to receive ATS notifications only when the application runs in the foreground. By default, if the application is suspended, the notification is delivered when the application comes to the foreground.

Scoping Options

Specify the scope to which an operation should apply or a notification schedule.

```
enum {
    kATSOptionFlagsDoNotNotify = 0x00000001 << 8,
    kATSOptionFlagsIterationScopeMask = 0x00000007 << 12,
    kATSOptionFlagsDefaultScope = 0x00000000 << 12,
    kATSOptionFlagsUnRestrictedScope = 0x00000001 << 12,
    kATSOptionFlagsRestrictedScope = 0x00000002 << 12,
    kATSOptionFlagsProcessSubdirectories = 0x00000001 << 6
};
```

Constants

`kATSOptionFlagsDoNotNotify`
Specifies not to send a notification after a font is activated or deactivated globally. You can set the `iOptions` parameter of the functions `ATSFontActivateFromFileSpecification` (page 83) or `ATSFontDeactivate` (page 16) to this constant. When you are done activating and deactivating fonts you can call the function `ATSFontNotify` (page 41) with the `action` parameter set to `kATSFontNotifyActionFontsChanged`. Then ATS notifies all applications who subscribe to notifications of the changes you made.

Available in Mac OS X v10.2 and later.

Declared in `ATSFont.h`.

`kATSOptionFlagsIterationScopeMask`
Specifies mask option bits 12-14 for iteration scopes.

Available in Mac OS X v10.1 and later.

Declared in `ATSFont.h`.

`kATSOptionFlagsDefaultScope`
Specifies to use the default scope, which is equivalent to `kATSOptionFlagsUnRestrictedScope`. You can pass this as a parameter to the functions `ATSFontFamilyIteratorCreate` (page 20), `ATSFontFamilyIteratorReset` (page 23), `ATSFontIteratorCreate` (page 35) and `ATSFontIteratorReset` (page 38).

Available in Mac OS X v10.1 and later.

Declared in `ATSFont.h`.

`kATSOptionFlagsUnRestrictedScope`
Specifies to use an unrestricted scope. You can pass this as a parameter to the functions `ATSFontFamilyIteratorCreate` (page 20), `ATSFontFamilyIteratorReset` (page 23), `ATSFontIteratorCreate` (page 35) and `ATSFontIteratorReset` (page 38).

Available in Mac OS X v10.1 and later.

Declared in `ATSFont.h`.

`kATSOptionFlagsRestrictedScope`
Specifies to use a restricted scope. You can pass this as a parameter to the functions `ATSFontFamilyIteratorCreate` (page 20), `ATSFontFamilyIteratorReset` (page 23), `ATSFontIteratorCreate` (page 35) and `ATSFontIteratorReset` (page 38).

Available in Mac OS X v10.1 and later.

Declared in `ATSFont.h`.

`kATSOptionFlagsProcessSubdirectories`
Specifies to process the font directories within a font directory. You can pass this as a parameter to the function `ATSFontActivateFromFileSpecification` (page 83).

Available in Mac OS X v10.2 and later.

Declared in `ATSFont.h`.

Discussion
Scope refers to whether a font's use is restricted or unrestricted. Fonts with a restricted scope can be used only by your application whereas fonts with an unrestricted scope cay be used by all applications.

Font Manager Constants

FM Filter Format

Specifies a filter format.

```
enum {
    kFMCurrentFilterFormat = 0
};
```

Constants
`kFMCurrentFilterFormat`
Specifies the current filter format. You can use this to set the format field when you initialize the `FMFilter` data type for use in creating an iterator object with the functions `FMCreateFontFamilyIterator` or `FMCreateFontIterator`. Currently, this is the only format you can specify.

Available in Mac OS X v10.0 and later.

Declared in `ATSTypes.h`.

FM Filter Selectors

Specifies a filter type.

```
typedef UInt32 FMFilterSelector;
enum {
    kFMFontTechnologyFilterSelector = 1,
    kFMFontContainerFilterSelector = 2,
    kFMGenerationFilterSelector = 3,
    kFMFontFamilyCallbackFilterSelector = 4,
    kFMFontCallbackFilterSelector = 5,
    kFMFontDirectoryFilterSelector = 6
};
```

Constants

`kFMFontTechnologyFilterSelector`
Selects font technology filter. You can use this filter only with a font iterator.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

`kFMFontContainerFilterSelector`
Selects font container filter. You can use this filter only with a font iterator.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

`kFMGenerationFilterSelector`
Selects generation filter. You can use this filter only with a font family iterator.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

`kFMFontFamilyCallbackFilterSelector`
Indicates a custom filter to be used only with a font family iterator.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

`kFMFontCallbackFilterSelector`
Indicates a custom filter to be used only with a font iterator.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

Discussion

You use these constants to specify a filter type in the `FMFilter` (page 58) data structure used by many Font Manager functions.

FM Font Technologies

Specify a font technology.

```
enum {
    kFMTrueTypeFontTechnology = 'true',
    kFMPostScriptFontTechnology = 'typ1'
};
```

Constants

`kFMTrueTypeFontTechnology`
Indicates True Type font technology.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

`kFMPostScriptFontTechnology`
Indicates Post Script font technology.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

Invalid Values

Specify an invalid value.

```
enum {
    kInvalidGeneration = 0,
    kInvalidFontFamily = -1,
    kInvalidFont = 0
};
```

Constants

`kInvalidGeneration`
Indicates an invalid generation value.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

`kInvalidFontFamily`
Indicates the font family reference is invalid.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

`kInvalidFont`
Indicates the font reference is invalid.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

Discussion

The `kInvalidGeneration`, `kInvalidFontFamily`, and `kInvalidFont` constants may be used to indicate invalid values for generation count, font family, and font data types.

ATSUI Constants

Convenience Constants

Represent numerical values that are commonly used in font calculations.

```
enum {
    kATSItalicQDSkew = (1 << 16) / 4,
    kATSBoldQDStretch = (1 << 16) * 3 / 2,
    kATSRadiansFactor = 1144
};
```

Constants

`kATSItalicQDSkew`

A `Fixed` value of `0.25` that represents the skew used by QuickDraw to draw italicized glyphs.

Available in Mac OS X v10.0 and later.

Declared in `ATSTypes.h`.

`kATSBoldQDStretch`

A `Fixed` value that represents the stretch-factor used by QuickDraw to draw bold-faced glyphs.

Available in Mac OS X v10.0 and later.

Declared in `ATSTypes.h`.

`kATSRadiansFactor`

A `Fixed` value of approximately `pi/180(0.0174560546875)` that represents an angle of 1 radian. This is a convenience constant you can use when you draw rotated text.

Available in Mac OS X v10.0 and later.

Declared in `ATSTypes.h`.

Discussion

These constants are provided for convenience. Your application can use them when it needs to perform font calculations.

Version Notes

Available beginning with ATSUI 1.0.

Curve Types

Specify a curve type used to draw a font.

```
typedef UInt16 ATSCurveType;
enum {
    kATSCubicCurveType = 0x0001,
    kATSQuadCurveType = 0x0002,
    kATSOtherCurveType = 0x0003
};
```

Constants

`kATSCubicCurveType`
Specifies a cubic curve.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

`kATSQuadCurveType`
Specifies a quadratic curve.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

`kATSOtherCurveType`
Specifies a curve other than cubic or quadratic.
Available in Mac OS X v10.0 and later.
Declared in `ATSTypes.h`.

Discussion

These are used in the ATSUI function `ATSUGetNativeCurveType`. See *Inside Mac OS X: ATSUI Reference* for more information.

Deleted Glyph Code

Specifies that a glyph is deleted.

```
enum {
    kATSDeletedGlyphcode = 0xFFFF
};
```

Constants

`kATSDeletedGlyphcode`
Indicates that a glyph is deleted. That is, the glyph is set to no longer appear in a text layout.
Available in Mac OS X v10.2 and later.
Declared in `ATSTypes.h`.

Discussion

This constant is used by ATSUI. When a glyph is deleted, ATSUI sets the corresponding `ATSGlyphRef` (page 64) to `kATSDeletedGlyphcode`. For more information, see *Inside Mac OS X: ATSUI Reference*.

Result Codes

The most common result codes returned by Apple Type Services for Fonts are listed below.

Result Code	Value	Description
`kATSIterationCompleted`	-980L	The iteration is complete. Available in Mac OS X v10.0 and later.
`kATSInvalidFontFamilyAccess`	-981L	Your application tried to access an invalid font family. Available in Mac OS X v10.0 and later.
`kATSInvalidFontAccess`	-982L	Your application tried to access an invalid font. Available in Mac OS X v10.0 and later.
`kATSIterationScopeModified`	-983L	The font database changed during an iteration. Available in Mac OS X v10.0 and later.
`kATSInvalidFontTableAccess`	-984L	Your application tried to access an invalid font table. Available in Mac OS X v10.0 and later.
`kATSInvalidFontContainerAccess`	-985L	Your application tried to access an invalid font container. Available in Mac OS X v10.0 and later.

Deprecated Apple Type Services for Fonts Functions

A function identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.5

ATSFontActivateFromFileSpecification

Activates one or more fonts from a file specification. (Deprecated in Mac OS X v10.5. Instead use `ATSFontActivateFromFileReference` (page 13).)

```
OSStatus ATSFontActivateFromFileSpecification (
   const FSSpec *iFile,
   ATSFontContext iContext,
   ATSFontFormat iFormat,
   void *iReserved,
   ATSOptionFlags iOptions,
   ATSFontContainerRef *oContainer
);
```

Parameters

iFile

A pointer to the file specification that specifies the name and location of a file or directory that contains the font data you want to activate.

iContext

A value that specifies the context of the activated font. If you want the activated font to be accessible only from your application use the `kATSFontContextLocal` constant. If you want the activated font to be accessible to all applications use the constant `kATSFontContextGlobal`. See "Context Options" (page 68) for more information.

iFormat

A value that represents the format identifier of the font. Pass `kATSFontFormatUnspecified` as the system automatically determines the format of the font. For more information on this constant, see "Font Formats" (page 71).

iReserved

An arbitrary 32-bit value. This parameter is currently reserved for future use, so you should pass `NULL`.

iOptions

An options flag. Pass `kATSOptionFlagsDefault` unless the font's data fork contains resource-fork information, you need to activate a directory of font directories, or you plan to call this function a number of time. If the font's data fork contains resource-fork information, pass the option `kATSOptionFlagsUseDataForkAsResourceFork`. If the you want to activate a font directory that contains font directories, you must pass the option `kATSOptionFlagsProcessSubdirectories`. If you plan to call this function a number of times, you can set the `iOptions` parameter to `kATSOptionFlagsDoNotNotify` set. When you are done activating fonts you can call the function `ATSFontNotify` (page 41) with the `action` parameter set to `kATSFontNotifyActionFontsChanged`. Then ATS notifies all applications who subscribe to notifications of the changes you made.

oContainer

On output, a reference to the font container that is activated from the file specification. You need this reference when you deactivate the font by calling the function `ATSFontDeactivate` (page 16).

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
You can use the function `ATSFontActivateFromFileSpecification` to activate one font or more fonts. Activating a font makes that font available for use either locally (available only to your application) or globally (available to all applications on the system). A font's availability—local or global—is referred to as its context.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`ATSFont.h`

ATSFontGetFileSpecification

Obtains the file specification for a font. (Deprecated in Mac OS X v10.5. Instead use `ATSFontGetFileReference` (page 28).)

```
OSStatus ATSFontGetFileSpecification (
   ATSFontRef iFont,
   ATSFSSpec *oFile
);
```

Parameters

iFont

A reference to the font whose file specification you want to obtain.

oFile

On output, points to the file specification that specifies the name and location of a file or directory that contains the font data specified by the `iFont` parameter.

Return Value
A result code. See "Apple Type Services for Fonts Result Codes" (page 80).

Discussion
The function `ATSFontGetFileSpecification` obtains the file specification for a font, not the font container. You must call the functions `ATSFontActivateFromFileSpecification` (page 83) or `ATSFontActivateFromMemory` (page 14) to obtain a font container reference.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`ATSFont.h`

Document Revision History

This table describes the changes to *Apple Type Services for Fonts Reference*.

Date	Notes
2007-12-11	Updated for Mac OS X v10.5.
	Added `ATSFontActivateFromFileReference` (page 13), `ATSFontSetGlobalAutoActivationSetting` (page 43), `ATSFontGetGlobalAutoActivationSetting` (page 30), `ATSFontSetAutoActivationSettingForApplication` (page 42), `ATSFontGetAutoActivationSettingForApplication` (page 26), `ATSFontGetContainerFromFileReference` (page 27), `ATSFontGetContainer` (page 27), `ATSFontGetFileReference` (page 28), `ATSFontSetEnabled` (page 42), and `ATSFontIsEnabled` (page 35).
	Added deprecation information for several functions.
2006-08-16	Made minor technical corrections.
2005-12-06	Fixed incorrect link.
2005-11-09	Provided information about three functions that return font resources in big endian format.
	See `ATSFontGetTableDirectory` (page 33), `ATSFontGetTable` (page 32), and `ATSFontGetFontFamilyResource` (page 28).
2005-07-07	Corrected typographical errors.
2003-04-23	Fixed minor typographical error.
2003-04-10	Fixed errors in the documentation for the function `ATSFontIteratorCreate` (page 35) and the description for the constants `Scoping Options` (page 75).
2002-12-09	First version of this document. Provides a complete reference to ATS for Fonts version 1.4.

Document Revision History

Index

A

C

D

F

G

I

K

N

S