CFXMLParser Reference

Core Foundation

2008-10-15

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

Simultaneously published in the United States and Canada.

Contents

CFXMLParser Reference

Derived From:	CFType
Framework:	CoreFoundation/CoreFoundation.h
Companion guide	XML Programming Topics for Core Foundation
Declared in	CFXMLParser.h

Overview

CFXMLParser provides an XML parser you can use to find and extract data in XML documents. You can use a high-level interface to load an XML document into a Core Foundation collection object. A low-level callback-based interface allows you to perform any action you wish on an XML structured type when it is detected by the parser. This opaque type is relevant for applications that need information about an XML document's structure or content.

Functions

CFXMLParserAbort

Causes a parser to abort with the given error code and description.

```
void CFXMLParserAbort (
    CFXMLParserRef parser,
    CFXMLParserStatusCode errorCode,
    CFStringRef errorDescription
);
```

Parameters

parser

The parser to abort.

errorCode

The error code to return to the parser.

errorDescription

The error description string to return to the parser. This value may not be `NULL`.

Discussion

This function cannot be called asynchronously. In other words, it must be called from within a parser callback function.

Availability
Available in CarbonLib v1.1 and later.
Available in Mac OS X v10.0 and later.

Declared In
`CFXMLParser.h`

CFXMLParserCopyErrorDescription

Returns the user-readable description of the current error condition.

```
CFStringRef CFXMLParserCopyErrorDescription (
    CFXMLParserRef parser
);
```

Parameters

parser
The XML parser to examine.

Return Value
A user-readable description of the current error condition, or `NULL` if no error occurred. Ownership follows the Create Rule.

Availability
Available in CarbonLib v1.1 and later.
Available in Mac OS X v10.0 and later.

Declared In
`CFXMLParser.h`

CFXMLParserCreate

Creates a new XML parser for the specified XML data.

```
CFXMLParserRef CFXMLParserCreate (
    CFAllocatorRef allocator,
    CFDataRef xmlData,
    CFURLRef dataSource,
    CFOptionFlags parseOptions,
    CFIndex versionOfNodes,
    CFXMLParserCallBacks *callBacks,
    CFXMLParserContext *context
);
```

Parameters

allocator
The allocator to use to allocate memory for the new object. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

xmlData
The XML data to parse. Do not pass `NULL`.

dataSource

The URL from which the XML data was obtained. The URL is used to resolve any relative references found in XML Data. Pass `NULL` if a valid URL is unavailable.

parseOptions

Flags which control how the XML data will be parsed. See Parsing Options (page 22) for the list of available options.

versionOfNodes

Determines which version of CFXMLNode objects are produced by the parser.

callBacks

Callbacks called by the parser as the XML is processed. The callbacks are called as each XML tag is encountered, when an external entity needs to be resolved, and when an error occurs. See `CFXMLParserCallBacks` (page 18) and the individual callbacks for more details. Do not pass `NULL`.

context

Determines what, if any, information pointer is passed to the callbacks as the parse progresses; *context* may be `NULL`.

Return Value

The newly created parser. Ownership follows the Create Rule.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserCreateWithDataFromURL

Creates a new XML parser for the specified XML data at the specified URL.

```
CFXMLParserRef CFXMLParserCreateWithDataFromURL (
    CFAllocatorRef allocator,
    CFURLRef dataSource,
    CFOptionFlags parseOptions,
    CFIndex versionOfNodes,
    CFXMLParserCallBacks *callBacks,
    CFXMLParserContext *context
);
```

Parameters

allocator

The allocator to use to allocate memory for the new object. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

dataSource

The URL from which to load the XML data. The URL is used to resolve any relative references found in XML Data. It must be a valid CFURL object; `NULL` is an unacceptable value.

parseOptions

Flags which control how the XML data will be parsed. See Parsing Options (page 22) for the list of available options.

versionOfNodes

Determines which version of CFXMLNode objects are produced by the parser.

callBacks

Callbacks called by the parser as the XML is processed. The callbacks are called as each XML tag is encountered, when an external entity needs to be resolved, and when an error occurs. See `CFXMLParserCallBacks` (page 18) and the individual callbacks for more details. Do not pass `NULL`.

context

Determines what, if any, information pointer is passed to the callbacks as the parse progresses; may be `NULL`.

Return Value

The newly created parser. Ownership follows the Create Rule.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserGetCallBacks

Returns the callbacks associated with an XML parser when it was created.

```
void CFXMLParserGetCallBacks (
    CFXMLParserRef parser,
    CFXMLParserCallBacks *callBacks
);
```

Parameters

parser

The XML parser to examine.

callBacks

On return, contains the callbacks for *parser*.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserGetContext

Returns the context for an XML parser.

```
void CFXMLParserGetContext (
    CFXMLParserRef parser,
    CFXMLParserContext *context
);
```

Parameters

parser

The XML parser to examine.

context

On return, a pointer to the context structure for *parser*.

Discussion

If you set a context for the parser, it will be passed to you as a parameter in each of the parser callback functions. The context data structure is application defined and associated with a parser using one of the `CFXMLParserCreate...` functions.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserGetDocument

Returns the top-most object returned by the create XML structure callback.

```
void *CFXMLParserGetDocument (
    CFXMLParserRef parser
);
```

Parameters

parser

The XML parser to examine.

Return Value

The top-most object returned by the `createXMLStructure` field in the `CFXMLParserCallBacks` (page 18) structure. If the returned value is a Core Foundation object, ownership follows the Get Rule.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserGetLineNumber

Returns the line number of the current parse location.

```
CFIndex CFXMLParserGetLineNumber (
    CFXMLParserRef parser
);
```

Parameters

parser

The XML parser to examine.

Return Value

The line number of the current location.

Discussion
This function is typically used in conjunction with the `CFXMLParserHandleErrorCallBack` (page 15) function so that error location information can be reported.

Availability
Available in CarbonLib v1.1 and later.
Available in Mac OS X v10.0 and later.

Declared In
`CFXMLParser.h`

CFXMLParserGetLocation

Returns the character index of the current parse location.

```
CFIndex CFXMLParserGetLocation (
    CFXMLParserRef parser
);
```

Parameters

parser
The XML parser to examine.

Return Value
The character index of the current parse location.

Discussion
This function is typically used in conjunction with the `CFXMLParserHandleErrorCallBack` (page 15) function so that error location information can be reported.

Availability
Available in CarbonLib v1.1 and later.
Available in Mac OS X v10.0 and later.

Declared In
`CFXMLParser.h`

CFXMLParserGetSourceURL

Returns the URL for the XML data being parsed.

```
CFURLRef CFXMLParserGetSourceURL (
    CFXMLParserRef parser
);
```

Parameters

parser
The XML parser to examine.

Return Value
The URL for the XML document being parsed. Ownership follows the Get Rule.

Availability
Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Declared In
`CFXMLParser.h`

CFXMLParserGetStatusCode

Returns a numeric code indicating the current status of the parser.

```
CFXMLParserStatusCode CFXMLParserGetStatusCode (
    CFXMLParserRef parser
);
```

Parameters

parser
The XML parser to examine.

Return Value
A status code indicating the current parser. See Parser Status Codes (page 20) for a list of possible status codes.

Discussion
If an error has occurred, the code for the last error is returned. If no error has occurred, a status code is returned.

Availability
Available in CarbonLib v1.1 and later.
Available in Mac OS X v10.0 and later.

Declared In
`CFXMLParser.h`

CFXMLParserGetTypeID

Returns the type identifier for the CFXMLParser opaque type.

```
CFTypeID CFXMLParserGetTypeID ();
```

Return Value
The type identifier for the CFXMLParser opaque type.

Availability
Available in CarbonLib v1.3 and later.
Available in Mac OS X v10.0 and later.

Declared In
`CFXMLParser.h`

CFXMLParserParse

Begins a parse of the XML data that was associated with the parser when it was created.

```
Boolean CFXMLParserParse (
    CFXMLParserRef parser
);
```

Parameters

parser

The XML parser to start.

Return Value

`true` if the parse was successful, `false` otherwise.

Discussion

Upon success, use the `CFXMLParserGetDocument` (page 9) function to get the product of the parse. Upon failure, use the `CFXMLParserGetContext` (page 8) or `CFXMLParserCopyErrorDescription` (page 6) functions to get information about the error. It is an error to call the `CFXMLParserParse` (page 11) function while a parse is already underway.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

Callbacks

CFXMLParserAddChildCallBack

Callback function invoked by the parser to notify your application of parent/child relationships between XML structures.

```
typedef void (*CFXMLParserAddChildCallBack) (
    CFXMLParserRef parser,
    void *parent,
    void *child,
    void *info
);
```

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

```
void MyCallBack (
    CFXMLParserRef parser,
    void *parent,
    void *child,
    void *info
);
```

Parameters

parser

The CFXMLParser object making the callback.

parent

The program-defined value representing the XML element to whom *child* is being added. This value was returned by the CFXMLParserCreateXMLStructureCallBack (page 14) callback when this element's open tag was detected.

child

The program-defined value representing the XML element that is being added to *parent*. This value was returned by the CFXMLParserCreateXMLStructureCallBack (page 14) callback when this element's open tag was detected.

info

The program-defined context data you specified in the CFXMLParserContext (page 19) structure when creating the parser.

Discussion
If the CFXMLParserCreateXMLStructureCallBack (page 14) function returns NULL for a given structure, that structure is omitted entirely, and this callback will *not* be called for either a NULL child or parent.

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

Declared In
CFXMLParser.h

CFXMLParserCopyDescriptionCallBack

Callback function invoked by the parser when handling the information pointer.

```
typedef CFStringRef (*CFXMLParserCopyDescriptionCallBack) (
    const void *info
);
```

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

```
CFStringRef MyCallBack (
    const void *info
);
```

Parameters

info

The program-defined context data you specified in the CFXMLParserContext (page 19) structure when creating the parser.

Return Value
A textual description of *info*. The caller is responsible for releasing this object.

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

Declared In
CFXMLParser.h

CFXMLParserCreateXMLStructureCallBack

Callback function invoked when the parser encounters an XML open tag.

```
typedef void *(*CFXMLParserCreateXMLStructureCallBack) (
    CFXMLParserRef parser,
    CFXMLNodeRef nodeDesc,
    void *info
);
```

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

```
void *MyCallBack (
    CFXMLParserRef parser,
    CFXMLNodeRef nodeDesc,
    void *info
);
```

Parameters

parser

The CFXMLParser object making the callback.

nodeDesc

The CFXMLNode object that represents the XML structure encountered.

info

The program-defined context data you specified in the `CFXMLParserContext` (page 19) structure when creating the parser.

Return Value

A program-defined value representing the new XML element or `NULL` to indicate that the given structure should be skipped. This value is passed to the other callbacks.

Discussion

If `NULL` is returned for a given structure, only minimal parsing is done for that structure (enough to correctly determine its end, and to extract any data necessary for the remainder of the parse, such as Entity definitions). This callback (or any of the tree-creation callbacks) will not be called for any children of the skipped structure. The only exception is that the top-most element will always be reported even if `NULL` was returned for the document as a whole. For performance reasons, the node passed to this callback cannot be safely retained by the client; the node as a whole must be copied (using the `CFXMLNodeCreateCopy` function), or its contents must be extracted and copied. You are required to implement this callback for the parser to operate.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserEndXMLStructureCallBack

Callback function invoked by the parser to notify your application that an XML structure (and all its children) have been completely parsed.

```
typedef void (*CFXMLParserEndXMLStructureCallBack) (
    CFXMLParserRef parser,
    void *xmlType,
    void *info
);
```

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

```
void MyCallBack (
    CFXMLParserRef parser,
    void *xmlType,
    void *info
);
```

Parameters

parser

The CFXMLParser object making the callback.

xmlType

The program-defined value representing the XML element whose end tag has been detected. This value was returned by the `CFXMLParserCreateXMLStructureCallBack` (page 14) callback.

info

The program-defined context data you specified in the `CFXMLParserContext` (page 19) structure when creating the parser.

Discussion

As elements are encountered, this callback is called first, then the `CFXMLParserAddChildCallBack` (page 12) callback to add the new structure to its parent, then the `CFXMLParserAddChildCallBack` (page 12) callback (potentially several times) to add the new structure's children to it, and then finally the `CFXMLParserEndXMLStructureCallBack` (page 14) callback to show that the structure has been fully parsed.This callback is optional.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserHandleErrorCallBack

Callback function invoked by the parser to notify your application that an error has occurred.

```
typedef Boolean (*CFXMLParserHandleErrorCallBack) (
    CFXMLParserRef parser,
    CFXMLParserStatusCode error,
    void *info
);
```

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

```
Boolean MyCallBack (
    CFXMLParserRef parser,
    CFXMLParserStatusCode error,
    void *info
```

```
);
```

Parameters

parser

A CFXMLParser object making the callback.

error

A status code describing the error.

info

The program-defined context data you specified in the `CFXMLParserContext` (page 19) structure when creating the parser.

Return Value

`true` if the parser should continue parsing the XML, `false` if the parser should stop.

Discussion

If this callback is not defined, the parser will silently attempt to recover. Otherwise, this callback may return `false` to force the parser to stop. If this callback returns `true`, the parser will attempt to recover (fatal errors will still cause the parse to abort immediately). This callback is optional.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserReleaseCallBack

Callback function invoked by the parser when it wants to release a reference to the information pointer.

```
typedef void (*CFXMLParserReleaseCallBack) (
    const void *info
);
```

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

```
void MyCallBack (
    const void *info
);
```

Parameters

info

The program-defined context data you specified in the `CFXMLParserContext` (page 19) structure when creating the parser.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserResolveExternalEntityCallBack

Callback function invoked by the parser to notify your application that an external entity has been referenced.

```
typedef CFDataRef (*CFXMLParserResolveExternalEntityCallBack) (
    CFXMLParserRef parser,
    CFXMLExternalID *extID,
    void *info
);
```

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

```
CFDataRef MyCallBack (
    CFXMLParserRef parser,
    CFXMLExternalID *extID,
    void *info
);
```

Parameters

parser

The CFXMLParser object making the callback.

extID

The identifier for the external entity.

info

The program-defined context data you specified in the `CFXMLParserContext` (page 19) structure when creating the parser.

Return Value

The external entity or `NULL` if it should not be resolved.

Discussion

If this callback is not defined, the parser uses its internal routines to try and resolve the entity. Otherwise, if this callback returns `NULL`, a place holder for the external entity is inserted into the tree. In this manner, the parser's client can prevent any external network or file accesses. This callback is optional.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserRetainCallBack

Callback function invoked by the parser when it needs another reference to the information pointer.

```
typedef const void *(*CFXMLParserRetainCallBack) (
    const void *info
);
```

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

```
const void *MyCallBack (
    const void *info
);
```

Parameters

info

The program-defined context data you specified in the `CFXMLParserContext` (page 19) structure when creating the parser.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

Data Types

CFXMLParserCallBacks

Contains version information and function pointers to callbacks needed when parsing XML.

```
struct CFXMLParserCallBacks {
    CFIndex version;
    CFXMLParserCreateXMLStructureCallBack createXMLStructure;
    CFXMLParserAddChildCallBack addChild;
    CFXMLParserEndXMLStructureCallBack endXMLStructure;
    CFXMLParserResolveExternalEntityCallBack resolveExternalEntity;
    CFXMLParserHandleErrorCallBack handleError;
};
typedef struct CFXMLParserCallBacks CFXMLParserCallBacks;
```

Fields

`version`

Version number. Must be `0`.

`createXMLStructure`

Called when an XML structure is created.

`addChild`

Called when a child is added.

`endXMLStructure`

Called when an XML structure has ended.

`resolveExternalEntity`

Called when an external entity needs to be resolved.

`handleError`

Called when a parse error needs to be handled.

Discussion

This structure is passed to one of the `CFXMLParserCreate...` functions. Only the *`createXMLStructure`*, *`addChild`*, and *`endXMLStructure`* fields are required. Set the others to `NULL` if you don't wish to implement them.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserContext

Contains version information and function pointers to callbacks used when handling a program-defined context.

```
struct CFXMLParserContext {
    CFIndex version;
    void *info;
    CFXMLParserRetainCallBack retain;
    CFXMLParserReleaseCallBack release;
    CFXMLParserCopyDescriptionCallBack copyDescription;
};
typedef struct CFXMLParserContext CFXMLParserContext;
```

Fields

`version`
: Version number of this structure. Must be 0.

`info`
: An arbitrary program-defined value passed to all the callbacks in this structure and in the `CFXMLParserCallBacks` (page 18) structure.

`retain`
: A retain callback for your program-defined context data. Optional.

`release`
: A release callback for your program-defined context data. Optional.

`copyDescription`
: A copy description callback for your program-defined context data. Optional.

Discussion

You can associate a context with a parser when the parser is created. The context can be anything you wish and will be passed as a parameter to all of the XML parser callbacks.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

CFXMLParserRef

A reference to an XML parser object.

```
typedef struct __CFXMLParser *CFXMLParserRef;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFXMLParser.h`

Constants

Parser Status Codes

The various status and error flags that can be returned by the parser.

```
enum CFXMLParserStatusCode {
    kCFXMLStatusParseNotBegun = -2,
    kCFXMLStatusParseInProgress = -1,
    kCFXMLStatusParseSuccessful = 0,
    kCFXMLErrorUnexpectedEOF = 1,
    kCFXMLErrorUnknownEncoding = 2,
    kCFXMLErrorEncodingConversionFailure = 3,
    kCFXMLErrorMalformedProcessingInstruction = 4,
    kCFXMLErrorMalformedDTD = 5,
    kCFXMLErrorMalformedName = 6,
    kCFXMLErrorMalformedCDSect = 7,
    kCFXMLErrorMalformedCloseTag = 8,
    kCFXMLErrorMalformedStartTag = 9,
    kCFXMLErrorMalformedDocument = 10,
    kCFXMLErrorElementlessDocument = 11,
    kCFXMLErrorMalformedComment = 12,
    kCFXMLErrorMalformedCharacterReference = 13,
    kCFXMLErrorMalformedParsedCharacterData = 14,
    kCFXMLErrorNoData = 15
};
typedef enum CFXMLParserStatusCode CFXMLParserStatusCode;
```

Constants

`kCFXMLStatusParseNotBegun`

Indicates the parser has not begun.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLStatusParseInProgress`

Indicates the parser is in progress.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLStatusParseSuccessful`

Indicates the parser was successful.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorUnexpectedEOF`

Indicates an unexpected EOF occurred.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorUnknownEncoding`

Indicates an unknown encoding error.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorEncodingConversionFailure`
Indicates an encoding conversion error.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorMalformedProcessingInstruction`
Indicates a malformed processing instruction.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorMalformedDTD`
Indicates a malformed DTD.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorMalformedName`
Indicates a malformed name.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorMalformedCDSect`
Indicates a malformed CDATA section.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorMalformedCloseTag`
Indicates a malformed close tag.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorMalformedStartTag`
Indicates a malformed start tag.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorMalformedDocument`
Indicates a malformed document.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorElementlessDocument`
Indicates a document containing no elements.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorMalformedComment`
Indicates a malformed comment.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorMalformedCharacterReference`
Indicates a malformed character reference.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorMalformedParsedCharacterData`
Indicates malformed character data.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLErrorNoData`
Indicates a no data error.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

Discussion
Parser status is determined by calling the `CFXMLParserGetStatusCode` (page 11) function. The parser reports errors to your application by invoking the `CFXMLParserHandleErrorCallBack` (page 15) function.

Parsing Options

Options you can use to control the parser's treatment of an XML document.

```
enum CFXMLParserOptions {
    kCFXMLParserValidateDocument = (1 << 0),
    kCFXMLParserSkipMetaData = (1 << 1),
    kCFXMLParserReplacePhysicalEntities = (1 << 2),
    kCFXMLParserSkipWhitespace = (1 << 3),
    kCFXMLParserResolveExternalEntities = (1 << 4),
    kCFXMLParserAddImpliedAttributes = (1 << 5),
    kCFXMLParserAllOptions = 0x00FFFFFF,
    kCFXMLParserNoOptions = 0
};
typedef enum CFXMLParserOptions CFXMLParserOptions;
```

Constants
`kCFXMLParserValidateDocument`
Validates the document against its grammar from the DTD, reporting any errors. Currently not supported.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLParserSkipMetaData`
Silently skip over metadata constructs (the DTD and comments).

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLParserReplacePhysicalEntities`
Replaces declared entities like `<`. Note that other than the 5 predefined entities (`lt`, `gt`, `quot`, `amp`, `apos`), these must be defined in the DTD. Currently not supported.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLParserSkipWhitespace`
Skip over all whitespace that does not abut non-whitespace character data. In other words, given "`<foo> <bar> blah </bar></foo>`," the whitespace between foo's open tag and bar's open tag would be suppressed, but the whitespace around `blah` would be preserved.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLParserResolveExternalEntities`
Resolves all external entities.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLParserAddImpliedAttributes`
Where the DTD specifies implied attribute-value pairs for a particular element, add those pairs to any occurrences of the element in the element tree. Currently not supported.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLParserAllOptions`
Makes the parser do the most work, returning only the pure elementtree.

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

`kCFXMLParserNoOptions`
Leaves the XML as "intact" as possible (reports all structures; performs no replacements).

Available in Mac OS X v10.0 and later.

Declared in `CFXMLParser.h`.

Discussion
These are the various options you use to configure the parser. An option flag of 0 (`kCFXMLParserNoOptions` (page 23)) leaves the XML as "intact" as possible (reports all structures; performs no replacements). Hence, to make the parser do the most work, returning only the pure element tree, set the option flag to `kCFXMLParserAllOptions` (page 23).

Document Revision History

This table describes the changes to *CFXMLParser Reference*.

Date	Notes
2008-10-15	Corrected description of dataSource parameter for CFXMLParserCreateWithDataFromURL.
2006-02-07	Made formatting changes.
2003-01-01	First version of this document.

Document Revision History

Index

C

K

P