

---

# NSXMLDTD Class Reference

[Cocoa > Data Management](#)



2007-02-27



Apple Inc.  
© 2007 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY**

**DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

---

## **NSXMLDTD Class Reference 5**

---

Overview	5
Tasks	6
Initializing an NSXMLDTD Object	6
Managing DTD Identifiers	6
Manipulating Child Nodes	6
Getting DTD Nodes by Name	6
Class Methods	7
predefinedEntityDeclarationForName:	7
Instance Methods	7
addChild:	7
attributeDeclarationForName:elementName:	8
elementDeclarationForName:	8
entityDeclarationForName:	9
initWithContentsOfURL:options:error:	9
initWithData:options:error:	10
insertChild:atIndex:	11
insertChildren:atIndex:	11
notationDeclarationForName:	12
publicID	12
removeChildAtIndex:	12
replaceChildAtIndex:withNode:	13
setChildren:	13
setPublicID:	14
setSystemID:	14
systemID	15

---

## **Document Revision History 17**

---

## **Index 19**

---



# NSXMLDTD Class Reference

---

<b>Inherits from</b>	NSXMLNode : NSObject
<b>Conforms to</b>	NSCopying (NSXMLNode) NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/Foundation.framework
<b>Availability</b>	Available in Mac OS X v10.4 and later.
<b>Companion guide</b>	Tree-Based XML Programming Guide for Cocoa
<b>Declared in</b>	NSXMLDTD.h

## Overview

An instance of the `NSXMLDTD` class represents a Document Type Definition. It is held as a property of an `NSXMLDocument` instance, accessed through the `NSXMLDocument` method `DTD` (and set via `setDTD:`).

In the data model, an `NSXMLDTD` object is conceptually similar to namespace and attribute nodes: it is not considered to be a child of the `NSXMLDocument` object although it is closely associated with it. It is at the “root” of a shallow tree consisting primarily of nodes representing DTD declarations. Acceptable child nodes are instances of the `NSXMLDTDNode` class as well as `NSXMLNode` objects representing comment nodes and processing-instruction nodes.

You create an `NSXMLDTD` object in one of three ways:

- By processing an XML document with its own internal (in-line) DTD
- By process a standalone (external) DTD
- Programmatically

Once an `NSXMLDTD` instance is in place, you can add, remove, and change the `NSXMLDTDNode` objects representing various DTD declarations. When you write the document out as XML, the new or modified internal DTD is included (assuming you set the DTD in the `NSXMLDocument` instance). You may also programmatically create an external DTD and write that out to its own file.

## Tasks

### Initializing an NSXMLDTD Object

- [initWithContentsOfURL:options:error:](#) (page 9)  
Initializes and returns an NSXMLDTD object created from the DTD declarations in a URL-referenced source.
- [initWithData:options:error:](#) (page 10)  
Initializes and returns an NSXMLDTD object created from the DTD declarations encapsulated in an NSData object

### Managing DTD Identifiers

- [setPublicID:](#) (page 14)  
Sets the public identifier of the receiver.
- [publicID](#) (page 12)  
Returns the receiver's public identifier.
- [setSystemID:](#) (page 14)  
Sets the system identifier of the receiver.
- [systemID](#) (page 15)  
Returns the receiver's system identifier.

### Manipulating Child Nodes

- [addChild:](#) (page 7)  
Adds a child node to the end of the list of existing children.
- [insertChild:atIndex:](#) (page 11)  
Inserts a child node in the receiver's list of children at a specific location in the list.
- [insertChildren:atIndex:](#) (page 11)  
Inserts an array of child nodes at a specified location in the receiver's list of children.
- [removeChildAtIndex:](#) (page 12)  
Removes the child node at a particular location in the receiver's list of children.
- [replaceChildAtIndex:withNode:](#) (page 13)  
Replaces a child at a particular index with another child.
- [setChildren:](#) (page 13)  
Removes all existing children of the receiver and replaces them with an array of new child nodes.

### Getting DTD Nodes by Name

- + [predefinedEntityDeclarationForName:](#) (page 7)  
Returns a DTD node representing the predefined entity declaration with the specified name.

- [elementDeclarationForName:](#) (page 8)  
Returns the DTD node representing an element declaration for a specified element.
- [attributeDeclarationForName:elementName:](#) (page 8)  
Returns the DTD node representing an attribute-list declaration for a given attribute and its element.
- [entityDeclarationForName:](#) (page 9)  
Returns the DTD node representing the entity declaration for a specified entity.
- [notationDeclarationForName:](#) (page 12)  
Returns the DTD node representing the notation declaration identified by the specified notation name.

## Class Methods

### **predefinedEntityDeclarationForName:**

Returns a DTD node representing the predefined entity declaration with the specified name.

```
+ (NSXMLDTDNode *)predefinedEntityDeclarationForName:(NSString *)name
```

#### **Parameters**

*name*

A string identifying a predefined entity declaration.

#### **Return Value**

An autoreleased NSXMLDTDNode object, or nil if there is no match for *name*.

#### **Discussion**

The five predefined entity references (or character references) are “&lt;” (less-than sign), “&gt;” (greater-than sign), “&amp;” (ampersand), “&quot;” (quotation mark), and “&apos;” (apostrophe).

#### **Availability**

Available in Mac OS X v10.4 and later.

#### **See Also**

- [entityDeclarationForName:](#) (page 9)

#### **Declared In**

NSXMLDTD.h

## Instance Methods

### **addChild:**

Adds a child node to the end of the list of existing children.

```
- (void)addChild:(NSXMLNode *)child
```

**Parameters***child*

The node object to add to the existing children.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- [insertChildAtIndex:](#) (page 11)
- [insertChildrenAtIndex:](#) (page 11)
- [removeChildAtIndex:](#) (page 12)
- [replaceChildAtIndex:withNode:](#) (page 13)
- [setChildren:](#) (page 13)

**Declared In**

NSXMLDTD.h

**attributeDeclarationForName:elementName:**

Returns the DTD node representing an attribute-list declaration for a given attribute and its element.

```
- (NSXMLDTDNode *)attributeDeclarationForName:(NSString *)attrName
  elementName:(NSString *)elementName
```

**Parameters***attrName*

A string object identifying the name of an attribute.

*elementName*

A string object identifying the name of an element.

**Return Value**

An autoreleased `NSXMLDTDNode` object, or `nil` if there is no matching attribute-list declaration.

**Discussion**

For example, in the attribute-list declaration:

```
<!ATTLIST person idnum CDATA "0000">
```

“idnum” would correspond to *attrName* and “person” would correspond to *elementName*.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

NSXMLDTD.h

**elementDeclarationForName:**

Returns the DTD node representing an element declaration for a specified element.

```
- (NSXMLDTDNode *)elementDeclarationForName:(NSString *)elementName
```



**Parameters***elementName*

A string that is the name of an element.

**Return Value**An autoreleased `NSXMLDTDNode` object, or `nil` if there is no match.**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

NSXMLDTD.h

**entityDeclarationForName:**

Returns the DTD node representing the entity declaration for a specified entity.

- (NSXMLDTDNode \*)entityDeclarationForName:(NSString \*)entityName

**Parameters***entityName*

A string that is the name of an entity.

**Return Value**An autoreleased `NSXMLDTDNode` object, or `nil` if there is no match.**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

NSXMLDTD.h

**initWithContentsOfURL:options:error:**Initializes and returns an `NSXMLDTD` object created from the DTD declarations in a URL-referenced source.

- (id)initWithContentsOfURL:(NSURL \*)url options:(NSUInteger)mask error:(NSError \*\*)error

**Parameters***url*An `NSURL` object identifying a URL source.*mask*A bit mask specifying input options; bit-OR multiple options. The current valid options are `NSXMLNodePreserveWhitespace` and `NSXMLNodePreserveEntities`; these constants are described in the "Constants" section of the `NSXMLNode` reference.*error*On return, this parameter holds an `NSError` object describing any errors and warnings related to parsing and remote connection.**Return Value**An initialized `NSXMLDTD` object or `nil` if initialization fails because of parsing errors or other reasons.

**Discussion**

You use this method to create a stand-alone DTD which you can thereafter query and use for validation. You can associate the DTD created through this message with a document by sending `setDTD:` to an `NSXMLDocument` object.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- [initWithData:options:error:](#) (page 10)
- `validateAndReturnError:` (`NSXMLDocument`)

**Declared In**

`NSXMLDTD.h`

**initWithData:options:error:**

Initializes and returns an `NSXMLDTD` object created from the DTD declarations encapsulated in an `NSData` object

```
- (id) initWithData:(NSData *)data options:(NSUInteger)mask error:(NSError **)error
```

**Parameters**

*data*

A data object containing DTD declarations.

*mask*

A bit mask specifying input options; bit-OR multiple options. The current valid options are `NSXMLNodePreserveWhitespace` and `NSXMLNodePreserveEntities`; these constants are described in the "Constants" section of the `NSXMLNode` reference.

*error*

On return, this parameter holds an `NSError` object describing any errors and warnings related to parsing and remote connection.

**Return Value**

An initialized `NSXMLDTD` object or `nil` if initialization fails because of parsing errors or other reasons.

**Discussion**

This method is the designated initializer for the `NSXMLDTD` class. You use this method to create a stand-alone DTD which you can thereafter query and use for validation. You can associate the DTD created through this message with a document by sending `setDTD:` to an `NSXMLDocument` object.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- [initWithContentsOfURL:options:error:](#) (page 9)
- `validateAndReturnError:` (`NSXMLDocument`)

**Declared In**

`NSXMLDTD.h`

## insertChild:atIndex:

Inserts a child node in the receiver's list of children at a specific location in the list.

```
- (void)insertChild:(NSXMLNode *)child atIndex:(NSUInteger)index
```

### Parameters

*child*

An XML-node object that represents the child to insert.

*index*

An integer identifying the location in the receiver's list of children to insert *child*. The indices of subsequent children in the list are incremented by one.

### Availability

Available in Mac OS X v10.4 and later.

### See Also

- [addChild:](#) (page 7)
- [insertChildren:atIndex:](#) (page 11)
- [removeChildAtIndex:](#) (page 12)
- [replaceChildAtIndex:withNode:](#) (page 13)
- [setChildren:](#) (page 13)

### Declared In

NSXMLDTD.h

## insertChildren:atIndex:

Inserts an array of child nodes at a specified location in the receiver's list of children.

```
- (void)insertChildren:(NSArray *)children atIndex:(NSUInteger)index
```

### Parameters

*children*

An array of NSXMLNode objects to insert as children of the receiver.

*index*

An integer identifying the location in the list of current children to make the insertion. The indices of subsequent children in the list are incremented by the number of inserted children.

### Availability

Available in Mac OS X v10.4 and later.

### See Also

- [addChild:](#) (page 7)
- [insertChild:atIndex:](#) (page 11)
- [removeChildAtIndex:](#) (page 12)
- [replaceChildAtIndex:withNode:](#) (page 13)
- [setChildren:](#) (page 13)

### Declared In

NSXMLDTD.h

**notationDeclarationForName:**

Returns the DTD node representing the notation declaration identified by the specified notation name.

- (NSXMLDTDNode \*)notationDeclarationForName:(NSString \*)*notationName*

**Parameters**

*notationName*

A string that is the name of a notation.

**Return Value**

An autoreleased NSXMLDTDNode object, or nil if there is no match.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

NSXMLDTD.h

**publicID**

Returns the receiver's public identifier.

- (NSString \*)publicID

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- [setPublicID:](#) (page 14)

**Declared In**

NSXMLDTD.h

**removeChildAtIndex:**

Removes the child node at a particular location in the receiver's list of children.

- (void)removeChildAtIndex:(NSUInteger)*index*

**Parameters**

*index*

An integer identifying the child node to remove. The indices of subsequent children in the list are decremented by one.

**Discussion**

The removed child node is released.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- [addChild:](#) (page 7)

- [insertChild:atIndex:](#) (page 11)

- [insertChildren:atIndex:](#) (page 11)
- [replaceChildAtIndex:withNode:](#) (page 13)
- [setChildren:](#) (page 13)

**Declared In**

NSXMLDTD.h

**replaceChildAtIndex:withNode:**

Replaces a child at a particular index with another child.

```
- (void)replaceChildAtIndex:(NSUInteger) index withNode:(NSXMLNode *) node
```

**Parameters***index*

An integer identifying the position of a node in the receiver's list of child nodes.

*node*

An NSXMLNode object to replace the object at *index*.

**Discussion**

The replaced child node is released.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- [addChild:](#) (page 7)
- [insertChild:atIndex:](#) (page 11)
- [insertChildren:atIndex:](#) (page 11)
- [removeChildAtIndex:](#) (page 12)
- [setChildren:](#) (page 13)

**Declared In**

NSXMLDTD.h

**setChildren:**

Removes all existing children of the receiver and replaces them with an array of new child nodes.

```
- (void)setChildren:(NSArray *) children
```

**Parameters***children*

An array of NSXMLNode objects. To remove all existing children, pass in `nil`.

**Discussion**

Replaced or removed child nodes are released.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- [addChild:](#) (page 7)
- [insertChild:atIndex:](#) (page 11)
- [insertChildren:atIndex:](#) (page 11)
- [removeChildAtIndex:](#) (page 12)
- [replaceChildAtIndex:withNode:](#) (page 13)

**Declared In**

NSXMLDTD.h

**setPublicID:**

Sets the public identifier of the receiver.

```
- (void)setPublicID:(NSString *)publicID
```

**Parameters**

*publicID*

A string object specifying a public identifier.

**Discussion**

This identifier should be in the default catalog in `/etc/xml/catalog` or in a path specified by the environment variable `XML_CATALOG_FILES`. When the public ID is set the system ID must also be set.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- [publicID](#) (page 12)
- [setSystemID:](#) (page 14)

**Declared In**

NSXMLDTD.h

**setSystemID:**

Sets the system identifier of the receiver.

```
- (void)setSystemID:(NSString *)systemID
```

**Parameters**

*systemID*

A string object that encapsulates a URL locating a valid DTD.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- [systemID](#) (page 15)

**Declared In**

NSXMLDTD.h

## **systemID**

Returns the receiver's system identifier.

- (NSString \*)systemID

### **Availability**

Available in Mac OS X v10.4 and later.

### **See Also**

- [setSystemID:](#) (page 14)

### **Declared In**

NSXMLDTD.h





# Document Revision History

---

This table describes the changes to *NSXMLDTD Class Reference*.

Date	Notes
2007-02-27	Revised task headings.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

addChild: **instance method** [7](#)  
attributeDeclarationForName:elementName:  
    **instance method** [8](#)

## E

---

elementDeclarationForName: **instance method** [8](#)  
entityDeclarationForName: **instance method** [9](#)

## I

---

initWithContentsOfURL:options:error: **instance method** [9](#)  
initWithData:options:error: **instance method** [10](#)  
insertChild:atIndex: **instance method** [11](#)  
insertChildren:atIndex: **instance method** [11](#)

## N

---

notationDeclarationForName: **instance method** [12](#)

## P

---

predefinedEntityDeclarationForName: **class method** [7](#)  
publicID **instance method** [12](#)

## R

---

removeChildAtIndex: **instance method** [12](#)

replaceChildAtIndex:withNode: **instance method** [13](#)

## S

---

setChildren: **instance method** [13](#)  
setPublicID: **instance method** [14](#)  
setSystemID: **instance method** [14](#)  
systemID **instance method** [15](#)