



PowerBuilder Extension Reference

## **PowerBuilder® Classic**

12.5

DOCUMENT ID: DC33821-01-1250-02

LAST REVISED: November 2011

Copyright © 2011 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

# Contents

<b>About This Book .....</b>	<b>xiii</b>
<b>CHAPTER 1            PowerBuilder Extensions .....</b>	<b>1</b>
About PowerBuilder extensions .....	1
Using PowerBuilder extensions.....	2
Getting information about PowerBuilder extensions .....	4
<b>CHAPTER 2            EJB Client .....</b>	<b>5</b>
EJBConnection .....	5
ConnectToServer .....	6
CreateJavaInstance .....	7
DisconnectServer .....	8
GetEJBTransaction .....	8
Lookup.....	9
EJBTransaction .....	10
Begin .....	10
Commit .....	11
GetStatus .....	12
Rollback.....	13
SetRollbackOnly .....	14
SetTransactionTimeout .....	15
JavaVM .....	16
CreateJavaVM.....	16
CreateJavaInstance .....	19
DynamicCast .....	20
GetActualClass.....	24
GetInterfaces .....	25
GetJavaClasspath .....	26
GetJavaVMVersion .....	27
GetSuperClass .....	27
IsJavaVMLoaded.....	28
LoadMappingTable.....	29

<b>CHAPTER 3</b>	<b>Web Services Client .....</b>	<b>31</b>
	SoapConnection.....	31
	AddToBypassList .....	32
	CreateInstance .....	32
	DynamicCast .....	34
	RemoveAuthentication .....	35
	RemoveBypassList.....	35
	SetBasicAuthentication .....	36
	SetBypassProxyOnLocal.....	37
	SetClientCertificateFile.....	37
	SetOptions.....	38
	SetProxyServer .....	40
	SetProxyServerOptions .....	41
	SetSoapLogFile.....	42
	SetTimeout .....	42
	SetUseDefaultProxySetting .....	43
	UseConnectionCache .....	43
	UseIntegratedWindowsAuthentication .....	44
	SoapException .....	44
	SoapPBCookie.....	46
	GetComment .....	46
	GetCommentUri .....	47
	GetExpired .....	47
	GetExpires.....	47
	GetHttpOnly.....	48
	GetName .....	48
	GetSecure .....	48
	GetTimeStamp .....	49
	GetURI .....	49
	GetValue .....	49
	GetVersion .....	49
	SetComment .....	50
	SetCommentUri.....	50
	SetExpired.....	50
	SetExpires .....	51
	SetHttpOnly .....	51
	SetName .....	52
	SetSecure.....	52
	SetURI.....	53
	SetValue.....	53
	SetVersion.....	53
	UDDIProxy .....	54
	setInquiryUrl .....	54
	setOption .....	55

	findBusiness .....	55
	getBusinessDetail.....	56
	findService.....	57
<b>CHAPTER 4</b>	<b>PowerBuilder Document Object Model.....</b>	<b>59</b>
	About PBDOM.....	59
	Node trees.....	60
	XML parser.....	60
	Objects and methods .....	60
	PBDOM objects.....	61
<b>CHAPTER 5</b>	<b>PBDOM_ATTRIBUTE Class.....</b>	<b>63</b>
	PBDOM_ATTRIBUTE .....	63
	AddContent .....	65
	Clone .....	65
	Detach .....	67
	Equals .....	69
	GetBooleanValue .....	70
	GetContent .....	72
	GetDateValue .....	72
	GetDateTimeValue .....	73
	GetDoubleValue .....	74
	GetIntValue .....	74
	GetLongValue .....	74
	GetName .....	75
	GetNamespacePrefix .....	76
	GetNamespaceUri .....	76
	GetObjectClass .....	77
	GetObjectClassString.....	77
	GetOwnerDocumentObject .....	78
	GetOwnerElementObject .....	80
	GetQualifiedName .....	81
	GetRealValue .....	82
	GetText.....	82
	GetTextNormalize .....	84
	GetTextTrim .....	86
	GetTimeValue .....	88
	GetUIntValue .....	88
	GetULongValue .....	89
	HasChildren.....	89
	InsertContent .....	91
	IsAncestorObjectOf .....	93
	RemoveContent .....	93

	SetBooleanValue.....	96
	SetContent .....	96
	SetDateValue .....	100
	SetDateTimeValue .....	101
	SetDoubleValue .....	102
	SetIntValue.....	102
	SetLongValue.....	102
	SetName .....	103
	SetNamespace.....	106
	SetOwnerElementObject.....	109
	SetRealValue .....	111
	SetText.....	111
	SetTimeValue.....	112
	SetUIntValue .....	113
	SetUlongValue .....	113
<b>CHAPTER 6</b>	<b>PBDOM_BUILDER Class.....</b>	<b>115</b>
	PBDOM_BUILDER .....	115
	BuildFromDataStore.....	116
	BuildFromFile .....	117
	BuildFromString.....	120
	GetParseErrors .....	121
<b>CHAPTER 7</b>	<b>PBDOM_CDATA Class .....</b>	<b>123</b>
	PBDOM_CDATA.....	123
	Append.....	125
	Clone .....	125
	Detach .....	127
	Equals .....	128
	GetObjectClass .....	128
	GetObjectClassString.....	129
	GetOwnerDocumentObject .....	129
	GetParentObject.....	129
	GetText.....	130
	GetTextNormalize .....	130
	GetTextTrim .....	131
	SetParentObject .....	131
	SetText.....	132
<b>CHAPTER 8</b>	<b>PBDOM_ENTITYREFERENCE Class.....</b>	<b>133</b>
	PBDOM_ENTITYREFERENCE .....	133
	Clone.....	134

Detach .....	136
Equals .....	136
GetName .....	137
GetObjectClass .....	137
GetObjectClassString .....	137
GetOwnerDocumentObject .....	138
GetParentObject .....	138
SetName .....	139
SetParentObject .....	139

**CHAPTER 9**

<b>PBDOM_CHARACTERDATA Class .....</b>	<b>141</b>
PBDOM_CHARACTERDATA .....	141
Append .....	142
Append Syntax 1 .....	143
Append Syntax 2 .....	146
Clone .....	147
Detach .....	149
Equals .....	151
GetOwnerDocumentObject .....	152
GetName .....	155
GetObjectClass .....	156
GetObjectClassString .....	156
GetParentObject .....	158
GetText .....	161
GetTextNormalize .....	162
GetTextTrim .....	166
HasChildren .....	170
IsAncestorObjectOf .....	171
SetParentObject .....	171
SetText .....	174

**CHAPTER 10**

<b>PBDOM_COMMENT Class .....</b>	<b>175</b>
PBDOM_COMMENT .....	175
Append .....	176
Append Syntax 1 .....	177
Append Syntax 2 .....	177
Clone .....	178
Detach .....	180
Equals .....	181
GetObjectClass .....	181
GetObjectClassString .....	182
GetOwnerDocumentObject .....	182
GetParentObject .....	182

GetText.....	183
GetTextNormalize .....	183
GetTextTrim .....	184
SetParentObject .....	184
SetText .....	185

**CHAPTER 11      PBDOM\_DOCTYPE Class ..... 187**

PBDOM_DOCTYPE.....	187
Clone .....	188
Detach .....	188
Equals .....	189
GetInternalSubset .....	189
GetName .....	189
GetObjectClass .....	190
GetObjectClassString.....	190
GetOwnerDocumentObject .....	191
GetParentObject.....	191
GetPublicID .....	191
GetSystemID .....	192
SetDocument.....	193
SetInternalSubset.....	193
SetName .....	194
SetParentObject .....	195
SetPublicID.....	196
SetSystemID .....	197

**CHAPTER 12      PBDOM\_DOCUMENT Class ..... 199**

PBDOM_DOCUMENT .....	199
AddContent .....	200
Clone .....	203
DetachRootElement .....	203
Equals .....	204
GetContent .....	204
GetDocType .....	206
GetElementsByTagName.....	206
GetObjectClass .....	207
GetObjectClassString.....	208
GetRootElement.....	208
HasChildren.....	209
HasRootElement .....	209
InsertContent.....	209
IsAncestorObjectOf .....	212
NewDocument.....	212



NewDocument Syntax 1 .....	213
NewDocument Syntax 2 .....	213
RemoveContent .....	216
SaveDocument .....	217
SaveDocumentIntoString .....	218
SetContent .....	219
SetDocType .....	220
SetRootElement .....	221

**CHAPTER 13**

<b>PBDOM_ELEMENT Class .....</b>	<b>223</b>
PBDOM_ELEMENT .....	223
AddContent .....	224
AddContent Syntax 1 .....	225
AddContent Syntax 2 .....	227
AddNamespaceDeclaration .....	228
Clone .....	230
Detach .....	231
Equals .....	231
GetAttribute .....	233
GetAttribute Syntax 1 .....	233
GetAttribute Syntax 2 .....	234
GetAttributes .....	235
GetAttributeValue .....	235
GetAttributeValue Syntax 1 .....	236
GetAttributeValue Syntax 2 .....	237
GetAttributeValue Syntax 3 .....	238
GetAttributeValue Syntax 4 .....	239
GetChildElement .....	239
GetChildElement Syntax 1 .....	240
GetChildElement Syntax 2 .....	240
GetChildElements .....	241
GetChildElements Syntax 1 .....	242
GetChildElements Syntax 2 .....	243
GetChildElements Syntax 3 .....	244
GetContent .....	245
GetName .....	246
GetNamespacePrefix .....	247
GetNamespaceUri .....	247
GetObjectClass .....	248
GetObjectClassString .....	249
GetOwnerDocumentObject .....	249
GetParentObject .....	250
GetQualifiedName .....	251
GetText .....	251

GetTextNormalize .....	252
GetTextTrim .....	252
HasAttributes .....	253
HasChildElements .....	254
HasChildren .....	255
InsertContent .....	256
IsAncestorObjectOf .....	258
IsRootElement .....	258
RemoveAttribute .....	259
RemoveAttribute Syntax 1 .....	259
RemoveAttribute Syntax 2 .....	260
RemoveAttribute Syntax 3 .....	260
RemoveChildElement .....	261
RemoveChildElement Syntax 1 .....	261
RemoveChildElement Syntax 2 .....	262
RemoveChildElements .....	263
RemoveChildElements Syntax 1 .....	263
RemoveChildElements Syntax 2 .....	264
RemoveChildElements Syntax 3 .....	264
RemoveContent .....	265
RemoveNamespaceDeclaration .....	266
SetAttribute .....	267
SetAttribute Syntax 1 .....	268
SetAttribute Syntax 2 .....	271
SetAttribute Syntax 3 .....	273
SetAttributes .....	276
SetContent .....	279
SetDocument .....	282
SetName .....	282
SetNamespace .....	283
SetParentObject .....	284
SetText .....	285

<b>CHAPTER 14</b>	<b>PBDOM_EXCEPTION Class .....</b>	<b>287</b>
	PBDOM exceptions .....	287
	PBDOM exception descriptions .....	288
	PBDOM_EXCEPTION .....	293
	GetExceptionCode .....	293

<b>CHAPTER 15</b>	<b>PBDOM_OBJECT Class .....</b>	<b>295</b>
	PBDOM_OBJECT .....	295
	AddContent .....	296
	Clone .....	296

Detach .....	297
Equals .....	298
GetContent .....	298
GetName .....	299
GetObjectClass .....	300
GetObjectClassString .....	301
GetOwnerDocumentObject .....	302
GetParentObject .....	303
GetText .....	304
GetTextNormalize .....	306
GetTextTrim .....	308
HasChildren .....	311
InsertContent .....	312
IsAncestorObjectOf .....	313
RemoveContent .....	315
SetContent .....	316
SetName .....	317
SetParentObject .....	318

**CHAPTER 16 PBDOM\_PROCESSINGINSTRUCTION Class..... 321**

PBDOM_PROCESSINGINSTRUCTION .....	321
Clone .....	322
Detach .....	323
Equals .....	323
GetData .....	324
GetName .....	324
GetNames .....	325
GetObjectClass .....	325
GetObjectClassString .....	326
GetOwnerDocumentObject .....	326
GetParentObject .....	326
GetTarget .....	327
GetText .....	327
GetTextNormalize .....	328
GetTextTrim .....	328
GetValue .....	329
RemoveValue .....	329
SetData .....	330
SetName .....	331
SetParentObject .....	332
SetValue .....	333

**CHAPTER 17 PBDOM\_TEXT Class..... 335**

	PBDOM_TEXT .....	335
	Append .....	336
	Append Syntax 1 .....	336
	Append Syntax 2 .....	337
	Clone .....	337
	Detach .....	339
	Equals .....	340
	GetObjectClass .....	340
	GetObjectClassString .....	341
	GetOwnerDocumentObject .....	341
	GetParentObject.....	341
	GetText.....	342
	GetTextNormalize .....	342
	GetTextTrim .....	343
	SetParentObject .....	343
	SetText .....	344
<b>CHAPTER 18</b>	<b>PBDOM Summary .....</b>	<b>345</b>
	Summary of PBDOM classes and methods.....	345
<b>Index .....</b>		<b>349</b>

# About This Book

<b>Audience</b>	This book is for programmers who build applications that use built-in PowerBuilder® extensions.
<b>How to use this book</b>	<p>This book describes syntax and usage information for built-in extensions to the PowerScript® language:</p> <ul style="list-style-type: none"><li>• Chapter 1 presents an overview of PowerBuilder extensions and how you use them in a PowerScript application.</li><li>• Chapter 2 describes the objects used to build clients for Enterprise JavaBeans components.</li><li>• Chapter 3 describes the objects used to build SOAP clients for Web services.</li><li>• Chapter 4 presents an overview of the PowerBuilder Document Object Model (PBDOM).</li><li>• Chapters 5 through 17 describe each of the objects that make up the PBDOM.</li><li>• Chapter 18 provides a quick-reference list of PBDOM methods.</li></ul>
<b>Related documents</b>	<p>Step-by-step instructions on building applications that use each of the built-in extensions are in <i>Application Techniques</i>.</p> <p>For a complete list of PowerBuilder documentation, see the preface of <i>PowerBuilder Getting Started</i>.</p>
<b>Other sources of information</b>	<p>Use the Sybase® Getting Started CD and the Sybase Product Documentation Web site to learn more about your product:</p> <ul style="list-style-type: none"><li>• The Getting Started CD contains release bulletins and installation guides in PDF format. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.</li></ul>

- The Sybase Product Documentation Web site is accessible using a standard Web browser. In addition to product documentation, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

To access the Sybase Product Documentation Web site, go to Product Documentation at <http://www.sybase.com/support/manuals/>.

## Conventions

The formatting conventions used in this manual are:

Formatting example	Indicates
Retrieve and Update	When used in descriptive text, this font indicates: <ul style="list-style-type: none"> <li>• Command, function, and method names</li> <li>• Keywords such as true, false, and null</li> <li>• Datatypes such as integer and char</li> <li>• Database column names such as emp_id and f_name</li> <li>• User-defined objects such as dw_emp or w_main</li> </ul>
<i>variable or file name</i>	When used in descriptive text and syntax descriptions, oblique font indicates: <ul style="list-style-type: none"> <li>• Variables, such as <i>myCounter</i></li> <li>• Parts of input text that must be substituted, such as <i>pblname.pbd</i></li> <li>• File and path names</li> </ul>
File>Save	Menu names and menu items are displayed in plain text. The greater than symbol (>) shows you how to navigate menu selections. For example, File>Save indicates “select Save from the File menu.”
dw_1.Update()	Monospace font indicates: <ul style="list-style-type: none"> <li>• Information that you enter in a dialog box or on a command line</li> <li>• Sample script fragments</li> <li>• Sample output fragments</li> </ul>

## If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the documentation or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

About this chapter

This chapter provides a brief introduction to PowerBuilder extensions.

Contents

Topic	Page
About PowerBuilder extensions	1
Using PowerBuilder extensions	2
Getting information about PowerBuilder extensions	4

## About PowerBuilder extensions

The PowerBuilder Native Interface (PBNI) is a standard programming interface that enables developers to extend the functionality of PowerBuilder. A PowerBuilder extension can be provided by Sybase, by you, or by a third party.

This book provides reference information for extensions provided by Sybase. In PowerBuilder, these extensions are for Enterprise JavaBeans clients, the PowerBuilder Document Object Model (PBDOM), SOAP clients for Web services, and the UDDIProxy class. Embedding these features in separate extension files instead of adding them to the core PowerBuilder runtime files helps keep the footprint of deployed applications as small as possible.

For information about building your own extensions, see the *PowerBuilder Native Interface Programmers Guide and Reference*.

To find out about extensions provided by other developers, check the PBNI section of the PowerBuilder CodeXchange site at <http://powerbuilder.codexchange.sybase.com>.

---

**Use with .NET targets**

You can use the built-in Web services client extension (*pbwsclient125.pbx*) in applications that you plan to deploy to .NET as PowerBuilder .NET Windows Forms applications. You *cannot* use any other PBNI extensions in a .NET target.

---

## Using PowerBuilder extensions

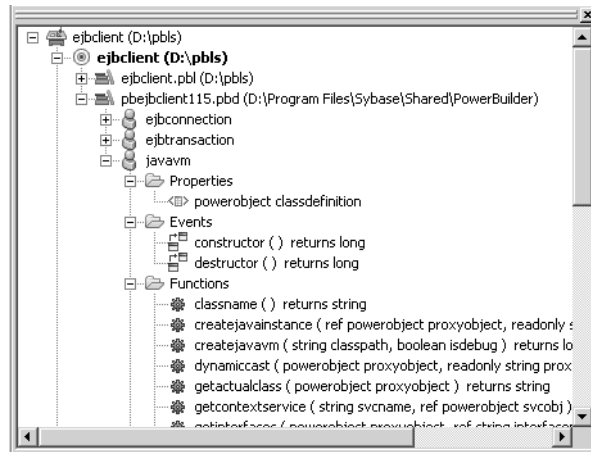
Every PowerBuilder extension requires a compiled C++ shared library, usually with the extension *.pbx* (for *PowerBuilder eXtension*). The C++ shared library file contains classes and methods that you use in your PowerScript target in the same way that you use PowerBuilder system objects or user objects.

To use the shared library in PowerBuilder, you place it in PowerBuilder's search path. In the System Tree, right-click a library in your PowerScript target, select Import PB Extension from the pop-up menu, navigate to the shared library, and select Open. This imports the definitions in the PBX into the library in your target. You can alternatively add the associated PBD file to the target's library search path. The PBD acts as a wrapper for the C++ shared library, enabling PowerBuilder to display the objects and methods it contains.

When you deploy an application that uses an extension, the C++ shared library must be deployed in the application's search path with the other PowerBuilder runtime files.



When you import an extension into a PowerScript target, the classes it contains display in the System Tree as user objects. You can expand the objects to display properties, events, and functions. You can inherit from extension objects and use drag-and-drop programming from the inherited objects in the System Tree as you do for other user objects.



#### Using nonvisual classes

In PowerScript, use the classes in a nonvisual extension just as you would a custom class user object: declare an instance of the object, use the CREATE statement to create the instance, invoke the object's functions, and destroy the instance when you have finished with it. You can inherit from the native classes if you want to add functions or events to the class.

At runtime, instances of the native class are created as normal PowerBuilder objects.

#### Using visual classes

You do not need to declare an instance of a visual class or use the CREATE statement to create an instance. To use a visual extension, select File>Inherit from the PowerBuilder menu, select the PBL or PBD that contains the visual class in the Libraries list in the Inherit from Object dialog box, select the visual class, and click OK.

In the User Object painter, size the visual object and make any other changes you need, then save the object. You can then drag the new user object from the System Tree directly onto a window or onto another visual control, such as a tab control, and use it like any other visual user object.

**PBXRuntimeError** PowerBuilder extensions can throw a special exception, `PBXRuntimeError`, that inherits from the `PowerBuilder RuntimeError` exception. If you use an extension in a PowerBuilder application, you should include try-catch blocks for this exception and report any occurrences to the provider of the extension. This exception is usually caused by programming errors within the extension.

## Getting information about PowerBuilder extensions

**Online Help** The classes and methods in the extensions provided by Sybase are described in this book, which is available in the PowerBuilder online Help. For PBDOM, each class is described in a separate chapter.

You can open the Help in several ways:

- Select *PowerBuilder Extension Reference* from the PowerBuilder Help Contents tab page.
- Double-click the file name (*pbextref125.chm*) in the *C:\Documents and Settings\All Users\Documents\Sybase\PowerBuilder125\Help* directory on Windows XP and Windows 2003, or in the *C:\Users\Public\Documents\Sybase\PowerBuilder 125\Help* directory on Windows Vista and Windows 2008.
- Type a method name in the Script view, then press Shift+F1 to open the PowerBuilder Help Index tab with the focus on the first index entry for that method name. The name of the extension class displays in parentheses after the method name on the Index tab page, and it displays above the name of the method when you open the Help for the method.

---

### **If a PowerScript function description displays**

If there is a PowerScript function with the same name, the Help opens automatically to display the PowerScript function. Click the Help Topics button in the Help window to display the Index tab so that you can select the extension method.

---

**HTML books** For information about using the extensions provided by Sybase in your applications, see *Application Techniques* in the compiled HTML Help.

**Third-party extensions** The PowerBuilder Help and documentation do not provide any specific information for extensions developed by third parties. To find out how to use a third-party extension, see the documentation provided with the extension.

**About this chapter**

This chapter describes the PowerBuilder extension classes that are used to connect to an application server and employ Enterprise JavaBeans (EJB) components. For more information about building clients for EJB components, see *Application Techniques*.

**Contents**

<b>Topic</b>	<b>Page</b>
EJBConnection	5
EJBTransaction	10
JavaVM	16

## EJBConnection

**Description**

The EJBConnection class connects to an EJB server and locates an EJB.

**Methods**

EJBConnection has five member functions:

- ConnectToServer
- CreateJavaInstance
- DisconnectServer
- GetEJBTransaction
- Lookup

## ConnectToServer

**Description** Connects a client application to an EJB server. The client application must call ConnectToServer before it can use a remote object on the server.

**Syntax** `connection.ConnectToServer ( string properties[ ] )`

<b>Argument</b>	<b>Description</b>
<code>connection</code>	The name of the EJBConnection object you want to use to establish the connection
<code>properties[ ]</code>	A string array used to pass name/value pairs that specify how the connection will be established

**Return value** None

**Throws** NamingException

**Examples** In this example, the client application connects to a WebLogic server application using the Connection object called conn:

```
ejbconnection conn
helloejbhome hellohome
helloejb hello
string properties[ ]
string msg
```

```
// Type each of the following statements on one line
properties[1]="javax.naming.Context.INITIAL_CONTEXT_FACTORY=weblogic.jndi.
WLInitialContextFactory"
properties[2]="javax.naming.Context.PROVIDER_URL=t3://svr1:7001"
properties[3]="javax.naming.Context.SECURITY_PRINCIPAL=myid"
properties[4]="javax.naming.Context.SECURITY_CREDENTIALS=mypass"
```

```
conn = create ejbconnection
TRY
    conn.connectToServer(properties)
CATCH (remoteexception re)
    messagebox("remoteexception", re.GetMessage())
CATCH (createexception ce)
    messagebox("createexception", ce.GetMessage())
END TRY
```

**Usage** You must provide ConnectToServer with a set of properties that specify how the connection will be established. Before calling ConnectToServer, declare a string array variable and assign values for the javax.naming.Context constants shown in the following table to the elements of the array.

<b>javax.naming.context constant</b>	<b>Value</b>
INITIAL_CONTEXT_FACTORY	Server dependent. For example: <b>EAServer:</b> com.sybase.ejb.InitialContextFactory <b>WebLogic:</b> weblogic.jndi.WLInitialContextFactory <b>WebSphere:</b> com.ibm.websphere.naming.WsnInitialContextFactory
PROVIDER_URL	URL for the Server's port. For example: <code>iiop://myserver:9000</code>
SECURITY_PRINCIPAL	User name required for access to the server.
SECURITY_CREDENTIALS	Credentials associated with the user name, typically a password.

See also                      CreateJavaInstance  
                                    Lookup

## CreateJavaInstance

Description                      Creates an instance of a Java object from a proxy name.

### Deprecated function

This function is maintained for backward compatibility. You should use the CreateJavaInstance function on the JavaVM object for new development. You do not need to be connected to a server to create a local instance of a Java object.

Syntax                              *connection*.CreateJavaInstance (powerobject *proxyobject*, string *proxyname*)

<b>Argument</b>	<b>Description</b>
<i>connection</i>	The name of the EJBConnection object used to establish the connection.
<i>proxyobject</i>	PowerObject into which the function places a reference to the object specified by proxyname. This argument is passed by reference.
<i>proxyname</i>	The name of the proxy object for the local Java class.

Return value                      Long. Returns 0 for success and one of the following values for failure:

- 1 Failed to create Java class.
- 2 Invalid proxy name.
- 3 Failed to create proxy object.

See also                              CreateJavaInstance

## DisconnectServer

Description Disconnects a client application from an EJB server application.

Syntax `connection.DisconnectServer ( )`

Argument	Description
<code>connection</code>	The name of the EJBConnection object used to establish the connection you want to sever

Return value None

Throws NamingException

Examples In this example, the client application disconnects from the server application using the EJBConnection object myconnect:

```
myconnect.DisconnectServer ( )
```

See also ConnectToServer

## GetEJBTransaction

Description Returns a reference to the EJBTransaction object associated with the client.

Syntax `connection.GetEJBTransaction ( )`

Argument	Description
<code>connection</code>	The name of the EJBConnection object used to establish the connection

Return value EJBTransaction

Examples This example shows the use of GetEJBTransaction to return a reference to the EJBTransaction object so that you can control transactions from the client:

```
// Instance variables:  
// EJBConnection myconnect  
EJBTransaction mytrans  
long ll_status  
  
mytrans = myconnect.GetEJBTransaction ( )  
ll_status = mytrans.GetStatus ( )
```

Usage The PowerBuilder client can control the transaction demarcation of EJBs. After a transaction has been started with the EJBTransaction Begin method, GetEJBTransaction can be used to return the name of the transaction.

See also Begin, Commit, GetStatus, Rollback, SetRollbackOnly, SetTransactionTimeout

## Lookup

**Description** Allows a PowerBuilder client to obtain the home interface of an EJB component in an application server in order to create an instance of the component.

**Syntax** `connection.Lookup (string proxyname, string JNDIname, string homeinterfacename )`

Argument	Description
<i>connection</i>	The name of the EJBConnection object used to establish the connection
<i>proxyname</i>	The name of the proxy object for the EJB component
<i>JNDIname</i>	The JNDI name of the EJB component
<i>homeinterfacename</i>	The fully-qualified class name of the EJB home interface

**Return value** Powerobject. A proxy object for the home interface of the EJB.

**Throws** NamingException

**Examples** The following example uses lookup to locate the home interface of the Multiply session EJB in the Java package `com.xyz.math`. The example assumes the connection to the EJB server has already been established:

```
// Instance variable:
// EJBConnection myconnect
Multiply myMultiply
MultiplyHome myMultiplyHome
long ll_product

TRY
    myMultiplyHome = myconnect.lookup("MultiplyHome", &
        "Math/Multiply", "com.xyz.math.MultiplyHome")
    myMultiply = myMultiplyHome.create()
    ll_product = myMultiply.multiply(1234, 4567)
catch (remoteexception re)
    messagebox("remoteexception", re.GetMessage())
catch (createexception ce)
    messagebox("createexception", ce.GetMessage())
CATCH (exception e)
    MessageBox("Exception", e.getmessage())
END TRY
```

The style used for the JNDI name depends on the EJB server.

**See also** ConnectToServer

## EJBTransaction

**Description** The EJB transaction class enables PowerBuilder clients to control a transaction on an EJB server. EJBTransaction maps closely to the `javax.transaction.UserTransaction` interface.

**Methods** EJBTransaction has six member functions:

- Begin
- Commit
- GetStatus
- Rollback
- SetRollbackOnly
- SetTransactionTimeout

### Begin

**Description** Creates a new transaction and associates it with the current thread.

**Syntax** `ejbtrans.Begin ( )`

Argument	Description
<code>ejbtrans</code>	The name of an EJBTransaction object

**Return value** None

**Examples** The following example shows the use of `begin` to create a transaction from a client:

```
EJBTransaction trans
EJBConnection conn
string properties[ ]

// set properties
.....
conn = create ejbconnection
TRY
    conn.connectToServer(properties)
    trans = conn.GetEjbTransaction
    trans.begin()
CATCH (exception e)
    messagebox("exception", e.getmessage())
END TRY
```

**See also** `Commit`, `GetStatus`, `GetEJBTransaction` (EJBConnection class), `Rollback`, `SetRollbackOnly`, `SetTransactionTimeout`



## Commit

Description Declares that the calling thread transaction should be committed.

Syntax `ejbtrans.Commit ( )`

Argument	Description
<code>ejbtrans</code>	The name of an EJBTransaction object

Return value None

Examples In this example, the client calls the `dopayroll` method on the `CmpnyAcct` EJB component, which processes a company payroll. If the company has sufficient funds to meet the payroll, the client commits the transaction. Otherwise, an exception is thrown and the client rolls back the transaction:

```
// Instance variables:
// EJBTransaction trans
// EJBConnection conn
// CmpnyAcctHome AcctHome
// CmpnyAcct Acct

TRY
    trans.begin()
    AcctHome = conn.lookup("CmpnyAcctHome",
        "Sample/CmpnyAcct", "sample.CmpnyAcctHome")
    Acct = AcctHome.create()
    Acct.dopayroll()
    trans.commit()
CATCH (remoteexception re)
    messagebox("remoteexception", re.GetMessage())
CATCH (createexception ce)
    messagebox("createexception", ce.GetMessage())
CATCH (exception e1)
    MessageBox ("exception", e1.getmessage() )
    TRY
        trans.rollback();
    CATCH (exception e2)
        MessageBox ("exception", e2.getmessage() )
END TRY
END TRY
```

Usage The `Commit` method completes the transaction associated with the calling thread. The transaction is not completed if any other participants in the transaction vote to roll back the transaction.

See also `Commit`, `GetStatus`, `GetEJBTransaction` (`EJBConnection` class), `Rollback`, `SetRollbackOnly`, `SetTransactionTimeout`

## GetStatus

Description Returns the status of the EJB transaction associated with the client.

Syntax `ejbtrans.GetStatus ( )`

Argument	Description
<code>ejbtrans</code>	The name of an EJBTransaction object

Return value A long value representing the transaction status

Possible values are:

- 1 Status active
- 2 Status marked rollback
- 3 Status prepared
- 4 Status committed
- 5 Status rolled back
- 6 Status unknown
- 7 Status no transaction
- 8 Status preparing
- 9 Status committing
- 10 Status rolling back

Examples This example shows the use of `GetStatus` to obtain the state of the current transaction:

```
// Instance variables:  
// EJBConnection myconnect  
EJBTransaction mytrans  
long ll_status  
  
mytrans = myconnect.GetEJBTransaction()  
ll_status = mytrans.GetStatus ()
```

Usage The `GetStatus` method can be used to determine the current status of a transaction by the client that initiated the transaction using the `Begin` method.

See also `Begin`  
`Commit`  
`GetEJBTransaction` (EJBConnection class)  
`Rollback`  
`SetRollbackOnly`  
`SetTransactionTimeout`

## Rollback

Description Rolls back the transaction associated with the calling thread.

Syntax `ejbtrans.Rollback ( )`

Argument	Description
<code>ejbtrans</code>	The name of an EJBTransaction object

Return value None

Examples This example shows the use of Rollback to roll back a transaction when an update does not succeed:

```
// Instance variables:
// EJBTransaction trans
//
TRY
    trans.begin()
    Acct.updateChecking(amount)
    trans.commit()
CATCH (exception e1)
    TRY
        trans.rollback()
    CATCH (exception e2)
        MessageBox("Rollback failed", e2.getMessage())
    END TRY
    MessageBox("Transaction failed", e1.getMessage())
END TRY
```

See also

- Begin
- Commit
- GetStatus
- GetEJBTransaction (EJBConnection class)
- SetRollbackOnly
- SetTransactionTimeout

## SetRollbackOnly

**Description**                      Modifies a transaction associated with a calling thread so that the only possible outcome is to roll back the transaction.

**Syntax**                              `ejbtrans.SetRollbackOnly ( )`

<b>Argument</b>	<b>Description</b>
<code>ejbtrans</code>	The name of an EJBTransaction object

**Return value**                      None

**Examples**                              In this example, a participant in a transaction has determined that it should be rolled back. The participant gets a reference to the current transaction and votes to roll back the transaction:

```
// Instance variables:  
// EJBConnection conn  
// EJBTransaction trans  
  
trans = conn.GetEJBTransaction()  
trans.SetRollbackOnly()
```

**Usage**                                      Rollback is typically called by the originator of the transaction, but another participant in a transaction can call `SetRollbackOnly` to vote that the transaction should be rolled back.

**See also**                                      `Begin`  
`Commit`  
`GetStatus`  
`GetEJBTransaction` (EJBConnection class)  
`Rollback`  
`SetTransactionTimeout`

## SetTransactionTimeout

**Description** Sets the timeout value for subsequent transactions. The transaction is rolled back if it does not complete before the timeout expires.

**Syntax** `ejbtrans.SetTransactionTimeout (long seconds)`

Argument	Description
<i>ejbtrans</i>	The name of an EJBTransaction object
<i>seconds</i>	A long that specifies the number of seconds that elapse before a transaction is rolled back

**Return value** None

**Examples** This example shows the use of SetTransactionTimeout to set the timeout period to five minutes:

```
// Instance variables:
// EJBConnection conn
// EJBTransaction trans

TRY
    trans.SetTransactionTimeout(300)
    trans.begin()
CATCH (exception e)
    MessageBox("Exception", e.getMessage())
END TRY
```

**Usage** The SetTransactionTimeout method specifies the number of seconds that can elapse before a transaction is rolled back. The timeout period applies to transactions created by subsequent invocations of Begin. If *seconds* is 0, no timeout period is in effect.

**See also**

- Begin
- Commit
- GetStatus
- GetEJBTransaction (EJBConnection class)
- Rollback
- SetRollbackOnly

## JavaVM

**Description** The JavaVM class provides a method for loading and initializing a Java VM. It also provides methods for obtaining the version of the Java VM and the classpath it is using, to get the class name, super class name, and interface name of a Java class from the PowerBuilder proxy for that class, and to down cast a PowerBuilder proxy to another PowerBuilder proxy.

**Methods** JavaVM has the following member functions:

- CreateJavaVM
- CreateJavaInstance
- DynamicCast
- GetActualClass
- GetInterfaces
- GetJavaClasspath
- GetJavaVMVersion
- GetSuperClass
- IsJavaVMLoaded
- LoadMappingTable

## CreateJavaVM

**Description** Loads and initializes a Java VM or attaches an existing Java VM to the current process.

**Syntax** `javavm.createJavaVM(string classpath, boolean isdebug)`

Argument	Description
<i>javavm</i>	An instance of the JavaVM class
<i>classpath</i>	A string specifying the classpath that contains files required by the EJB server, such as the path to the EJB classes
<i>isdebug</i>	A boolean that determines whether debug information is saved to a file called <i>VM.out</i> in the directory where the current application is located

**Return value** Integer. Returns one of the following integer values:

- 1** Success. The Java VM had already been loaded and was attached to the current process.
- 0** Success. The Java VM was loaded and initialized and attached to the current process.

- 1 Failure. The Java VM was not loaded, possibly because *jvm.dll* was not found in the classpath.
- 2 Failure. The *pbejbclient125.jar* file was not found.

**Examples**

This example shows how `createJavaVM` might be used with a connection to `EAServer`:

```

JavaVM l_jvm
EJBConnection l_ejbconn
java_integer val
long rc
l_jvm = CREATE JavaVM
l_EJBConn = CREATE EJBConnection

TRY
  IF l_jvm.createJavaVM("", false) >= 0 THEN
    string ls_props[]
    ls_props[1] = "javax.naming.Context.INITIAL_CONTEXT_FACTORY=
      com.sybase.ejb.InitialContextFactory"
    ls_props[2] = "javax.naming.Context.PROVIDER_URL=iiop://localhost:9000"
    ls_props[3] = "javax.naming.Context.SECURITY_PRINCIPAL=jagadmin"
    ls_props[4] = "javax.naming.Context.SECURITY_CREDENTIALS=jagadmin"
    l_EJBConn.connectToServer(ls_props)
    l_EJBConn.createJavaInstance(val, "java_integer")
    val.java_integer(17)
    MessageBox("The value is", val.IntValue())
  ELSE
    MessageBox("createJavaVM", "Failed", StopSign!)
  END IF
CATCH (Throwable g)
  MessageBox("Exception in createJavaInstance", g.getMessage())
END TRY

```

**Usage**

The *isdebug* argument is used to record information about the Java VM, including class loads, in the file *VM.out* in the directory where the current application is located.

The *classpath* argument must include the classes and JAR files required by the server, if they are not already listed in the classpath used by the Java VM.

---

**Classpath argument has no effect if the JVM is already running**

Files and directories passed only in the *classpath* argument are not available to the Java VM if it has already been started by another process. In the development environment, you can check whether the Java VM is running and, if so, which classpath it is using, on the Java page of the System Options dialog box. At runtime, you can use the `IsJavaVMLoaded` method to determine whether the Java VM is already running, and the `GetJavaClasspath` method to find the classpath.

---

In the development environment, the classpath used by the Java VM is constructed by concatenating these paths:

- A classpath added programmatically when the JVM is started. For example, the classpath you pass to this method.
- The PowerBuilder runtime static registry classpath. This path is built into the *pbjvm125.dll* and contains classes required at runtime for features such as PDF generation and EJB clients.
- The PowerBuilder system classpath. This path resides in a Windows registry key installed when you install PowerBuilder. It contains classes required at design time for Java-related PowerBuilder features.
- The PowerBuilder user classpath. This is the path that you specify on the Java page of the System Options dialog box.
- The system CLASSPATH environment variable.
- The current directory.

The JVM uses the following classpath at runtime:

- A classpath added programmatically when the JVM is started
- The PowerBuilder runtime static registry classpath
- The system CLASSPATH environment variable
- The current directory

See also

`ConnectToServer`  
`GetJavaClasspath`  
`GetJavaVMVersion`  
`IsJavaVMLoaded`



## CreateJavaInstance

Description Creates an instance of a Java object from a proxy name.

Syntax *javavm.CreateJavaInstance (powerobject proxyobject, string proxyname )*

Argument	Description
<i>javavm</i>	An instance of the JavaVM class.
<i>proxyobject</i>	PowerObject into which the function places a reference to the object specified by proxyname. This argument is passed by reference.
<i>proxyname</i>	The name of the proxy object for the local Java class.

Return value Long. Returns 0 for success and one of the following values for failure:

- 1 Failed to create Java class.
- 2 Invalid proxy name.
- 3 Failed to create proxy object.

Examples

In this example, the create method accepts a Java Integer class argument. PowerBuilder creates a proxy called `java_integer` (the prefix `java_` is required to prevent a conflict with the PowerBuilder integer type). The call to `CreateJavaInstance` sets the value of that variable so you can call the EJB create method:

```
CustomerRemoteHome homeobj
CustomerRemote beanobj
java_integer jint_a

try
    homeobj = conn.lookup("CustomerRemoteHome", &
        "custpkg/Customer", "custpkg.CustomerRemoteHome" )
catch (Exception e)
    MessageBox( "Exception in Lookup", e.getMessage() )
    return
end try

try
    g_jvm.createJavaInstance(jint_a, "java_integer")
    jint_a.java_integer("8")
    beanobj = homeobj.create( jint_a, sle_name.text )
catch (RemoteException re)
    MessageBox( "Remote Exception", re.getMessage() )
    return
catch (CreateException ce)
    MessageBox( "Create Exception", ce.getMessage() )
    return
```

```

catch (Throwable t)
    MessageBox(" Other Exception", t.getMessage())
end try

MessageBox( "Info", &
    "This record has been successfully saved " &
    + "~r~ninto the database" )

```

**Usage** Use this method when an EJB method accepts a Java class as an argument. For example, if the primary key class argument to the `findByPrimaryKey` method is a Java class, use the `CreateJavaInstance` method to create the primary key class. You then use a `PowerBuilder proxy` to communicate with the Java class.

## DynamicCast

**Description** Converts an instantiated `PowerBuilder proxy object` to a proxy for the passed-in proxy name.

**Syntax** `javavm.DynamicCast(powerobject proxyobject, readonly string proxyname)`

Argument	Description
<i>javavm</i>	An instance of the JavaVM class
<i>proxyobject</i>	An instantiated <code>PowerBuilder proxy object</code>
<i>proxyname</i>	A string containing the name of the proxy to be instantiated

**Return value** `Powerobject`. A new proxy object for the Java class referenced by *proxyname*. This method returns null if the proxy cannot be created.

**Examples** **Example 1** In the following example, the object returned from the `nextElement` method is represented by a proxy for the `Employee` class. The `GetActualClass` method is used to determine whether the object is actually a `SalariedEmployee`, and if it is, the proxy `px_Employee` is down cast to the proxy `px_SalariedEmployee` so that the `adjustSalary` method can be called:

```

DepartmentHome    px_DeptHome
Department        px_Dept
Enumeration       px_EmployeeList
Employee          px_Employee
Salaried          px_SalariedEmployee
Contract          px_ContractEmployee
EJBConnection    conn

conn = create ejbconnection
try
    conn.connectToServer(properties)

```

```

px_DeptHome = conn.lookup("DepartmentHome", &
    "Department", &
    "com.joesportinggoods.ejbs.DepartmentHome")
px_Dept = px_DeptHome.findByPrimaryKey(as_DeptName)

px_EmployeeList = px_Dept.getEmployees()
DO WHILE px_EmployeeList.hasMoreElements()
    px_Employee = px_EmployeeList.nextElement()
    IF i_jvm.getActualClass(px_Employee) = &
        "com.joesportinggoods.ejbs.Salaried" THEN
        px_SalariedEmployee = &
            i_jvm.dynamicCast(px_Employee, "Salaried")
        px_SalariedEmployee.adjustSalary(al_increase)
    END IF
LOOP
catch (Exception e)
    THROW CREATE ApplyRaiseException
end try

```

**Example 2** In this example, `getAllItems` returns a `java.lang.Object` in the EJB declaration, which maps to the PowerBuilder Any data type. The call to `GetInterfaces` determines whether what is returned is a `java.util.List`. If it is, a call to `DynamicCast` obtains a proxy for `List`, which is used to obtain the size of the list before using its `Get` method to obtain the elements of the list. A method such as `getAllItems` can be used in many situations, such as to get a list of part numbers for any type of product.

```

ItemManagerHome px_ItemMgrHome
ItemManager px_ItemMgr
Item px_Item
List px_ItemList
any any_Object
boolean ib_isAList = FALSE
string is_IFs[]
string is_actualClass
long ll_row

TRY
    px_ItemMgrHome =
        g_EJBConn.Lookup("ItemManagerHome", &
            "ItemManager", "com.xapic.ItemManagerHome")
    px_ItemMgr = px_ItemMgrHome.create()
    any_Object = px_ItemMgr.getAllItems()
    // check if object implements java.util.List interface
    integer i
    FOR i = 1 to g_javaVM.getInterfaces(any_Object, &
        is_IFs)

```

```
IF is_IFs[i] = "java.util.List" THEN
    ib_isAList = TRUE
    EXIT
END IF
NEXT
// if it is a list
IF ib_isAList THEN
    px_ItemList = g_javaVM.dynamicCast(any_Object, &
        "list")
    // traverse the list
    FOR i = 0 TO px_ItemList.size() - 1
        // get item on the list
        any_Object = px_ItemList.get(i)
        // determine its class and dynamically cast it
        is_actualClass = &
            g_javaVM.getActualClass(any_Object)
        is_actualClass = Mid(is_actualClass, &
            LastPos(is_actualClass, ".") + 1, &
            Len(is_actualClass))
        px_Item = g_javaVM.dynamicCast(any_Object,
            is_actualClass)
        // add item to datastore
        ll_row = ads_Items.insertRow(0)
        ads_Items.object.id[ll_row] = px_Item.getID()
        ads_Items.object.type[ll_row] = is_actualClass
    NEXT
END IF
CATCH (Throwable t)
    // Handle exception
END TRY
```

## Usage

There are two scenarios in which a Java object returned from a call to an EJB method can be represented by a proxy that does not provide the methods you need:

- If the class of a Java object returned from an EJB method call is dynamically generated, PowerBuilder uses a proxy for the first interface implemented by the Java class.
- The prototype of an EJB method that actually returns *someclass* can be defined to return a class that *someclass* extends or implements.

For example, the prototype of a method that actually returns an object of type `java.util.ArrayList` can be defined to return `java.util.Collection` instead. (The `java.util.ArrayList` class inherits from `java.util.AbstractList`, which inherits from `java.util.AbstractCollection`, which implements `java.util.Collection`.) If the method prototype has a return type of `java.util.Collection`, `PowerBuilder` uses a proxy for `java.util.Collection`.

The `DynamicCast` method allows you to cast the returned proxy object to a proxy for the interface you require, or for the actual class of the object returned at runtime so that the methods of that object can be used.

You can obtain the actual class of the object using the `GetActualClass` method. You can also use the `DynamicCast` method with the `GetSuperClass` method, which returns the immediate parent of the Java class, and the `GetInterfaces` method, which writes a list of interfaces implemented by the class to an array of strings.

For example, consider the following class:

```
public class java.util.LinkedList extends java.util.AbstractSequentialList
implements java.util.List, java.lang.Cloneable, java.io.Serializable
```

`GetActualClass` returns `java.util.LinkedList`, `GetSuperClass` returns `java.util.AbstractSequentialList`, and `GetInterfaces` returns 3 and writes three strings to the referenced string array: `java.util.List`, `java.lang.Cloneable`, and `java.io.Serializable`.

See also

`CreateJavaVM`  
`GetActualClass`  
`GetInterfaces`  
`GetSuperClass`

## GetActualClass

**Description** Returns the class of the Java object that a PowerBuilder proxy object represents.

**Syntax** `javavm.GetActualClass(powerobject proxyobject)`

<b>Argument</b>	<b>Description</b>
<i>javavm</i>	An instance of the JavaVM class
<i>proxyobject</i>	An instantiated PowerBuilder proxy object

**Return value** String

**Usage** If an EJB method is defined to return a Java class that is not the actual object returned at runtime, but is instead a class that the actual object's class extends or implements, you can use `GetActualClass` to return the class of the actual object returned. You can then use the `DynamicCast` method to cast the proxy returned from the method to a proxy for the actual class of the object.

For more information and an example, see the description of the `DynamicCast` method.

**See also** `CreateJavaVM`  
`DynamicCast`  
`GetInterfaces`  
`GetSuperClass`

## GetInterfaces

**Description** Populates a string array with the names of interfaces implemented by the Java object that a PowerBuilder proxy object represents.

**Syntax** `javavm.GetInterfaces(powerobject proxyobject, ref string interfacename[ ] )`

Argument	Description
<i>javavm</i>	An instance of the JavaVM class
<i>proxyobject</i>	An instantiated PowerBuilder proxy object
<i>interfacename[ ]</i>	A reference to an unbounded array of strings to hold the names of interfaces implemented by the Java object represented by the PowerBuilder proxy object

**Return value** Integer. Returns the number of interfaces implemented by the Java object represented by *proxyobject*. If no interfaces are implemented by the Java object, this method returns 0. If *proxyobject* is invalid, this method returns -1.

**Usage** If a class implements multiple interfaces, the proxy returned from an EJB method call that returns a Java object maps to the first interface implemented by the Java class. This method writes a list of interfaces implemented by the class to an array of strings. It can be used in conjunction with the DynamicCast method to cast the returned proxy to the interface required.

For more information, see the description of the DynamicCast method.

**See also** CreateJavaVM  
DynamicCast  
GetActualClass  
GetSuperClass

## GetJavaClasspath

Description	Gets the classpath of the current Java VM.
Syntax	<code>javavm.getJavaClasspath()</code>
Return value	String
Examples	This example shows how to use <code>GetJavaClasspath</code> to get the classpath when the JVM is started and write it to a log file:

```
// instance variables:
// JavaVM i_jvm
// boolean i_jvm_started = false
// string is_classes

//Start JavaVM and Prepare to Connect to EJB server
string classpath
Integer li_ret

//create JAVAVM
if ib_jvm_started = false then
    i_jvm = create javavm

    classpath = is_classes
    li_ret = i_jvm.createJavaVM(classpath, true)
    if li_ret = -1 then
        MessageBox("Error", "Failed to load JavaVM")
    end if
    if li_ret = -2 then
        MessageBox("Error", "Failed to load EJBLocator")
    end if

    ib_jvm_started = true

    integer li_FileNum
    string ls_classpath, ls_string

    li_FileNum = FileOpen("C:\temp\classpath.log", &
        LineMode!, Write!, LockWrite!, Append!)
    ls_classpath = i_jvm.getjavaclasspath()
    ls_string = String(Today()) + " " + String(Now())
    ls_string += ": ~r~n" + ls_classpath + "~r~n"

    FileWrite(li_FileNum, ls_string)
    FileClose(li_filenum)
end if
```

See also

`CreateJavaVM`, `GetJavaVMVersion`, `IsJavaVMLoaded`



## GetJavaVMVersion

Description	Gets the version number of the current Java VM.
Syntax	<code>javavm.getJavaVMVersion()</code>
Return value	String representing the Java VM version. For example, for JDK 1.4, <code>GetJavaVMVersion</code> returns 1.4.0.
Examples	This example shows how to use <code>GetJavaVMVersion</code> : <pre> // global variable JavaVM g_jvm string ls_javaVMVersion ls_javaVMVersion = g_jvm.<b>getJavaVMVersion()</b> </pre>
See also	<code>CreateJavaVM</code> <code>GetJavaClasspath</code> <code>IsJavaVMLoaded</code>

## GetSuperClass

Description	Returns the name of the super class of the class of the Java object that a <code>PowerBuilder proxy object</code> represents.						
Syntax	<code>javavm.GetSuperClass(powerobject proxyobject)</code>						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>javavm</i></td> <td>An instance of the JavaVM class</td> </tr> <tr> <td><i>proxyobject</i></td> <td>An instantiated <code>PowerBuilder proxy object</code></td> </tr> </tbody> </table>	Argument	Description	<i>javavm</i>	An instance of the JavaVM class	<i>proxyobject</i>	An instantiated <code>PowerBuilder proxy object</code>
Argument	Description						
<i>javavm</i>	An instance of the JavaVM class						
<i>proxyobject</i>	An instantiated <code>PowerBuilder proxy object</code>						
Return value	String. If the current Java object is <code>Java.lang.Object</code> or an interface, returns null.						
Examples	This example assumes that you have subclassed the <code>Java Decimal</code> class. Your class, <code>My.Decimal</code> , extends <code>java.lang.Decimal</code> . After you build a proxy project for this class, you can determine the real Java class name that the proxy represents with code like the following:						

```

java_decimal dec_num
string classname, supername

conn.createjavainstance(dec_num, "java_decimal")
classname = g_javavm.getactualclass(dec_num) &
  classname = "My.Decimal"
supername = g_javavm.getsuperclass(dec_num) &
  supername = "java.lang.Decimal"

```

Usage	<p>This method returns the name of the immediate parent of the class referenced by the proxy object. For example, if <i>proxyobject</i> is a <code>java.io.FilterReader</code>, <code>GetSuperClass</code> returns <code>java.io.Reader</code>. <code>GetSuperClass</code> can be used in conjunction with the <code>GetInterfaces</code> and <code>DynamicCast</code> methods to cast a proxy object returned from an EJB method call to a different object.</p> <p>For more information, see the description of the <code>DynamicCast</code> method.</p>
See also	<p><code>CreateJavaVM</code> <code>DynamicCast</code> <code>GetActualClass</code> <code>GetInterfaces</code></p>

## IsJavaVMLoaded

Description	Determines whether the Java VM has been loaded.				
Syntax	<pre>javavm.IsJavaVMLoaded( )</pre> <table border="1"><thead><tr><th>Argument</th><th>Description</th></tr></thead><tbody><tr><td><i>javavm</i></td><td>An instance of the JavaVM class</td></tr></tbody></table>	Argument	Description	<i>javavm</i>	An instance of the JavaVM class
Argument	Description				
<i>javavm</i>	An instance of the JavaVM class				
Return value	Boolean. Returns true if the Java VM has already been loaded and false if it has not.				
Examples	<p>This example tests whether the Java VM has been loaded before attempting to create and load a Java VM:</p> <pre>if (IsJavaVMLoaded) then     // skip some processing else     // perform processing end if</pre>				
Usage	<p>Use this method if you need to determine whether the Java VM is loaded before proceeding. You might want to enable or disable some features of your application if the Java VM has already been loaded. For example, if your application provides a window in which the user can specify a list of classes that is added to the classpath used by the <code>CreateJavaVM</code> method, you can disable this feature if the Java VM has already been loaded, because any changes made in that window would have no effect.</p>				
See also	<p><code>CreateJavaVM</code> <code>GetJavaClasspath</code> <code>GetJavaVMVersion</code></p>				

## LoadMappingTable

**Description** Loads the the mapping table between the Java class and a specified PowerBuilder EJB proxy.

**Syntax** `javavm.LoadMappingTable(proxyname)`

Argument	Description
<code>javavm</code>	An instance of the JavaVM class
<code>proxyname</code>	The name of the proxy object for the local JavaVM class

**Return value** Boolean. Returns true if the mapping table is successfully loaded, and false if the load fails.

**Examples** This example creates a Java VM, then tests whether the EBJ mapping table has been loaded before attempting to perform operations involving the VM:

```

JavaVM g_jvm
string classpath
boolean isdebug
foo l_foo
classpath = "D:\tests\javasample\bin;"
isdebug = false
g_jvm.CreateJavaVM(classpath, isdebug)
g_jvm.CreateJavaInstance(l_foo, "foo")
if (LoadMappingTable("foo")) then
    // perform normal processing
else
    // handle failure to load mapping table
end if

```

**Usage** Call LoadMappingTable after calling JavaVM.create, otherwise an exception is thrown.

**See also** CreateJavaVM  
GetJavaClasspath  
GetJavaVMVersion



## About this chapter

This chapter describes the PowerBuilder extension classes used to connect to a SOAP server that hosts a Web service you want to access. It also describes the extension classes that enable you to search UDDI registries for a Web Service. For more information about working with Web services, see *Application Techniques*.

## Contents

Topic	Page
SoapConnection	31
SoapException	44
SoapPBCookie	46
UDDIProxy	54

**SoapConnection**

## Description

The SoapConnection class is used to create a proxy object for a specific Web service and set options for the connection.

## Methods

SoapConnection has the following methods:

AddToBypassList	SetOptions
CreateInstance	SetProxyServer
DynamicCast	SetProxyServerOptions
RemoveAuthentication	SetSoapLogFile
RemoveBypassList	SetTimeout
SetBasicAuthentication	SetUseDefaultProxySetting
SetBypassProxyOnLocal	UseConnectionCache
SetClientCertificateFile	UseIntegratedWindowsAuthentication

The GenerateProxy method is currently not implemented.

## AddToBypassList

**Description** Adds URIs to a list of locations that can be accessed without connecting to a proxy server. This method is available for .NET Web services only.

**Syntax** `conn.AddToBypassList (string value)`

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection
<i>value</i>	A regular expression that defines URIs that can be accessed without connecting to a proxy server.

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**Usage** You can use asterisks for wild cards in expressions for domain or host names and addresses. You can add multiple URIs to the bypass list in a single call by including semicolon separators to the *value* string expression.

**See also** RemoveBypassList  
SetBypassProxyOnLocal

## CreateInstance

**Description** Creates a proxy instance with a default URL for a SOAP server, which comes from a user-supplied WSDL file. The client application must create a proxy instance before it can access a Web service.

**Syntax** `conn.CreateInstance (ref powerobject proxy_obj, string proxy_name, {string portname})` throws SoapException

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection
<i>proxy_obj</i>	The referenced name of your proxy object
<i>proxy_name</i>	The name of the proxy, based on the port name from a URL in the WSDL file stored in the proxy
<i>portname</i>	(Optional) The port name from a URL not stored in the proxy

**Return value** Long. Valid values are:

Value	Description
0	Successful
100	Invalid proxy name
101	Failed to create proxy

## Examples

**Example 1** In this example, the client application creates a proxy instance to access the Web services at *http://my.server/soap/myport*. The proxy name "syb\_myport" is generated by the Web Service Proxy wizard when you select "syb\_" as a prefix for a service port (endpoint) called "myport".

```
syb_myport myproxy
long ret

ret = Conn.CreateInstance(myproxy, "syb_myport",
    "http://my.server/soap/myport")
```

**Example 2** The following script creates a connection to a Web service on a SOAP server. It sets the connection properties using an endpoint defined in the CreateInstance method. (If the endpoint is not defined in the CreateInstance method, a default URL stored in the proxy is used). The script uses the SetOptions method to specify a log file. It displays a return value in an application message box:

```
SoapConnection conn // Define SoapConnection
syb_currencyexchangeport proxy_obj // Declare proxy
long rVal, lLog
real amount

//Define endpoint. You can omit it, if you want to use
//the default endpoint inside proxy

string str_endpoint

str_endpoint = "http://services.xmethods.net:80/soap"
conn = create SoapConnection //Instantiated connection

lLog = conn.SetOptions("SoapLog=~"C:\mySoapLog.log~")

// Set trace file to record soap interchange data,
// if string is "", disables the feature

rVal = Conn.CreateInstance(proxy_obj, &
    "syb_currencyexchangeport", str_endpoint)

// Create proxy object
try

    amount = proxy_obj.getrate("us","japan")
    // Invoke service
```

```

        messagebox("Current Exchange Rate", "One US Dollar"&
        + " is equal to " + string(amount) + " Japanese Yen")
    catch ( SoapException e )
        messagebox ("Error", "Cannot invoke Web service")
        // error handling
    end try
    destroy conn

```

**Usage** After you instantiate a proxy, you are ready to call the SOAP methods you want from the associated Web service port.

**See also** SetOptions  
SetProxyServerOptions

## DynamicCast

**Description** Dynamically casts a variable from one datatype (nonvisual object or structure) to another datatype, and copies runtime data from the source datatype to the target datatype. However, you must make sure the data in source datatype can be converted to the target datatype before you call this method.

This method is available for .NET Web services only.

**Syntax** `conn.DynamicCast (powerobject src, string targettype)`

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection
<i>src</i>	The PowerScript datatype that you want to convert
<i>targettype</i>	A string specifying the datatype to which you want to convert the object

**Return value** Powerobject. An object of the datatype specified by the *targettype* variable.

**Examples** The following code converts a returned message from the msgA datatype to the msgB datatype.

```

    Try
        msgA = myReport.GetMessage ()
        MessageB msgB
        msgB = lsc_connection.dynamiccast (msgA, "MessageB")

    Catch (SoapException e)
        ...
    End Try

```



**Usage** Some Web services return runtime data of a subclass even though the definition of the Web service method uses a base class. You can call the `DynamicCast` method to cast the proxy object for the subclass to the proxy object for the base class.

After you convert the object to the datatype you want, you can access every field in that object.

## RemoveAuthentication

**Description** Removes authentication for a Web service connection.

**Syntax** `conn.RemoveAuthentication ()`

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection.

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**Usage** This method clears Basic, Digest, and Integrated Windows Authentication information. You can set authentication with the `UseIntegratedWindowsAuthentication` (.NET Web service clients only), `SetBasicAuthentication`, or `SetOptions` methods.

**See also** `SetBasicAuthentication`  
`SetOptions`  
`UseIntegratedWindowsAuthentication`

## RemoveBypassList

**Description** Removes the list of URIs to access without connecting to a proxy server. This method is available for .NET Web services only.

**Syntax** `conn.RemoveBypassList ()`

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection.

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**See also** `AddToBypassList`

## SetBasicAuthentication

**Description** Determines whether the SoapConnection object uses basic authentication for a Web service connection.

**Syntax** `conn.SetBasicAuthentication` (string *domain*, string *userID*, string *password*)

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection.
<i>domain</i>	A string for the Web domain to which the user belongs. This could be a domain name, such as “sybase.com”, or a machine name.
<i>userID</i>	A string for an https connection.
<i>password</i>	A string for an https connection.

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**Usage** You can call the SetBasicAuthentication method instead of including client identification information in the *options* argument of the SetOptions method.

If you are using .NET Web services, you can call the UseIntegratedWindowsAuthentication method for Integrated Windows Authentication.

**See also** RemoveAuthentication  
SetOptions  
UseIntegratedWindowsAuthentication

## SetBypassProxyOnLocal

Description	Indicates whether to bypass the proxy server when connecting to Web services running on local servers. This method is available for .NET Web services only.						
Syntax	<code>conn.SetBypassProxyOnLocal</code> (boolean <i>bypass</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>conn</i></td> <td>The name of the SoapConnection object that establishes the connection.</td> </tr> <tr> <td><i>bypass</i></td> <td>A boolean value that, when true, allows a connection to a local resource without using a proxy server. All internet requests are made through the proxy server when this value is false.</td> </tr> </tbody> </table>	Argument	Description	<i>conn</i>	The name of the SoapConnection object that establishes the connection.	<i>bypass</i>	A boolean value that, when true, allows a connection to a local resource without using a proxy server. All internet requests are made through the proxy server when this value is false.
Argument	Description						
<i>conn</i>	The name of the SoapConnection object that establishes the connection.						
<i>bypass</i>	A boolean value that, when true, allows a connection to a local resource without using a proxy server. All internet requests are made through the proxy server when this value is false.						
Return value	Long. Valid values are 0 for success, and 50 for failure.						
Usage	Local requests use the localhost or loopback domains, or a local IP address. Addresses without a period in the URI are also identified as being local.						
See also	AddToBypassList SetUseDefaultProxySetting						

## SetClientCertificateFile

Description	Sets the certificate file or files to use to connect to a Web service. This method is available for .NET Web services only.						
Syntax	<code>conn.SetClientCertificateFile</code> (string <i>filename</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>conn</i></td> <td>The name of the SoapConnection object that establishes the connection.</td> </tr> <tr> <td><i>filename</i></td> <td>A string containing the name of the certificate file or files you want to use to connect to a Web service. You must use a semicolon as a separator for multiple files. The value can include local files with a full path and URLs to remote certificate files. To discontinue use of certificates, enter an empty string ("").</td> </tr> </tbody> </table>	Argument	Description	<i>conn</i>	The name of the SoapConnection object that establishes the connection.	<i>filename</i>	A string containing the name of the certificate file or files you want to use to connect to a Web service. You must use a semicolon as a separator for multiple files. The value can include local files with a full path and URLs to remote certificate files. To discontinue use of certificates, enter an empty string ("").
Argument	Description						
<i>conn</i>	The name of the SoapConnection object that establishes the connection.						
<i>filename</i>	A string containing the name of the certificate file or files you want to use to connect to a Web service. You must use a semicolon as a separator for multiple files. The value can include local files with a full path and URLs to remote certificate files. To discontinue use of certificates, enter an empty string ("").						
Return value	Long. Valid values are 0 for success, and 50 for failure.						
Usage	You can call the SetClientCertificateFile method instead of including certificate information in the <i>options</i> argument of the SetOptions method.						
See also	SetBasicAuthentication SetOptions UseIntegratedWindowsAuthentication						

## SetOptions

Description

Sets connection options for SoapConnection class.

Syntax

**conn.SetOptions** (string *options*)

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection.
<i>options</i>	<p>Options you want to set for your connection. The string values for the option names are not case sensitive. These can be:</p> <p><b>SoapLog</b> (EasySoap Web service engine only) The file path for SoapLog. To disable the log, enter "".</p> <p><b>UserID</b> A string value for an https connection.</p> <p><b>Password</b> A string value for an https connection.</p> <p><b>Domain</b> (.NET Web service engine only) A string value for the Web domain to which the user belongs. This could be a domain name, such as "sybase.com", or a machine name.</p> <p><b>UseWindowsAuthentication</b> (.NET Web service engine only) A "yes" or "no" value to determine whether to use "Integrated Windows Authentication." The value you enter can be a boolean or a string. If this option is set to "yes," you do not need to set the UserID, Password, or Domain options.</p> <p><b>AuthenticationMode</b> (.NET Web service engine only) A string value for the authentication mode to use. This can be "basic" or "digest". These AuthenticationMode values are described on the Microsoft MSDN Web site at <a href="http://msdn.microsoft.com/en-us/library/aa833874(VS.80).aspx">http://msdn.microsoft.com/en-us/library/aa833874(VS.80).aspx</a>.</p> <p><b>CertificateFile</b> (.NET Web service engine only) A string value for the certificate file or files that you want to send from the Web service client to the server. The string value could include local files with a full path and URLs to remote certificate files. You must use a semicolon as a separator for multiple files.</p> <p><b>Timeout</b> A number for the maximum wait time in seconds. The default timeout value is 0, meaning that no limit to the connection time is set.</p> <p><b>ConnectionCache</b> (EasySoap Web service engine only) A boolean that determines whether the http connection of the proxy instance is kept alive after a call to the proxy. The default value is false. For Web services on EAServer, you must not change the default.</p>

Return value

Long. Valid values are 0 for success, and 50 for failure. If multiple options are specified and the return value is 50, options specified before the failure are still valid.

## Examples

In this example, the application enables the logging function and attempts to connect to an endpoint for which no user ID, password, or timeout has been set.

```
lOpt=Conn.SetOptions ("SoapLog=~"airportweather.log~")
```

To avoid using escape characters before a second pair of quotation marks, use single quotation marks instead, or you can start an exterior string with single quotation marks and use double quotation marks around an interior string:

```
lOpt=Conn.SetOptions ('SoapLog="airportweather.log"')
```

## Usage

User ID and password values can be set in an endpoint used by the SoapConnection class or by including these values as arguments to the SetOptions method.

Priority is given to values set in an endpoint (port) that is passed as an argument to the CreateInstance method of the SoapConnection class. However, a default endpoint is used when an endpoint is not set in the CreateInstance method. In this case, priority is given to user ID and password values defined in the SetOptions method.

If the endpoint used by the SoapConnection class does not have user ID and password values, and you do not set a user ID or password with the SetOptions method, the SoapConnection class connects to a SOAP server without giving a user ID or password.

If a user ID is defined in either the endpoint or the SetOptions method but is not a password, the password value is taken to be an empty string.

When you set a timeout other than the default, an exception is thrown after the Web service connection times out. Even if you do not set a timeout value from the client, the Web server can cause the request to time out on the server side.

If you include ConnectionCache as an argument in a SetOptions call, you must not use quotation marks to enclose the value that you set for this option.

Although SetOptions takes a single string argument for all available options, you can set each of the options with more specific methods. You can use the following methods to replace SetOptions:

For .NET Web services	For EasySoap Web services
<ul style="list-style-type: none"> <li>• SetBasicAuthentication</li> <li>• SetClientCertificateFile</li> <li>• SetTimeout</li> <li>• UseIntegratedWindowsAuthentication</li> <li>• RemoveAuthentication</li> </ul>	<ul style="list-style-type: none"> <li>• SetBasicAuthentication</li> <li>• SetSoapLogFile</li> <li>• SetTimeout</li> <li>• UseConnectionCache</li> <li>• RemoveAuthentication</li> </ul>

## See also

CreateInstance, SetProxyServerOptions

## SetProxyServer

**Description** Sets the address, port, user name, and password for the proxy server. This method has two syntaxes.

**Syntax**

*conn*.**SetProxyServer** (string *address*, string *userID*, string *password*)

*conn*.**SetProxyServer** (string *hostname*, long *port*, string *userID*, string *password*)

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection
<i>address</i>	A string containing the host name address and port of the proxy server, and optionally, an endpoint— in the format: <code>http://hostname:port/path</code>
<i>hostname</i>	A string containing the host name
<i>port</i>	A long for the proxy server port
<i>userID</i>	A string containing the user ID for the proxy server
<i>password</i>	A string containing the proxy server password

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**Examples** This example uses the four-argument syntax of SetProxyServer:

```
long ll_return
ll_return = Conn.SetProxyServer &
("http://myProxyServer", 8080, "My Name", "My Pass")
```

**Usage** This method does the same thing as the SetProxyServerOptions method, but it has a different syntax.

Use this method or the SetProxyServerOptions method if the proxy server requires authentication. The user ID and password that you supply with the SetOptions or other authentication methods apply to the URL of the Web service, not the proxy server.

**See also** SetOptions  
SetProxyServerOptions

## SetProxyServerOptions

**Description** Sets the proxy address, user name, and password for the proxy server.

**Syntax** `conn.SetProxyServerOptions` (string *optionstring*)

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection.
<i>optionstring</i>	A string containing comma-separated name/value pairs. The format is: <pre>"address='proxy_endpoint' {, userID='name', password='password'}"</pre> The address is required and can have a format such as: <pre>http://hostname:port/path</pre> Specify values for userID and password if the proxy server requires them.

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**Examples** This example specifies a user name and password, as well as the proxy endpoint:

```
long ll_return
string ls_string
ls_string = "address='http://Srvr:8080/endpoint',"
ls_string += "userID='MyName', password='mypass'"
ll_return = Conn.SetProxyServerOptions (ls_string)
```

**Usage** Use this method or the SetProxyServer method if the proxy server requires authentication. The user ID and password that you supply with the SetOptions or other authentication methods apply to the URL of the Web service, not the proxy server.

**See also** CreateInstance  
SetOptions  
SetProxyServer

## SetSoapLogFile

**Description** Sets the name of a file for logging raw SOAP messages. This method is available for EasySoap Web services only.

**Syntax** `conn.SetSoapLogFile` (string *filename*)

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection.
<i>filename</i>	A string containing the full file name for the SOAP log file. To disable logging, enter an empty string ("").

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**Usage** You can call the SetSOAPLogFile method instead of including a log file name in the *options* argument of the SetOptions method.

**See also** SetOptions

## SetTimeout

**Description** Sets the timeout value for a SOAP connection.

**Syntax** `conn.SetTimeout` (long *seconds*)

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection.
<i>seconds</i>	The timeout value in seconds. If this option is set to 0, no timeout will be set on the client side. (The Web service might still have a timeout value on the server side.)

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**Usage** You can call the SetTimeout method instead of including a timeout value in the *options* argument of the SetOptions method.

**See also** SetOptions



## SetUseDefaultProxySetting

**Description** Indicates whether to use Internet Explorer proxy settings for a SOAP connection. This method is available for .NET Web services only.

**Syntax** `conn.SetUseDefaultProxySetting` (boolean *useDefault*)

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection.
<i>useDefault</i>	A boolean value that, when true, uses the Internet Explorer proxy settings to connect to a Web service. When this value is false (default), the proxy server settings can be assigned by the SetProxyOption, SetBypassOnLocal, AddToBypassList, and RemoveBypassList methods.

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**Usage** If you do not set a proxy server, PowerBuilder uses the Internet Explorer proxy settings.

**See also** AddToBypassList  
RemoveBypassList  
SetOptions

## UseConnectionCache

**Description** Determines whether a connection cache is used for the Web service connection. This method is available for EasySoap Web services only.

**Syntax** `conn.UseConnectionCache` (boolean *cache*)

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection.
<i>cache</i>	A boolean that determines whether the http connection of the proxy instance is kept alive after a call to the proxy. The default value is false. For Web services on EAServer, you must not change the default.

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**Usage** You can call the UseConnectionCache method instead of setting a connection cache in the *options* argument of the SetOptions method.

**See also** SetOptions  
SetSoapLogFile

## UseIntegratedWindowsAuthentication

**Description** Determines whether the SoapConnection object uses Integrated Windows Authentication to connect to a Web service. This method is available for .NET Web services only.

**Syntax** `conn.UseIntegratedWindowsAuthentication` (boolean *useIWA*)

Argument	Description
<i>conn</i>	The name of the SoapConnection object that establishes the connection.
<i>useIWA</i>	A boolean that determines whether to use Integrated Windows Authentication. If this option is set to “yes,” you do not need to set the UserID, Password, or Domain options.

**Return value** Long. Valid values are 0 for success, and 50 for failure.

**Usage** You can call the UseIntegratedWindowsAuthentication method to set connection authentication instead of the *options* argument of the SetOptions method.

**See also** RemoveAuthentication  
SetBasicAuthentication  
SetOptions

## SoapException

**Description** The SoapException class is a PBNI class that inherits from the PowerBuilder RuntimeError class. When an exception occurs in a Web service method call, it is converted into a SoapException and thrown. The methods of the classes in *PBSoapClient125.pbx* and *PBWSClient125.pbx* can also throw SoapException.

**Properties**

Exception property	Data type	Description
Text	String	Contains the text of the error message

**Methods** The following table defines methods inherited by a SoapException object from the RuntimeError class.

Exception method	Data type returned	Description
GetMessage	String	Returns the error message from objects of type RuntimeError

Exception method	Data type returned	Description
SetMessage	—	Sets an error message for an object of type RuntimeException

**Usage**

The following example demonstrates how to use the SoapException class. The ServiceProxy fails to be invoked and returns the error message. The code has three catch clauses: for SoapException, PBXRuntimeError, and RuntimeException. PBXRuntimeError is an exception class that inherits from RuntimeException and is thrown when a PowerBuilder extension raises an error that is not caught by the extension.

```

string s1,s2
s1 = "abcd"
SoapConnection conn
long ret
ServiceProxy proxy

//ServiceProxy is a proxy generated by Web service
//wizard
try
    conn = create SoapConnection
    ret = conn.CreateInstance(proxy, "ServiceProxy")
    if (ret <> 0)then
        MessageBox("Fail", "Cannot create proxy " &
            + "ServiceProxy")
        return
    end if
    s2 = proxy.EchoString(s1)
    MessageBox("Successful", "The return string is '" &
        + s2 + "'")
catch (SoapException e1)
    MessageBox("Fail", "Can't invoke service
'EchoString'")
catch (PBXRuntimeError e2)
    MessageBox("Fail", "There is a runtime error when" &
        + "invoking Web service")
catch (RuntimeException e3)
    MessageBox("Fail", "There is an unknown error when"&
        + "invoking Web service")
end try

```

**See also**

GetMessage in the *PowerScript Reference*  
 RuntimeException object in *Objects and Controls*  
 SetMessage in the *PowerScript Reference*

## SoapPBCookie

**Description** Use the SoapPBCookie class to get or set cookies for the Web service.

**Methods** SoapPBCookie has the following methods:

GetComment	SetComment
GetCommentUri	SetCommentUri
GetExpired	SetExpired
GetExpires	SetExpires
GetHttpOnly	SetHttpOnly
GetName	SetName
GetSecure	SetSecure
GetTimeStamp	SetURI
GetURI	SetValue
GetValue	SetVersion
GetVersion	

### GetComment

**Description** Gets a comment that the server provides with a cookie.

**Syntax** `acookie.GetComment ( )`

<b>Argument</b>	<b>Description</b>
<code>acookie</code>	The name of an instance of the SoapPBCookie object

**Return value** String. Returns a comment provided with the cookie.

**Usage** An optional comment added by the server typically includes information about privacy policy or intended uses of the cookie.

## GetCommentUri

Description Gets a URI comment that the server provides with a cookie.

Syntax `acookie.GetCommentUri ( )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object

Return value String. Returns a URI comment provided with the cookie.

Usage An optional comment added by the server that represents the intended use of the URI reference for the cookie.

## GetExpired

Description Gets the current state of a cookie.

Syntax `acookie.GetExpired ( )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object

Return value Boolean. Returns true if the cookie has expired. Otherwise, returns false.

## GetExpires

Description Gets the expiration date and time for a cookie.

Syntax `acookie.GetExpires ( )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object

Return value DateTime. Gets the expiration date and time of a cookie.

Usage A session cookie returns a DateTime value of January 1, 0001, 00:00:00.0000000.

## GetHttpOnly

Description Gets the accessibility of a cookie to page scripts or other active content.

Syntax `acookie.GetHttpOnly ( )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object

Return value Boolean. Returns false when a page script or other active content is able to access the cookie. Otherwise, returns true.

## GetName

Description Gets the name of a cookie.

Syntax `acookie.GetName ( )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object

Return value String. Returns the name of the cookie.

Usage For an example using `GetName`, see the description for the `PBGetCookies` function in the *PowerScript Reference*.

## GetSecure

Description Gets the security level of a cookie.

Syntax `acookie.GetSecure ( )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object

Return value Boolean. Returns true if HTTPS is required. Otherwise, returns false.

## GetTimeStamp

Description Gets the time when the cookie was issued.

Syntax `acookie.GetTimeStamp ( )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object

Return value DateTime. Gets the date and time when the cookie was issued.

## GetURI

Description Gets the URI for which the cookie is valid.

Syntax `acookie.GetURI ( )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object

Return value String. Returns the URI.

## GetValue

Description Gets the value of the cookie.

Syntax `acookie.GetValue ( )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object

Return value String. Returns the cookie value.

## GetVersion

Description Gets the version of the HTTP state maintenance to which a cookie conforms.

Syntax `acookie.GetVersion ( )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object

Return value Integer. Returns 1 if the cookie conforms to RFC 2109, and 2 if the cookie conforms to RFC 2965.

## SetComment

Description Sets a comment that the server can add to a cookie.

Syntax `acookie.SetComment ( string comment )`

Argument	Description
<code><i>acookie</i></code>	The name of an instance of the SoapPBCookie object
<code><i>comment</i></code>	String for a comment that you want the server to provide with a cookie

Return value Long. Returns 0 for success, and 50 for failure.

Usage Comments are optional. Typical comments include information about privacy policy and intended use of a cookie.

## SetCommentUri

Description Sets a comment.

Syntax `acookie.SetCommentURI ( string commentUri )`

Argument	Description
<code><i>acookie</i></code>	The name of an instance of the SoapPBCookie object
<code><i>commentUri</i></code>	String for a URI comment that you want the server to provide with a cookie

Return value Long. Returns 0 for success, and 50 for failure.

Usage URI comments are optional, but must conform to the URI format when used. Typical URI comments include information about how the server uses a cookie.

## SetExpired

Description Sets the state of a cookie.

Syntax `acookie.SetExpired ( boolean expired )`

Argument	Description
<code><i>acookie</i></code>	The name of an instance of the SoapPBCookie object
<code><i>expired</i></code>	Set to true if you want to terminate the cookie. The expired value is false by default.

Return value Long. Returns 0 for success, and 50 for failure.



## SetExpires

Description Sets the expiration date and time for a cookie.

Syntax *acookie*.SetExpires ( datetime *expires* )

Argument	Description
<i>acookie</i>	The name of an instance of the SoapPBCookie object
<i>expires</i>	A DateTime value for the expiration date and time you want to set for a cookie

Return value Long. Returns 0 for success, and 50 for failure.

Usage You set a session cookie by entering a DateTime value of January 1, 0001, 00:00:00.0000000.

## SetHttpOnly

Description Determines whether a cookie can be accessed by page scripts or other active content.

Syntax *acookie*.SetHttpOnly ( boolean *httpOnly* )

Argument	Description
<i>acookie</i>	The name of an instance of the SoapPBCookie object
<i>httpOnly</i>	Set to true if you want to restrict cookie to HTTP access only. Set to false if you want page scripts or other active content to be able to access the cookie.

Return value Long. Returns 0 for success, and 50 for failure.

## SetName

Description Sets the name for a cookie.

Syntax `acookie.SetName ( string name )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object
<code>name</code>	The name that you want to set for the cookie

Return value Long. Returns 0 for success, and 50 for failure.

Usage The name must be initialized before setting an instance of the Cookie class. The following characters cannot be used for the cookie name: equal sign (=), semicolon (;), comma (,), new line (\n), return (\r), and tab (\t). The dollar sign (\$) cannot be used as the first character in the name.

Cookies are considered the same if the values of both their URI and name are the same. If a cookie already exists in the Web service with the same name and URI, it will be replaced with the new cookie when you call a Web service method.

For an example using SetName, see the description for the PBAddCookie function in the *PowerScript Reference*.

## SetSecure

Description Sets the security level for a cookie.

Syntax `acookie.SetSecure ( boolean secure )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object
<code>secure</code>	Set this to true if you want the client to return the cookie only when Secure Hypertext Transfer Protocol (HTTPS) is used.

Return value Long. Returns 0 for success, and 50 for failure.

Usage SetSecure is false by default.

## SetURI

Description Sets the URI for which the cookie is valid.

Syntax `acookie.SetURI ( string uri )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object
<code>uri</code>	The URI for which the cookie is valid

Return value Long. Returns 0 for success, and 50 for failure.

Usage The URI value you set must conform to the URI format.

## SetValue

Description Sets the value for a cookie.

Syntax `acookie.SetValue ( string value )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object
<code>value</code>	A string value that you want to set for the cookie

Return value Long. Returns 0 for success, and 50 for failure.

Usage Semicolons and commas cannot be used in the value that you set for a cookie.

## SetVersion

Description Sets the HTTP state maintenance version to which a cookie conforms.

Syntax `acookie.SetVersion ( int version )`

Argument	Description
<code>acookie</code>	The name of an instance of the SoapPBCookie object
<code>version</code>	The HTTP version to which you want the cookie to conform.

Return value Long. Returns 0 for success, and 50 for failure.

Usage If you set `version` to 1, the cookie must conform to RFC 2109. If you set the cookie to 2, the cookie must conform too RFC 2965.

## UDDIProxy

**Description** The UDDIProxy class is used to create a proxy object for a UDDI search and set options for that search.

**Methods** UDDIProxy has the following methods:

```
setInquiryUrl  
setOption  
findBusiness  
getBusinessDetail  
findService
```

### setInquiryUrl

**Description** Sets the UDDI inquiry URL.

**Syntax** *proxy.setinquiryurl* (readonly string *url*)

Argument	Description
<i>proxy</i>	The name of the UDDIProxy object
<i>url</i>	A valid UDDI inquiry URL

**Return value** Integer. Valid values are 1 for success, and 0 for failure.

**Examples** The following code sets the inquiry URL to a UDDI registry on the IBM Web site:

```
uddiproxy proxy  
int ret  
proxy = create uddiproxy  
ret = proxy.setinquiryurl  
    ("http://www-3.ibm.com/services/uddi/inquiryapi")  
...//search processing  
destroy proxy
```

## setOption

**Description** Sets UDDI search options for match precision, case sensitivity, result sort order, and the maximum number of rows returned.

**Syntax** `proxy.setoption` (boolean *exactMatch*, boolean *caseSensitive*, integer *sort*, integer *maxRow*)

Argument	Description
<i>proxy</i>	The name of the UDDIProxy object.
<i>exactMatch</i>	If true, search returns exact matches only.
<i>caseSensitive</i>	If true, search result must match the case used by search key word.
<i>sort</i>	Determines whether or how search results are sorted. Values are: <ul style="list-style-type: none"> <li>• -1 sorts results in descending order</li> <li>• 0 performs no sorting</li> <li>• 1 sorts results in ascending order</li> </ul>
<i>maxRow</i>	Maximum number of items a search can return.

**Return value** Integer. Valid values are 1 for success, and 0 for failure.

**Examples** The following code sets options for case sensitivity and the maximum number of rows returned:

```
ret = proxy.setoption (false, true, 0, 5)
```

## findBusiness

**Description** Finds business items using business names in a UDDI search.

**Syntax** `proxy.findBusiness` (readonly string *businessName*, ref integer *count*, ref string *busNameResult* [ ], ref string *busDescriptionResult* [ ], ref string *busKeyResult* [ ])

Argument	Description
<i>proxy</i>	The name of the UDDIProxy object
<i>businessName</i>	Business name to search in UDDI registry
<i>count</i>	Number of search results returned; never larger than the <i>maxRow</i> input parameter in a corresponding <code>setOption</code> call
<i>busNameResult</i>	Array of business names matching the search criteria
<i>busDescriptionResult</i>	Array of descriptions for businesses matching the search criteria
<i>busKeyResult</i>	Array of globally unique identifiers (GUIDs) for each business matching the search criteria

**Return value** Integer. Valid values are 1 for success, and 0 for failure.

**Examples** The following code finds business names, descriptions, and keys in the IBM UDDI registry:

```

uddiproxy proxy
proxy = create uddiproxy
int count
string businessName[], businessDescription[]
string businessKey []
proxy.findbusiness("IBM", count, businessName, &
    businessDescription, businessKey)
    
```

## getBusinessDetail

**Description** Gets business details using a business key that is typically obtained from the findBusiness method.

**Syntax** *proxy*.**getBusinessDetail** (readonly string *businessKey*, ref integer *count*, ref string *serviceNameResult* [ ], ref string *serviceDescriptionResult* [ ], ref string *serviceKeyResult* [ ], ref string *wsdl* [ ])

Argument	Description
<i>proxy</i>	The name of the UDDIProxy object
<i>businessKey</i>	Business key to search in UDDI registry
<i>count</i>	Number of search results returned; never larger than the <i>maxRow</i> input parameter in a corresponding <i>setOption</i> call
<i>serviceNameResult</i>	Array of services matching the search criteria
<i>serviceDescriptionResult</i>	Array of descriptions for services matching the search criteria
<i>serviceKeyResult</i>	Array of globally unique identifiers (GUIDs) for each service matching the search criteria
<i>wsdl</i>	Array of WSDL file names for services matching search criteria

**Return value** Integer. Valid values are 1 for success, and 0 for failure.

## Examples

The following code gets business details from business keys obtained by a `findBusiness` call on an instantiated `uddiproxy` object (*proxy*):

```
int i, count, count2
string businessName[], businessDescription[]
string businessKey []
string serviceName[], serviceDescription[]
string serviceKey [], wsdl [ ]
...//set search options and inquiry URL
proxy.findbusiness ("IBM", count, businessName, &
    businessDescription, businessKey)
FOR i = 1 TO count
    proxy.getbusinessdetail (businessKey [i], count2, &
        serviceName, serviceDescription, serviceKey, wsdl)
    ...//call findService in secondary FOR/NEXT loop
NEXT
```

## findService

## Description

Finds service details using a service name.

## Syntax

`proxy.findService` (readonly string *serviceName*, ref integer *count*, ref string *serviceNameResult* [ ], ref string *serviceDescriptionResult* [ ], ref string *serviceKeyResult* [ ], ref string *busNameResult* [ ], ref string *wsdl* [ ])

Argument	Description
<i>proxy</i>	The name of the UDDIProxy object
<i>serviceName</i>	Service name to search in UDDI registry
<i>count</i>	Number of search results returned; never larger than the <i>maxRow</i> input parameter in a corresponding <code>setOption</code> call
<i>serviceNameResult</i>	Array of services matching the search criteria
<i>serviceDescriptionResult</i>	Array of descriptions for services matching the search criteria
<i>serviceKeyResult</i>	Array of globally unique identifiers (GUIDs) for each service matching the search criteria
<i>busNameResult</i>	Array of business names matching the search criteria
<i>wsdl</i>	Array of WSDL file names for services matching search criteria

## Return value

Integer. Valid values are 1 for success, and 0 for failure.

Examples

The following code gets service details for the “Weather” service using an instantiated uddiproxy object (*proxy*):

```
int ret, count
string serviceName[], serviceDescription[]
string serviceKey [], businessName [], wsdl [ ]
ret = proxy.findService("Weather", count, serviceName,&
    serviceDescription, serviceKey, businessName, wsdl)
```



# PowerBuilder Document Object Model

About this chapter

This chapter presents an overview of the PowerBuilder Document Object Model (PBDOM). For more information about using PBDOM, see the chapter on using XML services in *Application Techniques*.

Contents

Topic	Page
About PBDOM	59
PBDOM objects	61

## About PBDOM

PBDOM is the PowerBuilder implementation of the Document Object Model (DOM), a programming interface defining the means by which XML documents can be accessed and manipulated.

Although PBDOM is not an implementation of the World Wide Web Consortium (W3C) DOM API, it is very similar. The PBDOM PowerBuilder API can be used for reading, writing, and manipulating standard-format XML from within PowerScript code. PBDOM portrays an XML document as a collection of interconnected objects and provides intuitive methods indicating the use and functionality of each object.

PBDOM is also similar to JDOM, which is a Java-based document object model for XML files.

For more information about W3C DOM, go to the W3C Document Object Model Web site at <http://www.w3.org/DOM/>. For more information about JDOM, go to the JDOM Web site at <http://www.jdom.org>.

## Node trees

PBDOM interacts with XML documents according to a tree-view model consisting of parent and child nodes. A document element represents the top-level node of a standalone XML document. This element has one or many child nodes that represent the branches of the tree. You access nodes in the node tree through the appropriate class methods.

## XML parser

The PBDOM XML parser is used to load and parse an XML document, and also to generate XML based on user-specified DOM nodes.

PBDOM provides the methods you need to traverse the node tree, access the nodes and attribute values (if any), insert and delete nodes, and serialize the node tree back to XML.

## Objects and methods

The PBDOM object hierarchy is described in "PBDOM objects" next. The methods for each object are described in the following chapters. The chapters are arranged in alphabetical order for ease of reference.

Chapter 18, "PBDOM Summary," provides quick reference tables showing the signatures of the methods defined in each PBDOM object. The tables are arranged in an order that reflects the object hierarchy shown in "Object hierarchy" on page 62.

## PBDOM objects

PBDOM\_OBJECT, the base class for PBDOM objects that represent XML nodes, inherits from the PowerBuilder NonVisualObject class. PBDOM represents node types by the following classes:

- PBDOM\_ATTRIBUTE
- PBDOM\_CDATA
- PBDOM\_CHARACTERDATA
- PBDOM\_COMMENT
- PBDOM\_DOCTYPE
- PBDOM\_DOCUMENT
- PBDOM\_ELEMENT
- PBDOM\_ENTITYREFERENCE
- PBDOM\_PROCESSINGINSTRUCTION
- PBDOM\_TEXT

You use methods from these classes to access objects in a PBDOM node tree.

The PBDOM\_BUILDER class does not represent DOM nodes but can be used to build a PBDOM object tree from XML. It inherits from the PowerBuilder NonVisualObject class.

The PBDOM\_EXCEPTION class inherits from the PowerBuilder Exception class and provides a method that obtains error codes.

Each of these classes and their methods are described in the chapters that follow.

Comparing PBDOM objects with W3C DOM and JDOM objects

The following table shows the W3C DOM and JDOM objects that correspond to each PBDOM object that represents a node in the DOM tree. Note that although these W3C DOM and JDOM objects correspond to PBDOM objects, they are not equivalent to the PBDOM objects.

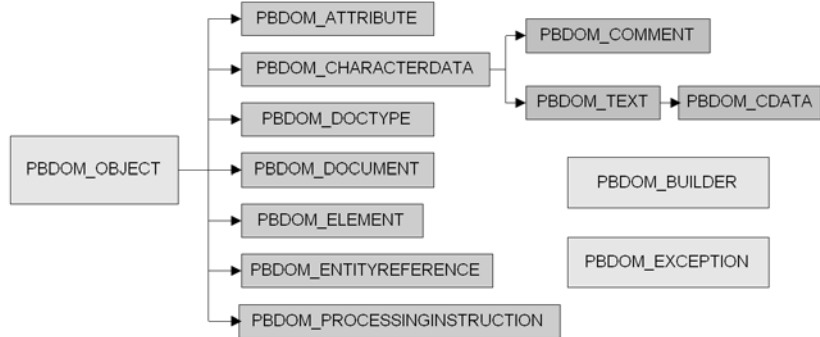
**Table 4-1: W3C DOM and JDOM objects that correspond to PBDOM objects**

PBDOM	W3C DOM	JDOM
PBDOM_ATTRIBUTE	ATTRIBUTE_NODE	Attribute
PBDOM_BUILDER	None	DOMBuilder
PBDOM_CDATA	CDATA_SECTION_NODE	CDATA
PBDOM_CHARACTERDATA	CHARACTER_DATA_NODE	None
PBDOM_COMMENT	COMMENT_NODE	Comment
PBDOM_DOCUMENT	DOCUMENT_NODE	Document
PBDOM_DOCTYPE	DOCUMENT_TYPE_NODE	DocType
PBDOM_ELEMENT	ELEMENT_NODE	Element
PBDOM_ENTITYREFERENCE	ENTITY_REFERENCE_NODE	EntityRef
PBDOM_OBJECT	NODE	None
PBDOM_PROCESSINGINSTRUCTION	PROCESSING_INSTRUCTION_NODE	Processinginstruction
PBDOM_TEXT	TEXT_NODE	Text

Object hierarchy

The W3C DOM and JDOM object hierarchies also differ from the PBDOM object hierarchy, which is shown in the following illustration.

**Figure 4-1: The PBDOM object hierarchy**



For more information about working with PBDOM, see the chapter on PowerBuilder XML services in *Application Techniques*.

About this chapter

This chapter describes the PBDOM\_ATTRIBUTE class.

## PBDOM\_ATTRIBUTE

Description

The PBDOM\_ATTRIBUTE class defines the behavior for an XML attribute, modeled in PowerScript. Its methods allow you to obtain the value of the attribute as well as namespace information.

A PBDOM\_ATTRIBUTE contains a subtree of child PBDOM\_OBJECTS. These children can be a combination of PBDOM\_TEXT and PBDOM\_ENTITYREFERENCE objects.

---

### **PBDOM\_ATTRIBUTE has no parent.**

A PBDOM\_ATTRIBUTE does not have a parent. However, it does have an owner PBDOM\_ELEMENT. Use the `GetOwnerElementObject` and `SetOwnerElementObject` to get and set the owner.

---

For more information about the PBDOM\_ATTRIBUTE object, including its default PBDOM\_TEXT object and its behavior with respect to XML namespaces, see the chapter on using XML services in *Application Techniques*.

Methods

Some of the inherited methods from PBDOM\_OBJECT serve no meaningful objective and only default or trivial functionalities result. These are described in the following table:

<b>Method</b>	<b>Always returns</b>
<code>GetParentObject</code>	null
<code>SetParentObject</code>	The current PBDOM_ATTRIBUTE, returned unmodified as a PBDOM_OBJECT

AddContent	GetUIntValue
Clone	GetTimeValue
Detach	GetUlongValue
Equals	HasChildren
GetBooleanValue	InsertContent
GetContent	IsAncestorObjectOf
GetDateValue	RemoveContent
GetDateTimeValue	SetBooleanValue
GetDoubleValue	SetContent
GetIntValue	SetDateValue
GetLongValue	SetDateTimeValue
GetName	SetDoubleValue
GetNamespacePrefix	SetIntValue
GetNamespaceUri	SetLongValue
GetObjectClass	SetName
GetObjectClassString	SetNamespace
GetOwnerDocumentObject	SetOwnerElementObject
GetOwnerElementObject	SetRealValue
GetQualifiedName	SetText
GetRealValue	SetTimeValue
GetText	SetUIntValue
GetTextNormalize	SetUlongValue
GetTextTrim	

PBDOM\_ATTRIBUTE has the following methods:

## AddContent

**Description** Adds the input PBDOM\_OBJECT as a child of the PBDOM\_ATTRIBUTE.

**Syntax** *pbdom\_attribute\_name*.AddContent(*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to add

**Return value** PBDOM\_OBJECT. The PBDOM\_ATTRIBUTE modified.

**Throws** EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT is not a PBDOM\_TEXT or PBDOM\_ENTITYREFERENCE object.

EXCEPTION\_USE\_OF\_UNNAMED\_OBJECT – If the input PBDOM\_OBJECT has not been given a user-defined name.

**Usage** *pbdom\_object\_ref* must be a reference to a PBDOM\_TEXT or PBDOM\_ENTITYREFERENCE object.

**See also** GetContent, InsertContent, RemoveContent, SetContent

## Clone

**Description** Creates a clone of the PBDOM\_ATTRIBUTE object.

**Syntax** *pbdom\_attribute\_name*.Clone(boolean *bDeep*)

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE.
<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are true for a deep clone and false for a shallow clone.

**Return value** PBDOM\_OBJECT. A clone of this PBDOM\_ATTRIBUTE returned as a PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_ATTRIBUTE object's internal implementation is null. The occurrence of this exception is rare but can take place if severe memory corruption occurs.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If this PBDOM\_ATTRIBUTE does not have or has not been assigned a user-defined name.

Examples

This example creates a PBDOM\_DOCUMENT from the string <abc My\_Attr="An Attribute"/>, gets the attribute from the root element, and creates a shallow clone and a deep clone from it. For the shallow clone, an empty string is returned in the message box. For the deep clone, the string An Attribute is returned:

```

PBDOM_BUILDER      pbdom_buildr
PBDOM_DOCUMENT     pbdom_doc
PBDOM_ATTRIBUTE    pbdom_attr
PBDOM_ATTRIBUTE    pbdom_attr_clone_deep
PBDOM_ATTRIBUTE    pbdom_attr_clone_shallow
string strXML = "<abc My_Attr=~"An Attribute~/>"

TRY
    pbdom_buildr = Create PBDOM_BUILDER
    pbdom_doc = pbdom_buildr.BuildFromString(strXML)
    pbdom_attr = pbdom_doc.GetRootElement(). &
        GetAttribute("My_Attr")
    pbdom_attr_clone_shallow = pbdom_attr.Clone(false)
    MessageBox ("Shallow Attribute Clone Text", &
        pbdom_attr_clone_shallow.GetText())
    pbdom_attr_clone_deep = pbdom_attr.Clone(true)
    MessageBox ("Deep Attribute Clone Text", &
        pbdom_attr_clone_deep.GetText())

    CATCH (PBDOM_EXCEPTION pbdom_except)
        MessageBox ("PBDOM_EXCEPTION", &
            pbdom_except.GetMessage())
END TRY

```

Usage

The Clone method creates and returns a duplicate of the current PBDOM\_ATTRIBUTE.

If a shallow clone is requested, this method clones the original PBDOM\_ATTRIBUTE together with its namespace information values. The subtree of child PBDOM\_TEXT and/or PBDOM\_ENTITYREFERENCE objects is not cloned.

If a deep clone is requested, this method additionally recursively clones the subtree under the PBDOM\_ATTRIBUTE. This subtree consists of a combination of PBDOM\_TEXT and PBDOM\_ENTITYREFERENCE objects that are the legal children of a PBDOM\_ATTRIBUTE.

A PBDOM\_ATTRIBUTE clone has no parent. However, the clone resides in the same PBDOM\_DOCUMENT as its original, and if the original PBDOM\_ATTRIBUTE is standalone, the clone is standalone.



## Detach

**Description** Detaches a PBDOM\_ATTRIBUTE from its owner PBDOM\_OBJECT, a PBDOM\_ELEMENT.

**Syntax** `pbdom_attribute_name.Detach()`

Argument	Description
<code>pbdom_attribute_name</code>	The name of the PBDOM_ATTRIBUTE

**Return value** PBDOM\_OBJECT. The PBDOM\_ATTRIBUTE object detached from its owner object.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_ATTRIBUTE object's internal implementation is null. The occurrence of this exception is rare but can take place if severe memory corruption occurs.

**Examples** The Detach method can be used to manipulate an XML document as follows:

```
PBDOM_BUILDER          pbdombuilder_new
PBDOM_DOCUMENT         pbdom_doc
PBDOM_ATTRIBUTE        pbdom_attr
PBDOM_ELEMENT          pbdom_elem
string strXML = "<abc My_Attr=~"My Attribute
Value~"><data>Data</data></abc>"

TRY
    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString (strXML)

    pbdom_attr = pbdom_doc.GetRootElement(). &
        GetAttribute("My_Attr")
    pbdom_attr.Detach()

    pbdom_elem = pbdom_doc.GetRootElement(). &
        GetChildElement("data")
    pbdom_elem.SetAttribute (pbdom_attr)

    Destroy pbdombuilder_new
    Destroy pbdom_doc

CATCH (PBDOM_Exception except)
    MessageBox ("Exception Occurred", except.Text)
END TRY
```

Here, the PBDOM\_Builder BuildFromString method is used to create the following PBDOM\_DOCUMENT object, pbdom\_doc, using an XML string:

```
<abc My_Attr="My Attribute Value">  
  <data>Data </data>  
</abc>
```

The GetAttribute method is used to obtain the attribute from the root element of pbdom\_doc. This value is assigned to the PBDOM\_ATTRIBUTE object pbdom\_attr. The pbdom\_attr object is detached from its parent element, and the data element is obtained from pbdom\_doc using the GetChildElement method. The data element is then assigned to the PBDOM\_ELEMENT object pbdom\_elem. The attribute assigned to pbdom\_attr is assigned to pbdom\_elem, yielding the following modified pbdom\_doc:

```
<abc>  
  <data My_Attr="My Attribute Value">Data</data>  
</abc>
```

### Usage

If the PBDOM\_ATTRIBUTE object has no owner PBDOM\_ELEMENT, the Detach method does nothing.

## Equals

**Description** Tests for equality between the supplied PBDOM\_OBJECT and the PBDOM\_ATTRIBUTE from which the method is invoked.

**Syntax** `pbdom_attribute_name.Equals(pbdom_object pbdom_object_ref)`

Argument	Description
<code><i>pbdom_attribute_name</i></code>	The name of the PBDOM_ATTRIBUTE
<code><i>pbdom_object_ref</i></code>	A PBDOM_OBJECT to be compared

**Return value** Boolean. Returns true if the current PBDOM\_ATTRIBUTE is equivalent to the input PBDOM\_OBJECT and false otherwise.

**Throws** EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If this PBDOM\_ATTRIBUTE does not have or has not been assigned a user-defined name.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – if the input PBDOM\_OBJECT is not a reference to an object derived from PBDOM\_OBJECT.

**Examples** **Example 1** The following code uses the Equals method to test for equivalence between a referenced PBDOM\_OBJECT and a cloned object.

```

pbdom_attr = Create PBDOM_Attribute
pbdom_attr.SetName("My_Attr")
pbdom_attr_clone = pbdom_attr.Clone(true)

if (pbdom_attr_clone.Equals(pbdom_attr)) then
    MessageBox ("Equals", "Yes")
else
    MessageBox ("Equals", "No")
end if

```

The SetName method names the newly created PBDOM\_ATTRIBUTE, which is subsequently cloned with the Clone method. The Equals method tests for equality between the cloned PBDOM\_ATTRIBUTE `pbdom_attr_clone` and the referenced PBDOM\_OBJECT `pbdom_attr`. A message box displays the result returned from the Equals method.

Note here that because a cloned object is never equivalent to the object from which it is cloned, the Equals method returns false.

**Example 2** The following code uses the Equals method to test for equivalence between two cloned objects.

```

pbdom_attr = Create PBDOM_Attribute
pbdom_attr.SetName("My_Attr")
pbdom_attr_clone = pbdom_attr.Clone(true)
pbdom_attr_2 = pbdom_attr_clone

if (pbdom_attr_clone.Equals(pbdom_attr_2)) then
    MessageBox ("Equals", "Yes")
else
    MessageBox ("Equals", "No")
end if
    
```

A newly created PBDOM\_ATTRIBUTE is cloned, and a reference to this clone is assigned to pbdom\_attr\_2. The Equals method tests for equality between the cloned PBDOM\_ATTRIBUTE pbdom\_attr\_clone and the reference to it, pbdom\_attr\_2. A message box displays the result returned from the Equals method.

Here the Equals method returns true.

Usage

Note that the clone of a PBDOM\_ATTRIBUTE is not considered equal to itself.

## GetBooleanValue

Description

Obtains the value of a PBDOM\_ATTRIBUTE object in boolean form.

Syntax

```
pbdom_attribute_name.GetBooleanValue()
```

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE

Return value

Boolean.

The following table lists the PBDOM\_ATTRIBUTE string values that are accepted as boolean and the corresponding return values from the GetBooleanValue method.

PBDOM_ATTRIBUTE string value	GetBooleanValue
1	true
0	false
TRUE	true
FALSE	false

PBDOM_ATTRIBUTE string value	GetBooleanValue
ON	true
OFF	false
YES	true
NO	false

Strings are treated without case sensitivity. If no conversion can occur, the `GetBooleanValue` method throws an exception.

Throws

`EXCEPTION_DATA_CONVERSION` – If data conversion fails.

Examples

The `GetBooleanValue` can be used to evaluate a `PBDOM_ATTRIBUTE` object as follows:

```
PBDOM_BUILDER          pbombuilder_new
PBDOM_DOCUMENT         pbdom_doc
PBDOM_ATTRIBUTE        pbdom_attr
string strXML = "<abc My_Boolean_Attribute
=~"on~"><data
An_Attribute=~"Some Text~">Data</data></abc>"

TRY
  pbdombuilder_new = Create PBDOM_Builder
  pbdom_doc = pbdombuilder_new.BuildFromString (strXML)

  pbdom_attr = pbdom_doc.GetRootElement(). &
    GetAttribute("My_Boolean_Attribute")

  MessageBox ("Boolean Value", &
    string(pbdom_attr.GetBooleanValue()))

  Destroy pbdombuilder_new
  Destroy pbdom_doc
CATCH (PBDOM_Exception except)
  MessageBox ("Exception Occurred", except.Text)
END TRY
```

The `BuildFromString` method is used to create a `PBDOM_DOCUMENT` object, `pbdom_doc`, using an XML string. The attribute value of the root element of `pbdom_doc` is assigned to the `PBDOM_ATTRIBUTE` object `pbdom_attr`. The attribute value, `on`, is evaluated with the `GetBooleanValue` method. A message box reports the return value of the `GetBooleanValue` method.

See also

`SetBooleanValue`

## GetContent

**Description** Returns an array of PBDOM\_OBJECT objects that are the children of the PBDOM\_ATTRIBUTE. The children of a PBDOM\_ATTRIBUTE can be only PBDOM\_TEXT or PBDOM\_ENTITYREFERENCE objects.

**Syntax** `pbdom_attribute_name.GetContent(ref pbdom_object pbdom_object_array[ ])`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>pbdom_object_array</i>	The referenced name of an array of PBDOM_OBJECTs that receives PBDOM_OBJECTs

**Return value** Boolean. This method always returns true.

**See also** AddContent, InsertContent, RemoveContent, SetContent

## GetDateValue

**Description** Returns the value of a PBDOM\_ATTRIBUTE object as type Date.

**Syntax** `pbdom_attribute_name.GetDateValue(string strDateFormat)`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>strDateFormat</i>	The date format for the return value, for example, MM:DD:YYYY

The value of the *strDateFormat* parameter can use slashes or colons as delimiters. The following table illustrates characters with special meaning in *strDateFormat*.

Character	Meaning	Example
D	Day number with no leading zero	5
DD	Day number with leading zero, if applicable	05
M	Month number with no leading zero	5
MM	Month number with leading zero, if applicable	05
YY	Two-digit year number	05
YYYY	Four-digit year number	2005

**Return value** Date.

**Throws** EXCEPTION\_DATA\_CONVERSION – If data conversion fails.

**See also** SetDateValue

## GetDateTimeValue

**Description** Returns the value of a PBDOM\_ATTRIBUTE object as type DateTime.

**Syntax** `pbdom_attribute_name.GetDateTimeValue(string strDateFormat, string strTimeFormat)`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>strDateFormat</i>	The date format for the return value, for example, MM:DD:YYYY
<i>strTimeFormat</i>	The time format for the return value, for example, HH:MM:SS

The value of the *strDateFormat* parameter can use slashes or colons as delimiters. The following table illustrates characters that have special meaning in *strDateFormat*.

Character	Meaning	Example
D	Day number with no leading zero	5
DD	Day number with leading zero, if applicable	05
M	Month number with no leading zero	5
MM	Month number with leading zero, if applicable	05
YY	Two-digit year number	05
YYYY	Four-digit year number	2005

The value of the *strTimeFormat* parameter can use slashes or colons as delimiters. The following table illustrates characters that have special meaning in *strTimeFormat*.

Character	Meaning	Example
H	Hour number with no leading zero	5
HH	Hour number with leading zero, if applicable	05
M	Minutes number with no leading zero	5
MM	Minutes number with leading zero, if applicable	05
S	Seconds number with no leading zero	5
SS	Seconds number with leading zero, if applicable	55

**Return value** DateTime.

**Throws** EXCEPTION\_DATA\_CONVERSION – If data conversion fails.

**See also** SetDateTimeValue

## GetDoubleValue

Description Returns the value of a PBDOM\_ATTRIBUTE object in double form.

Syntax *pbdom\_attribute\_name*.GetDoubleValue()

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE

Return value Double.

Throws EXCEPTION\_DATA\_CONVERSION – If data conversion fails.

Usage Throws exception\_data\_conversion if the method fails to convert data.

See also SetDoubleValue

## GetIntValue

Description Returns the value of a PBDOM\_ATTRIBUTE object as type int.

Syntax *pbdom\_attribute\_name*.GetIntValue()

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE

Return value Int.

Throws EXCEPTION\_DATA\_CONVERSION – If data conversion fails.

See also SetIntValue

## GetLongValue

Description Returns the value of a PBDOM\_ATTRIBUTE object as type long.

Syntax *pbdom\_attribute\_name*.GetLongValue()

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE

Return value Long.

Throws EXCEPTION\_DATA\_CONVERSION – If data conversion fails.

See also SetLongValue



## GetName

**Description** Retrieves the local name of the PBDOM\_ATTRIBUTE object.

**Syntax** `pbdom_attribute_name.GetName()`

Argument	Description
<code>pbdom_attribute_name</code>	The name of the PBDOM_ATTRIBUTE

**Return value** String.

**Throws** EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If this PBDOM\_ATTRIBUTE does not have or has not been assigned a user-defined name.

**Examples** **Example 1** When the GetName method is invoked for the attribute name in the following element, it returns the string ATTRIBUTE\_1:

```
<abc ATTRIBUTE_1="My Attribute">
```

**Example 2** When the GetName method is invoked for the name of the eMusic:Type attribute in the following element, it returns the string Type:

```
<eMusic:CD
xmlns:eMusic="http://www.eMusic_Records.com"
eMusic:Type="Jazz"/>
```

The namespace prefix is not part of the return string.

**Usage** For an XML attribute that appears in the form `[namespace_prefix]:[attribute_name]`, the local attribute name is `attribute_name`. Where the XML attribute has no namespace prefix, the local name is simply the attribute name.

Use the GetNamespacePrefix method to obtain the namespace prefix for a PBDOM\_ATTRIBUTE object. Use the GetQualifiedName method to obtain the fully qualified name for a PBDOM\_ATTRIBUTE object.

**See also** GetNamespacePrefix  
GetNamespaceUri  
GetQualifiedName  
SetName  
SetNamespace

## GetNamespacePrefix

**Description** Obtains the namespace prefix of a PBDOM\_ATTRIBUTE object. The GetNamespacePrefix method returns an empty string if the PBDOM\_ATTRIBUTE has no namespace.

**Syntax** `pbdom_attribute_name.GetNamespacePrefix()`

<b>Argument</b>	<b>Description</b>
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE

**Return value** String

For a PBDOM\_ATTRIBUTE object that has the form [*namespacePrefix*]:[*attributeName*], the namespace prefix is [*namespacePrefix*].

**See also** GetNamespaceUri  
GetQualifiedName  
SetName  
SetNamespace

## GetNamespaceUri

**Description** Obtains the namespace URI of a PBDOM\_ATTRIBUTE object. The GetNamespaceUri method returns an empty string if the PBDOM\_ATTRIBUTE has no namespace.

**Syntax** `pbdom_attribute_name.GetNamespaceUri()`

<b>Argument</b>	<b>Description</b>
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE

**Return value** String.

**See also** GetNamespacePrefix  
GetQualifiedName  
SetName  
SetNamespace

## GetObjectClass

Description	Returns a long integer code that indicates the class of the current PBDOM_OBJECT.				
Syntax	<i>pbdom_object_name</i> .GetObjectClass() <table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_object_name</i></td> <td>The name of a PBDOM_OBJECT</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT
Argument	Description				
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT				
Return value	Long. GetObjectClass returns a long integer code that indicates the class of the current PBDOM_OBJECT. If <i>pbdom_object_name</i> is a PBDOM_ATTRIBUTE, the returned value is 5.				
Examples	This example illustrates polymorphism: <i>pbdom_obj</i> is declared as PBDOM_OBJECT but instantiated as PBDOM_ATTRIBUTE. A message box returns the result of the GetObjectClass method invoked for PBDOM_ATTRIBUTE. Here the result is 5, indicating that <i>pbdom_obj</i> is a PBDOM_ATTRIBUTE object. <pre>PBDOM_OBJECT pbdom_obj  pbdom_obj = Create PBDOM_ATTRIBUTE MessageBox ("Class", &amp;     string(pbdom_obj.GetObjectClass()))</pre>				
Usage	This method can be used for diagnostic purposes to dynamically determine the type of a PBDOM_OBJECT at runtime.				
See also	GetObjectClassString				

## GetObjectClassString

Description	Returns a string form of the class of the PBDOM_OBJECT.				
Syntax	<i>pbdom_object_name</i> .GetObjectClassString() <table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_object_name</i></td> <td>The name of a PBDOM_OBJECT</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT
Argument	Description				
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT				
Return value	String. GetObjectClassString returns a string that indicates the class of the current PBDOM_OBJECT. If <i>pbdom_object_name</i> is a PBDOM_ATTRIBUTE, the returned string is “pbdom_attribute”.				
Examples	The GetObjectClass method returns a string specific to the class of the object from which the method is invoked.				

This example illustrates polymorphism: `pbdom_obj` is declared as `PBDOM_OBJECT` but instantiated as `PBDOM_ATTRIBUTE`. A message box returns the result of the `GetObjectClassString` method invoked for `PBDOM_ATTRIBUTE`. Here the result is `pbdom_attribute`, indicating that `pbdom_obj` is a `PBDOM_ATTRIBUTE` object.

```
PBDOM_OBJECT pbdom_obj

pbdom_obj = Create PBDOM_ATTRIBUTE
MessageBox ("Class", pbdom_obj.GetObjectClassString())
```

- Usage                      This method can be used for diagnostic purposes to dynamically determine the actual type of a `PBDOM_OBJECT` at runtime.
- See also                   `GetObjectClass`

## GetOwnerDocumentObject

Description               Returns the `PBDOM_DOCUMENT` object that owns the `PBDOM_ATTRIBUTE`.

Syntax                    `pbdom_attribute_name.GetObjectDocumentObject()`

Argument	Description
<code>pbdom_attribute_name</code>	The name of the <code>PBDOM_ATTRIBUTE</code>

Return value              `PBDOM_DOCUMENT`. The `PBDOM_DOCUMENT` that owns the `PBDOM_ATTRIBUTE` object from which the `GetOwnerDocumentObject` method is invoked.

A return value of null indicates the `PBDOM_ATTRIBUTE` object is not owned by any `PBDOM_DOCUMENT`.

Examples                   The `GetOwnerDocumentObject` method can be used to identify the `PBDOM_DOCUMENT` object that owns a `PBDOM_ATTRIBUTE` object.

Here, the `BuildFromString` method is used to create the following `PBDOM_DOCUMENT` object, `pbdom_doc`, using an XML string:

```
<abc My_Attr="My Attribute Value">
  <data>Data </data>
</abc>
```

The `GetAttribute` method is used to obtain the attribute from the root element of `pbdom_doc`. This value is assigned to the `PBDOM_ATTRIBUTE` object `pbdom_attr`. The `GetOwnerDocumentObject` method is used to obtain the `pbdom_doc` that owns `pbdom_attr`. The result of the `GetOwnerDocumentObject` method is assigned to the `PBDOM_DOCUMENT` object `pbdom_doc_2`. Then `pbdom_doc_2` is compared to `pbdom_doc` using the `Equals` method, and the result is displayed in a message box.

```

PBDOM_Builder pbdombuilder_new
pbdom_document pbdom_doc
pbdom_document pbdom_doc_2
PBDOM_ATTRIBUTE pbdom_attr
string strXML = "<abc My_Attr=~"My Attribute
Value~"><data>Data </data></abc>"

TRY
    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString (strXML)

    pbdom_attr = pbdom_doc.GetRootElement(). &
        GetAttribute("My_Attr")
    pbdom_doc_2 = pbdom_attr.GetOwnerDocumentObject()

    if (pbdom_doc.Equals(pbdom_doc_2)) then
        MessageBox ("Equals", "pbdom_doc equals " &
            + "pbdom_attr.GetOwnerDocumentObject()")
    end if

    Destroy pbdombuilder_new

CATCH (PBDOM_Exception except)
    MessageBox ("Exception Occurred", except.Text)
END TRY

```

See also

`GetOwnerElementObject`  
`SetOwnerElementObject`

## GetOwnerElementObject

**Description** Returns the owner PBDOM\_ELEMENT of this PBDOM\_ATTRIBUTE. If there is no owner element, null is returned.

**Syntax** `pbdom_attribute_name.GetOwnerElementObject()`

<b>Argument</b>	<b>Description</b>
<code>pbdom_attribute_name</code>	The name of the PBDOM_ATTRIBUTE

**Return value** PBDOM\_ELEMENT. The owner PBDOM\_ELEMENT of this PBDOM\_ATTRIBUTE or null if this PBDOM\_ATTRIBUTE has no owner element.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_ATTRIBUTE object's internal implementation is null. The occurrence of this exception is rare but can take place if severe memory corruption occurs.

**Examples** This example creates a PBDOM\_DOCUMENT from a string strXML in which the abc root element contains one attribute, My\_Attr. The code gets this attribute, calls GetOwnerElementObject on it to obtain the owner element, then calls GetName to return the string abc. Finally, it sets My\_Attr as an attribute of the child element Data:

```
PBDOM_BUILDER      pbdombuilder_new
PBDOM_DOCUMENT     pbdom_doc
PBDOM_ATTRIBUTE    pbdom_attr
PBDOM_ELEMENT      pbdom_elem
string strXML = "<abc My_Attr=~\"My Attribute
Value~\"><data>Data</data></abc>"

TRY
    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString (strXML)

    // Get the attribute
    pbdom_attr = pbdom_doc.GetRootElement(). &
        GetAttribute("My_Attr")

    MessageBox ("pbdom_attr Owner Element Name", &
        pbdom_attr.GetOwnerElementObject().GetName())

    pbdom_attr.Detach()

    pbdom_elem = pbdom_doc.GetRootElement(). &
        GetChildElement("data")
    pbdom_elem.SetAttribute (pbdom_attr)
```

```

        MessageBox ("pbdom_attr Owner Element Name", &
            pbdom_attr.GetOwnerElementObject().GetName())

        Destroy pbdombuilder_new
        Destroy pbdom_doc

    CATCH (PBDOM_Exception except)
        MessageBox ("Exception Occurred", except.Text)
    END TRY

```

See also [SetOwnerElementObject](#)

## GetQualifiedName

**Description** Obtains the qualified name of a PBDOM\_ATTRIBUTE. The GetQualifiedName method returns the local name for a PBDOM\_ATTRIBUTE that has no namespace.

**Syntax** `pbdom_attribute_name.GetQualifiedName()`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE

**Return value** String.

**Usage** For a PBDOM\_ATTRIBUTE object that has the form [*namespacePrefix*]:[*attributeName*], the qualified name for the PBDOM\_ATTRIBUTE consists of the entire name, [*namespacePrefix*], and [*attributeName*].

To obtain the local name of the PBDOM\_ATTRIBUTE, use the GetName method.

To obtain the namespace prefix for the PBDOM\_ATTRIBUTE, use the GetNamespacePrefix method.

See also [GetName](#)  
[GetNamespacePrefix](#)  
[GetNamespaceUri](#)  
[SetName](#)  
[SetNamespace](#)

## GetRealValue

Description	Returns the value of a PBDOM_ATTRIBUTE object as type real.				
Syntax	<i>pbdom_attribute_name</i> .GetRealValue() <table><thead><tr><th>Argument</th><th>Description</th></tr></thead><tbody><tr><td><i>pbdom_attribute_name</i></td><td>The name of the PBDOM_ATTRIBUTE</td></tr></tbody></table>	Argument	Description	<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
Argument	Description				
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE				
Return value	Real.				
Throws	EXCEPTION_DATA_CONVERSION – If data conversion fails.				
Usage	GetRealValue is the exact counterpart of the JDOM getFloatValue method.				
See also	SetRealValue				

## GetText

Description	Returns the text value of the PBDOM_ATTRIBUTE object.				
Syntax	<i>pbdom_attribute_name</i> .GetText() <table><thead><tr><th>Argument</th><th>Description</th></tr></thead><tbody><tr><td><i>pbdom_attribute_name</i></td><td>The name of the PBDOM_ATTRIBUTE</td></tr></tbody></table>	Argument	Description	<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
Argument	Description				
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE				
Return value	String.				
Throws	EXCEPTION_USE_OF_UNNAMED_PBDOM_OBJECT – If this PBDOM_ATTRIBUTE does not have or has not been assigned a user-defined name.				
Examples	<b>Example 1</b> The GetText method is invoked for the attribute in the following element:				

```
<abc ATTRIBUTE_1="My Attribute">
```

The GetText method returns the following string:

```
My Attribute
```

**Example 2** This example sets an attribute called `my_attr` for the root element with text value `text part`. A PBDOM\_ENTITYREFERENCE with the name `ent_ref` and a PBDOM\_TEXT with the text value `text part` again are then added as part of the contents of `my_attr`. A call to GetText on `my_attr` returns the following text:

```
"text part &ent_ref; text part again."
```



The entity reference `&ent_ref;` is not expanded. If an entity reference is included in an input XML document that is parsed, then the entity reference is expanded before the XML document is transformed into a DOM tree in memory.

```

PBDOM_DOCUMENT          pbdom_doc
PBDOM_ATTRIBUTE         pbdom_attr
PBDOM_ENTITYREFERENCE   pbdom_entref
PBDOM_TEXT              pbdom_txt

try
    pbdom_doc = Create PBDOM_DOCUMENT
    pbdom_entref = Create PBDOM_ENTITYREFERENCE
    pbdom_txt = Create PBDOM_TEXT

    // Create a new document object.
    pbdom_doc.NewDocument ("root")

    // Set the text of "pbdom_txt".
    pbdom_txt.SetText (" text part again.")

    // Add an attribute "my_attr" to the root element.
    pbdom_doc.GetRootElement().SetAttribute("my_attr", &
        "text part ")

    // Set the name of the PBDOM_ENTITYREFERENCE.
    pbdom_entref.SetName ("ent_ref")

    // Append the entity reference to the root
    // element's "my_attr" attribute.
    pbdom_doc.GetRootElement(). &
        GetAttribute("my_attr").AddContent(pbdom_entref)

    // Append a new text node to the "my_attr" attribute.
    pbdom_doc.GetRootElement(). &
        GetAttribute("my_attr").AddContent (pbdom_txt)

    // Now test the text contents of "my_attr "
    if pbdom_doc.GetRootElement(). &
        GetAttribute("my_attr").GetText() = &
            "text part &ent_ref; text part again." then
        MessageBox ("Pass", &
            "GetText() on my_attr is correct.")
    else
        MessageBox ("Fail", &
            "GetText() on my_attr is incorrect.")

```

```

        end if

        catch (pbdom_exception pbdom_e)
            MessageBox ("PBDOM_EXCEPTION", pbdom_e.GetMessage())
        end try
    
```

**Usage** This method returns the actual textual value of this PBDOM\_ATTRIBUTE, including all text within the quotation marks. If there are any PBDOM\_ENTITYREFERENCE objects included within the PBDOM\_ATTRIBUTE, the PBDOM\_ENTITYREFERENCE object's name is returned together with the leading ampersand ('&') character plus the terminating semicolon character(';').

**See also** GetTextNormalize  
 GetTextTrim  
 SetText

## GetTextNormalize

**Description** Returns the text data contained within a PBDOM\_ATTRIBUTE object with surrounding whitespace characters removed and internal whitespace characters replaced by a single space.

**Syntax** `pbdom_attribute_name.GetTextNormalize()`

Argument	Description
<code>pbdom_attribute_name</code>	The name of the PBDOM_ATTRIBUTE

**Return value** String.

**Examples** **Example 1** The GetTextNormalize method is invoked for the PBDOM\_ATTRIBUTE of the following element:

```
<abc ATTRIBUTE_1=" My Attribute ">
```

The GetTextNormalize method returns the following string:

```
My Attribute
```

**Example 2** This example creates a PBDOM\_DOCUMENT based on the following DOM tree, which has a Tab character between the words “My” and “Attribute” in the My\_Attr attribute, specified by the &#9; entity reference. There are also several space characters:

```
<abc My_Attr="My&#9;Attribute Value ">
    <data>Data</data>
</abc>
```

The call to `GetAttribute` stores `My_Attr` in `pbdom_attr`. Calling `GetText` on `pbdom_attr` returns the entire string content of `My_Attr`, including the beginning Tab character. Calling `GetTextNormalize` returns the string with all surrounding whitespace characters removed, and the whitespace characters between the words, including the Tab character, replaced by a single space.

```

PBDOM_BUILDER      pbdombuilder_new
PBDOM_DOCUMENT     pbdom_doc
PBDOM_ATTRIBUTE    pbdom_attr
string strXML = "<abc My_Attr=~"My&#9;Attribute
Value ~"><data>Data</data></abc>"

TRY
    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString (strXML)

    pbdom_attr = pbdom_doc.GetRootElement(). &
        GetAttribute("My_Attr")

    MessageBox ("pbdom_attr text", "[" &
        "+ pbdom_attr.GetText() + "]")
    MessageBox ("pbdom_attr text normalize", "[" &
        "+ pbdom_attr.GetTextNormalize() + "]")

    Destroy pbdombuilder_new
    Destroy pbdom_doc

    CATCH (PBDOM_Exception except)
        MessageBox ("Exception Occurred", except.Text)
END TRY

```

#### Usage

Surrounding whitespace characters are removed from the returned text data, and internal whitespace characters are normalized to a single space. The `GetTextNormalize` method returns an empty string if no text value exists for the `PBDOM_ATTRIBUTE` or if the text value contains only whitespace characters.

If this `PBDOM_ATTRIBUTE` contains any `PBDOM_ENTITYREFERENCE` objects, the name of the `PBDOM_ENTITYREFERENCE` object is returned as part of the normalized string.

JDOM does not provide a `getTextNormalize` method for its `Attribute` class.

#### See also

[GetText](#)  
[GetTextTrim](#)  
[SetText](#)

## GetTextTrim

**Description** Returns the text data contained within a PBDOM\_ATTRIBUTE object with surrounding spaces removed.

**Syntax** `pbdom_attribute_name.GetTextTrim()`

Argument	Description
<code>pbdom_attribute_name</code>	The name of the PBDOM_ATTRIBUTE

**Return value** String.

**Examples** **Example 1** The `GetTextTrim` method is invoked for the PBDOM\_ATTRIBUTE of the following element:

```
<abc ATTRIBUTE_1=" My Attribute ">
```

The `GetTextNormalize` method returns the following string:

```
My Attribute
```

Note that the whitespace characters surrounding the string are removed, but the whitespace characters within the string remain.

**Example 2** This example builds a PBDOM\_DOCUMENT based on the following XML tree:

```
<abc My_Attr="&#32;&#32;&#32;My&#9;Attribute
Value&#32;&#32;&#32;">
  <data>Data</data>
</abc>
```

The `My_Attr` attribute contains an entity reference for a Tab character (`&#9;`) and several entity references for the space character (`&#32;`). The message boxes in the following code show that `GetText` returns the complete text string of the attribute, whereas `GetTextTrim` returns the string with the surrounding whitespace characters removed. The Tab character between the words is not removed:

```
PBDOM_BUILDER      pbdombuilder_new
PBDOM_DOCUMENT     pbdom_doc
PBDOM_ATTRIBUTE    pbdom_attr
string              strXML

TRY
    strXML = "<abc
My_Attr=~" &#32;&#32;&#32;My&#9;Attribute
Value&#32;&#32;&#32;~"><data>Data</data></abc>"
    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString (strXML)

    pbdom_attr = pbdom_doc.GetRootElement(). &
        GetAttribute("My_Attr")

    MessageBox ("pbdom_attr text", "[" &
        + "pbdom_attr.GetText() + "]")
    MessageBox ("pbdom_attr text normalize", &
        "[" + pbdom_attr.GetTextTrim() + "]")

Destroy pbdombuilder_new
Destroy pbdom_doc

CATCH (PBDOM_Exception except)
    MessageBox ("Exception Occurred", except.Text)
END TRY
```

#### Usage

Surrounding whitespace characters are removed from the returned text data. The `GetTextTrim` method returns an empty string if no text value exists for the `PBDOM_ATTRIBUTE` or if the text value contains only whitespace characters.

If this `PBDOM_ATTRIBUTE` contains any `PBDOM_ENTITYREFERENCE` objects, the name of the `PBDOM_ENTITYREFERENCE` object is returned as part of the trimmed string.

#### See also

`GetText`  
`GetTextNormalize`  
`SetText`

## GetTimeValue

**Description** Returns the value of a PBDOM\_ATTRIBUTE object as type Time.

**Syntax** *pbdom\_attribute\_name*.GetTimeValue(string *strTimeFormat*)

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>strTimeFormat</i>	The time format for the return value, for example, HH:MM:SS

The value of the *strTimeFormat* parameter can use slashes or colons as delimiters. The following table illustrates characters that have special meaning in *strTimeFormat*.

Character	Meaning	Example
H	Hour number with no leading zero	5
HH	Hour number with leading zero, if applicable	05
M	Minutes number with no leading zero	5
MM	Minutes number with leading zero, if applicable	05
S	Seconds number with no leading zero	5
SS	Seconds number with leading zero, if applicable	55

**Return value** Time.

**Throws** EXCEPTION\_DATA\_CONVERSION – If data conversion fails.

**See also** SetTimeValue

## GetUIntValue

**Description** Returns the value of a PBDOM\_ATTRIBUTE object as type UInt.

**Syntax** *pbdom\_attribute\_name*.GetUIntValue()

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE

**Return value** UInt.

**Throws** EXCEPTION\_DATA\_CONVERSION – If data conversion fails.

**See also** SetUIntValue

## GetUlongValue

Description Returns the value of a PBDOM\_ATTRIBUTE object as type Ulong.

Syntax *pbdom\_attribute\_name*.GetUlongValue()

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE

Return value Ulong.

Throws EXCEPTION\_DATA\_CONVERSION – If data conversion fails.

See also SetUlongValue

## HasChildren

Description Determines whether this PBDOM\_ATTRIBUTE object contains any child PBDOM\_OBJECTs.

Syntax *pbdom\_attribute\_name*.HasChildren()

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE

Return value Boolean. Returns true if this PBDOM\_ATTRIBUTE contains child objects and false otherwise.

Throws EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object is not associated with a derived PBDOM\_OBJECT class object.

**Examples**

This example creates a PBDOM\_DOCUMENT from a string. The XML document in the string already contains a root element named root that contains an attribute attr that contains an empty string. It then represents attr as a PBDOM\_ATTRIBUTE object and calls its HasChildren method, which returns true because a PBDOM\_ATTRIBUTE always contains at least one child object. After a call to GetContent, the message box shows that attr contains only one child, a PBDOM\_TEXT that represents the empty string:

```
PBDOM_BUILDER pbdom_buildr
PBDOM_DOCUMENT pbdom_doc
PBDOM_ATTRIBUTE pbdom_attr
string strXML = "<root attr=~\"~\"></root>"

try
    pbdom_buildr = Create PBDOM_BUILDER
    pbdom_doc = pbdom_buildr.BuildFromString(strXML)

    pbdom_attr = pbdom_doc.GetRootElement(). &
        GetAttribute("attr")

    if (pbdom_attr.HasChildren()) then
        PBDOM_OBJECT pbdom_obj_array[]
        long l = 0

        pbdom_attr.GetContent(pbdom_obj_array)

        for l = 1 to UpperBound (pbdom_obj_array)
            MessageBox ("Attr Child Object", &
                pbdom_obj_array[l].GetObjectClassString())
        next

    end if

catch (pbdom_exception pbdom_e)
    MessageBox ("PBDOM_EXCEPTION", pbdom_e.GetMessage())
end try
```

**Usage**

This method checks to see if this PBDOM\_ATTRIBUTE object contains any child PBDOM\_OBJECTs and returns true if it does. Note that according to the W3C DOM specification, a DOM Attribute Node can contain only Text and Entity Reference Nodes, therefore a PBDOM\_ATTRIBUTE object can contain only PBDOM\_TEXT and PBDOM\_ENTITYREFERENCE objects.

Even if a PBDOM\_ATTRIBUTE object's text value is an empty string, it always contains at least one PBDOM\_TEXT object that represents the empty string.



## InsertContent

**Description** Inserts a PBDOM\_OBJECT as a child of the PBDOM\_ATTRIBUTE at a position specified by a referenced PBDOM\_OBJECT.

**Syntax** `pbdom_attribute_name.InsertContent(pbdom_object pbdom_object_new, pbdom_object pbdom_object_ref)`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>pbdom_object_new</i>	The PBDOM_OBJECT to be inserted
<i>pbdom_object_ref</i>	A positional reference to a PBDOM_OBJECT before which <i>pbdom_object_new</i> is to be inserted

**Return value** PBDOM\_OBJECT. The PBDOM\_ATTRIBUTE returned as a PBDOM\_OBJECT.

**Throws** EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – The PBDOM\_OBJECT to be inserted is nameable and has not been given a user-defined name.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – The PBDOM\_OBJECT to be inserted already has a parent.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – The PBDOM\_OBJECT to be inserted is not valid to be inserted as a child of this PBDOM\_ATTRIBUTE.

EXCEPTION\_WRONG\_PARENT\_ERROR – The reference PBDOM\_OBJECT is not a child of this PBDOM\_ATTRIBUTE.

**Examples** This example adds an attribute to the root element with the name `my_attr` and text content “attribute text”. It then creates a PBDOM\_ENTITYREFERENCE object named `ent_ref` and inserts it before the attribute’s current content. Testing the new content of the attribute should return “&ent\_ref;attribute text”;

Consider the following code :

```
PBDOM_DOCUMENT          pbdom_doc
PBDOM_ATTRIBUTE         pbdom_attr
PBDOM_ENTITYREFERENCE   pbdom_entref
PBDOM_OBJECT            pbdom_obj_array[]

try
    pbdom_doc = Create PBDOM_DOCUMENT
    pbdom_entref = Create PBDOM_ENTITYREFERENCE
```

```

// Create a new document object.
pbdom_doc.NewDocument ("root")
// Add an attribute "my_attr" to the root element.
pbdom_doc.GetRootElement().SetAttribute("my_attr", &
    "attribute text")
// Set the name of the PBDOM_ENTITYREFERENCE.
pbdom_entref.SetName ("ent_ref")

// Get the existing contents of my_attr
pbdom_doc.GetRootElement().GetAttribute("my_attr").&
    GetContent(pbdom_obj_array)

// Insert the entity reference to the root element's
// my_attr attribute before the attribute text.
pbdom_doc.GetRootElement().GetAttribute("my_attr").&
    InsertContent(pbdom_entref, pbdom_obj_array[1])

// Test the text contents of "my_attr"
if pbdom_doc.GetRootElement(). &
    GetAttribute("my_attr").GetText() = &
    "&ent_ref;attribute text" then
    MessageBox ("Pass", &
        "GetText() on my_attr is correct.")
else
    MessageBox ("Fail", &
        "GetText() on my_attr is incorrect.")
end if

catch (pbdom_exception pbdom_except)
    MessageBox ("PBDOM_EXCEPTION", &
        pbdom_except.GetMessage())
end try

```

**Usage**

This method inserts the input PBDOM\_OBJECT as a child at a specific position (before the reference PBDOM\_OBJECT). Currently, only a PBDOM\_TEXT and a PBDOM\_ENTITYREFERENCE object can be inserted as a child of a PBDOM\_ATTRIBUTE.

If the reference PBDOM\_OBJECT is null, the PBDOM\_OBJECT to be inserted is inserted at the end of this PBDOM\_ATTRIBUTE object's list of children.

**See also**

AddContent  
 GetContent  
 RemoveContent  
 SetContent

## IsAncestorObjectOf

Description	Determines whether the current PBDOM_ATTRIBUTE object is the ancestor of another PBDOM_OBJECT.						
Syntax	<i>pbdom_attribute_name</i> .IsAncestorObjectOf( <i>pbdom_object</i> <i>pbdom_object_ref</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_document_name</i></td> <td>The name of a PBDOM_ATTRIBUTE object</td> </tr> <tr> <td><i>pbdom_object_ref</i></td> <td>A reference to a PBDOM_OBJECT to check against</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_document_name</i>	The name of a PBDOM_ATTRIBUTE object	<i>pbdom_object_ref</i>	A reference to a PBDOM_OBJECT to check against
Argument	Description						
<i>pbdom_document_name</i>	The name of a PBDOM_ATTRIBUTE object						
<i>pbdom_object_ref</i>	A reference to a PBDOM_OBJECT to check against						
Return value	Boolean. Returns true if this PBDOM_ATTRIBUTE is the ancestor of the input PBDOM_PBOBJECT and false otherwise.						
Throws	EXCEPTION_INVALID_ARGUMENT– The input PBDOM_OBJECT is invalid. This can happen if it has not been initialized properly or is a null object reference.						
Usage	This method checks to see whether the current PBDOM_ATTRIBUTE is the ancestor object of the input PBDOM_OBJECT. According to the W3C DOM specification, only a PBDOM_TEXT and a PBDOM_ENTITYREFERENCE object can become a child object of a PBDOM_ATTRIBUTE, and therefore a PBDOM_ATTRIBUTE can only be an ancestor of a PBDOM_TEXT or a PBDOM_ENTITYREFERENCE object.						

## RemoveContent

Description	Removes the input PBDOM_OBJECT from the PBDOM_ATTRIBUTE.						
Syntax	<i>pbdom_attribute_name</i> .RemoveContent( <i>pbdom_object</i> <i>pbdom_object_ref</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_attribute_name</i></td> <td>The name of the PBDOM_ATTRIBUTE</td> </tr> <tr> <td><i>pbdom_object_ref</i></td> <td>The PBDOM_OBJECT child to be removed from this PBDOM_ATTRIBUTE</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE	<i>pbdom_object_ref</i>	The PBDOM_OBJECT child to be removed from this PBDOM_ATTRIBUTE
Argument	Description						
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE						
<i>pbdom_object_ref</i>	The PBDOM_OBJECT child to be removed from this PBDOM_ATTRIBUTE						
Return value	Boolean. Returns true if the content has been successfully removed and false otherwise.						

Throws

EXCEPTION\_INVALID\_ARGUMENT – The input PBDOM\_OBJECT is invalid. This can happen if it has not been initialized properly or is a null object reference.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_ATTRIBUTE object or the input PBDOM\_OBJECT is not associated with a derived PBDOM\_OBJECT class object.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – This PBDOM\_ATTRIBUTE or the PBDOM\_OBJECT to be removed is nameable and has not been given a user-defined name.

EXCEPTION\_WRONG\_DOCUMENT\_ERROR – The input PBDOM\_OBJECT is not contained within the same PBDOM\_DOCUMENT as this PBDOM\_ATTRIBUTE.

EXCEPTION\_WRONG\_PARENT\_ERROR – The input PBDOM\_OBJECT is not a child of the current PBDOM\_ATTRIBUTE.

Examples

This example adds an attribute to the root element with the name my\_attr and text content “attribute text”. It then creates a PBDOM\_ENTITYREFERENCE object named ent\_ref and inserts it before the attribute’s current content.

At this point, my\_attr contains two child PBDOM\_OBJECTS: a PBDOM\_TEXT containing “attribute text” and a PBDOM\_ENTITYREFERENCE named ent\_ref. The element looks like this when serialized:

```
<root my_attr="attribute text&ent_ref;">
```

A call to GetContent returns an array containing these two PBDOM\_OBJECTS. pbdom\_obj\_array[1] should point to the PBDOM\_TEXT. After pbdom\_obj\_array[1] is removed from my\_attr, the element looks like this when serialized: <root my\_attr="&ent\_ref;">.

```
PBDOM_DOCUMENT      pbdom_doc
PBDOM_ATTRIBUTE     pbdom_attr
PBDOM_ENTITYREFERENCE pbdom_entref
PBDOM_OBJECT        pbdom_obj_array[]
```

```
try
    pbdom_doc = Create PBDOM_DOCUMENT
    pbdom_entref = Create PBDOM_ENTITYREFERENCE

    // Create a new document object.
    pbdom_doc.NewDocument ("root")
    // Add an attribute "my_attr" to the root element.
    pbdom_doc.GetRootElement().SetAttribute("my_attr", &
```

```

        "attribute text")

    // Set the name of our PBDOM_ENTITYREFERENCE.
    pbdom_entref.SetName ("ent_ref")

    // Add the entity reference to the root
    // element's "my_attr" attribute.
    pbdom_doc.GetRootElement(). &
        GetAttribute("my_attr"). AddContent (pbdom_entref)

    // Get the existing contents of "my_attr"
    pbdom_doc.GetRootElement().GetAttribute("my_attr").&
        GetContent (pbdom_obj_array)

    // Remove PBDOM_TEXT object from "my_attr"
    pbdom_doc.GetRootElement().GetAttribute("my_attr").&
        RemoveContent (pbdom_obj_array[1])

    // Test the text contents of "my_attr "
    if pbdom_doc.GetRootElement(). &
        GetAttribute("my_attr").GetText() = &
            "&ent_ref;" then
        MessageBox ("Pass", &
            "GetText() on my_attr is correct.")
    else
        MessageBox ("Fail",
            "GetText() on my_attr is incorrect.")
    end if

    catch (pbdom_exception pbdom_e)
        MessageBox ("PBDOM_EXCEPTION", pbdom_e.GetMessage())
    end try

```

**Usage** The RemoveContent method removes the input PBDOM\_OBJECT from this PBDOM\_ATTRIBUTE. Currently, only a PBDOM\_TEXT and a PBDOM\_ENTITYREFERENCE object can be part of the contents of a PBDOM\_ATTRIBUTE. Therefore, the input PBDOM\_OBJECT must be either a PBDOM\_TEXT or a PBDOM\_ENTITYREFERENCE object.

**See also**

- AddContent
- GetContent
- InsertContent
- SetContent

## SetBooleanValue

**Description** Sets the text value of a PBDOM\_ATTRIBUTE object. The SetBooleanValue method creates this text value by serializing the provided boolean value into a string.

**Syntax** `pbdom_attribute_name.SetBooleanValue(boolean boolValue)`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>boolValue</i>	A boolean value to be set for the PBDOM_ATTRIBUTE

**Return value** PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE from which the SetBooleanValue method was invoked.

**See also** GetBooleanValue

## SetContent

**Description** Sets the content of this PBDOM\_ATTRIBUTE.

**Syntax** `pbdom_attribute_name.SetContent(pbdom_object pbdom_object_array)`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>pbdom_object_array</i>	An array of PBDOM_OBJECTs

**Return value** PBDOM\_OBJECT. This PBDOM\_ATTRIBUTE modified.

**Throws** EXCEPTION\_ILLEGAL\_PBOBJECT – One of the array items is not a valid PBDOM object. This can happen if the array item has not been initialized properly or is a null object reference. This is similar to EXCEPTION\_INVALID\_ARGUMENT.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – One of the array items is nameable and has not been given a user-defined name.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – One of the array items is not associated with a derived PBDOM\_OBJECT.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – One of the array items already has a parent.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – One of the array items is not allowed to be set as part of the contents of a PBDOM\_ATTRIBUTE.

## Examples

This example demonstrates setting the contents of a PBDOM\_ATTRIBUTE object. It creates a PBDOM\_DOCUMENT with root element root and attaches to it a PBDOM\_DOCTYPE with the following internal subset:

```
<!ELEMENT root ANY>
<!ATTLIST root attr CDATA #REQUIRED>
<!ENTITY ent_ref "MY ENTITY REFERENCE">
```

It also creates a PBDOM\_ATTRIBUTE, attr, and sets as its contents an array of three PBDOM\_OBJECTS:

- A PBDOM\_TEXT with the text value “start text”
- A PBDOM\_ENTITYREFERENCE named ent\_ref
- A PBDOM\_TEXT with the text value “end text.”

This removes the original contents of attr and sets new contents so that when the document is serialized into an external file, the root element looks like this:

```
<root attr="start text &ent_ref; end text."/>
```

Finally, a user-defined function called GetAttributeText parses the external serialized XML file and retrieves the text value of the attr attribute.

The code for GetAttributeText function is as follows:

```
PBDOM_BUILDER      pbdom_buildr
PBDOM_DOCUMENT    pbdom_doc
string             strReturn

try
    pbdom_buildr = Create PBDOM_BUILDER
    pbdom_doc = pbdom_buildr. &
        BuildFromFile (strXMLFileName)

    strReturn = pbdom_doc.GetRootElement(). &
        GetAttribute(strAttributeName).GetText()
catch (PBDOM_EXCEPTION pbdom_except)
    strReturn = ""
end try
return strReturn
```

This function builds a PBDOM\_DOCUMENT from the external XML file (its first argument) and gets the text value of an attribute (its second argument) from the root element.

The code that sets the content of the PBDOM\_ATTRIBUTE is as follows:

```
PBDOM_DOCUMENT      pbdom_doc
PBDOM_DOCTYPE       pbdom_doctyp
PBDOM_ATTRIBUTE     pbdom_attr
PBDOM_TEXT          pbdom_txt
PBDOM_OBJECT        pbdom_obj_array_set []
long l = 0

try
    pbdom_doc = Create PBDOM_DOCUMENT
    pbdom_doc.NewDocument ("root")

    pbdom_doctyp = Create PBDOM_DOCTYPE
    pbdom_doctyp.SetName ("root")
    pbdom_doctyp.setinternalsubset("<!ELEMENT root
ANY><!ATTLIST root attr CDATA #REQUIRED><!ENTITY
ent_ref ~"MY ENTITY REFERENCE~">")

    pbdom_doc.SetDocType (pbdom_doctyp)

    pbdom_doc.GetRootElement().SetAttribute("attr", "")

    pbdom_obj_array_set [1] = Create PBDOM_TEXT
    pbdom_txt = pbdom_obj_array_set [1]
    pbdom_txt.SetText ("start text ")

    pbdom_obj_array_set [2] = Create PBDOM_ENTITYREFERENCE
    pbdom_obj_array_set [2].SetName ("ent_ref")

    pbdom_obj_array_set [3] = Create PBDOM_TEXT
    pbdom_txt = pbdom_obj_array_set [3]
    pbdom_txt.SetText (" end text.")

    pbdom_doc.GetRootElement().GetAttribute("attr"). &
        SetContent (pbdom_obj_array_set)

    pbdom_doc.SaveDocument &
        ("c:\xmltests\attr_set_content.xml")

    MessageBox ("Attribute Text", GetAttributeText &
        ("c:\xmltests\attr_set_content.xml", "attr"))

catch (PBDOM_EXCEPTION pbdom_e)
    MessageBox ("PBDOM_EXCEPTION", pbdom_e.GetMessage())
end try
```



Usage

This method sets the content of this PBDOM\_ATTRIBUTE. The supplied array should contain only objects of type PBDOM\_TEXT and PBDOM\_ENTITYREFERENCE.

When all objects in the supplied array are legal and before the new content is added, all objects in the old content will have their parentage set to null (no parent) and the old content list will be cleared from this PBDOM\_ATTRIBUTE.

This has the effect that the items of any active array (previously obtained with a call to GetContent) also change to reflect the new condition. In addition, all objects in the supplied array have their parentage set to this PBDOM\_ATTRIBUTE.

Passing a null value or an empty array clears the existing content of this PBDOM\_ATTRIBUTE.

See also

AddContent, GetContent, RemoveContent, SetContent

## SetDateValue

**Description** Sets the text value of a PBDOM\_ATTRIBUTE object. The SetDateValue method creates this text value by serializing the provided date value into a string.

**Syntax** `pbdom_attribute_name.SetDateValue(date dateValue, strDateFormat)`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>dateValue</i>	A date value to be set for the PBDOM_ATTRIBUTE
<i>strDateFormat</i>	The format in which the date value is to be set for the PBDOM_ATTRIBUTE, for example, MM:DD:YYYY

The value of the *strDateFormat* parameter can include slashes or colons as delimiters. The following table illustrates characters having special meaning in *strDateFormat*.

Character	Meaning	Example
D	Day number with no leading zero	5
DD	Day number with leading zero, if applicable	05
M	Month number with no leading zero	5
MM	Month number with leading zero, if applicable	05
YY	Two-digit year number	05
YYYY	Four-digit year number	2005

**Return value** PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE from which the SetDateValue method was invoked.

**See also** GetDateValue

## SetDateTimeValue

**Description** Sets the text value of a PBDOM\_ATTRIBUTE object and creates this text value by serializing the provided datetime value into a string.

**Syntax** `pbdom_attribute_name.SetDateTimeValue(datetime datetimeValue, string strDateFormat, string strTimeFormat)`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>datetimeValue</i>	A datetime value to be set for the PBDOM_ATTRIBUTE
<i>strDateFormat</i>	The format in which the date part of the datetime value is to be set for the PBDOM_ATTRIBUTE, for example, MM:DD:YYYY
<i>strTimeFormat</i>	The format in which the time part of the datetime value is to be set for the PBDOM_ATTRIBUTE, for example, HH:MM:SS

The value of the *strDateFormat* parameter can use slashes or colons as delimiters. The following table illustrates characters that have special meaning in *strDateFormat*.

Character	Meaning	Example
D	Day number with no leading zero	5
DD	Day number with leading zero, if applicable	05
M	Month number with no leading zero	5
MM	Month number with leading zero, if applicable	05
YY	Two-digit year number	05
YYYY	Four-digit year number	2005

The value of the *strTimeFormat* parameter can include slashes or colons as delimiters. The following table illustrates characters that have special meaning in *strTimeFormat*.

Character	Meaning	Example
H	Hour number with no leading zero	5
HH	Hour number with leading zero, if applicable	05
M	Minutes number with no leading zero	5
MM	Minutes number with leading zero, if applicable	05
S	Seconds number with no leading zero	5
SS	Seconds number with leading zero, if applicable	55

**Return value** PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE from which the SetDateTimeValue method was invoked.

See also                      GetDateTimeValue

## SetDoubleValue

**Description**                      Sets the text value of a PBDOM\_ATTRIBUTE object. The SetDoubleValue method creates this text value by serializing the provided double value into a string.

**Syntax**                              *pbdom\_attribute\_name*.SetDoubleValue( double *doubleValue*)

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>doubleValue</i>	A double value to be set for the PBDOM_ATTRIBUTE

**Return value**                      PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE from which the SetDoubleValue method was invoked.

See also                              GetDoubleValue

## SetIntValue

**Description**                      Sets the text value of a PBDOM\_ATTRIBUTE object. The SetIntValue method creates this text value by serializing the provided int value into a string.

**Syntax**                              *pbdom\_attribute\_name*.SetIntValue(integer *intValue*)

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>intValue</i>	An int value to be set for the PBDOM_ATTRIBUTE

**Return value**                      PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE from which the SetIntValue method was invoked.

See also                              GetIntValue

## SetLongValue

**Description**                      Sets the text value of a PBDOM\_ATTRIBUTE object. The SetLongValue method creates this text value by serializing the provided long value into a string.

**Syntax**                              *pbdom\_attribute\_name*.SetLongValue( long *longValue*)

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>longValue</i>	A long value to be set for the PBDOM_ATTRIBUTE

Return value PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE from which the SetLongValue method was invoked.

See also GetLongValue

## SetName

Description Sets the local name of the PBDOM\_ATTRIBUTE object.

Syntax *pbdom\_attribute\_name*.SetName(string *strName*)

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>strName</i>	The new local name for the PBDOM_ATTRIBUTE

Return value Boolean. Returns true if the local name of the PBDOM\_ATTRIBUTE has been changed and false otherwise.

Throws EXCEPTION\_INVALID\_NAME – If the input name is not valid for a local name of a PBDOM\_ATTRIBUTE. This happens if the name is an empty string, if the name contains a namespace prefix, or if the name is already the name of an existing attribute of the owning element.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

Examples This example shows how to set the local name of a PBDOM\_ATTRIBUTE and demonstrates that the namespace information it contains is not affected by a change in name.

The sample code first builds a PBDOM\_DOCUMENT from a string that contains XML that has a single root element with a namespace declaration and an attribute *a*.

The GetAttribute method obtains the attribute *a*, which does not belong to a namespace, and the returned PBDOM\_ATTRIBUTE is tested and should be valid. After a call to SetName, the code confirms the name change and tests that the namespace information remains the same (the namespace prefix and URI are both still empty strings):

```
PBDOM_BUILDER    pbdom_buildr
```

```
PBDOM_DOCUMENT    pbdom_doc
PBDOM_ATTRIBUTE   pbdom_attr
string strXML = "<root xmlns:n1=~\"http://www.n.com~\"
a=~\"123~/>"

try
  pbdom_buildr = Create PBDOM_BUILDER
  pbdom_doc = pbdom_buildr.BuildFromString (strXML)

  pbdom_attr = pbdom_doc.GetRootElement(). &
    GetAttribute("a")

  if (IsValid(pbdom_attr)) then
    MessageBox ("Pass", &
      "PBDOM_ATTRIBUTE a is retrieved via the " &
      + "NONAMESPACE GetAttribute() method.")
  else
    MessageBox ("Fail", &
      "PBDOM_ATTRIBUTE should have been retrievable.")
  end if

  pbdom_attr.SetName ("b")

  if pbdom_attr.GetName() = "b" then
    MessageBox ("Pass", "Name has been changed to b.")
  else
    MessageBox ("Fail", &
      "Name should have been changed to b.")
  end if

  if pbdom_attr.GetNamespacePrefix() = "" then
    MessageBox ("Pass", &
      "Namespace Prefix is an empty string.")
  else
    MessageBox ("Fail", "Namespace Prefix is : " &
      + pbdom_attr.GetNamespacePrefix() &
      + " which is incorrect.")
  end if

  if pbdom_attr.GetNamespaceURI() = "" then
    MessageBox ("Pass", &
      "Namespace URI is an empty string.")
  else
    MessageBox ("Fail", "Namespace URI is : " &
      + pbdom_attr.GetNamespaceURI() &
      + " which is incorrect.")
  end if
end try
```

```

        end if

        catch (PBDOM_EXCEPTION pbdom_e)
            MessageBox("PBDOM_EXCEPTION", pbdom_e.GetMessage())
        end try

```

**Usage**

This method sets the local name of the PBDOM\_ATTRIBUTE. When a PBDOM\_ATTRIBUTE is first created, it has no name and the namespace information is by default set to the NONNAMESPACE namespace. (Its NS Prefix and URI are both empty strings.)

The SetName method is used to set the local name of the PBDOM\_ATTRIBUTE. The SetNamespace method is used to set the Namespace Prefix and URI of the PBDOM\_ATTRIBUTE.

If a PBDOM\_ATTRIBUTE is retrieved programmatically from a parsed document, then the name and namespace information of the PBDOM\_ATTRIBUTE are inherited from the referred attribute of the parsed document. The name and namespace information of the PBDOM\_ATTRIBUTE, however, can still be modified using the SetName and SetNamespace methods.

Note that according to the W3C “Namespaces in XML” specification, when the SetName method is invoked on a PBDOM\_ATTRIBUTE, if the PBDOM\_ATTRIBUTE (PBDOM\_ATTRIBUTE 1) has an owner PBDOM\_ELEMENT that contains an existing PBDOM\_ATTRIBUTE (PBDOM\_ATTRIBUTE 2) with the same name (to be set for PBDOM\_ATTRIBUTE 1) and namespace URI as PBDOM\_ATTRIBUTE 1, the EXCEPTION\_INVALID\_NAME exception will be thrown.

**See also**

GetName  
SetOwnerElementObject

## SetNamespace

**Description** Sets the namespace for a PBDOM\_ATTRIBUTE object based on the specified namespace prefix and URI.

**Syntax** *pbdom\_attribute\_name*.SetNamespace(string*strNamespacePrefix*, string*strNamespaceUri*, boolean *bVerifyNamespace*)

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>strNamespacePrefix</i>	A string containing the namespace prefix to be set for the PBDOM_ATTRIBUTE
<i>strNamespaceUri</i>	A string containing the namespace URI to be set for the PBDOM_ATTRIBUTE
<i>bVerifyNamespace</i>	A boolean value to indicate whether to search for an in-scope namespace declaration that matches the input namespace prefix and URI

**Return value** Long. Returns 0 if namespace information was set successfully and -1 if no in-scope namespace matching the input prefix and URI exists.

**Throws** EXCEPTION\_INVALID\_NAME – If the input namespace prefix or the URI or the combination of prefix and URI is not valid. This occurs if:

- The namespace prefix is an empty string and the URI is not an empty string. If both are empty strings, the NONNAMESPACE namespace is being specified and this prefix/URI combination is correct.
- The namespace Prefix is `xmlns` and the URI is not `http://www.w3.org/2000/xmlns/`. This namespace prefix/URI pair is unique and exclusive. Its elements cannot be used individually and separately. The use of this pair signifies a namespace declaration.
- The namespace prefix string is invalid. That is, it does not conform to the W3C “Namespaces in XML” specifications for the name of a prefix.
- The namespace URI string is invalid. That is, it does not conform to the W3C specifications for a URI string.
- The owner Element of this PBDOM\_ATTRIBUTE already contains an attribute that has the same name as the current PBDOM\_ATTRIBUTE and belongs to the namespace that is to be set for the current PBDOM\_ATTRIBUTE.

EXCEPTION\_INVALID\_ARGUMENT – If the input namespace prefix string or the URI string has been set to null.



EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – If there is insufficient memory to allocate for internal strings.

EXCEPTION\_INTERNAL\_XML\_ENGINE\_ERROR – If some internal error occurred in the XML engine.

## Examples

This example demonstrates how to set the namespace prefix and URI for a PBDOM\_ATTRIBUTE. It creates a PBDOM\_DOCUMENT based on the following XML document:

```
<root xmlns:pre1="http://www.pre.com">
  <child1 pre1:a="123" b="456"/>
</root>
```

The namespace *http://www.pre.com*, which has the prefix *pre1*, is defined in the root element. The child element *child1* has an attribute *a* that belongs to the declared namespace and an attribute *b* that does not belong to a namespace.

The example uses `GetAttribute` to get and store the attribute *b* in `pbdom_attr`, then calls `SetNamespace` on `pbdom_attr`, specifying the strings “*pre1*” and “*http://www.pre.com*” as the prefix and URI, and setting the *bVerifyNamespace* parameter to `true`. This tells `SetNamespace` to check first to see if the owner element of *b* or the owner element's ancestor elements contain a namespace declaration for the *pre1/http://www.pre.com* namespace prefix/URI pair.

The search for this prefix/URI pair succeeds because the root element contains such a namespace declaration.

```
PBDOM_BUILDER    pbdom_buildr
PBDOM_DOCUMENT  pbdom_doc
PBDOM_ATTRIBUTE  pbdom_attr
string strXML = "<root
xmlns:pre1=~\"http://www.pre.com~\"><child1
pre1:a=~\"123~\" b=~\"456~\"/></root>"

try
  pbdom_buildr = Create PBDOM_BUILDER
  pbdom_doc = pbdom_buildr.BuildFromString (strXML)

  pbdom_attr =
  pbdom_doc.GetRootElement().GetChildElement("child1").GetAttribute("b", "", "")

  pbdom_attr.SetNamespace("pre1",
"http://www.pre.com", true)

  MessageBox ("NS Prefix",
```

```
pbdom_attr.GetNamespacePrefix()  
    MessageBox ("NS URI", pbdom_attr.GetNamespaceURI())  
    MessageBox ("Name", pbdom_attr.getName())  
    MessageBox ("Text", pbdom_attr.getText())  
  
    pbdom_doc.SaveDocument ("ns.xml")  
  
catch (PBDOM_EXCEPTION pbdom_except)  
    MessageBox ("PBDOM_EXCEPTION",  
pbdom_except.GetMessage())  
end try
```

There is no other attribute inside `child1` that has the name `b` and that also belongs to the `http://www.pre.com namespace`, so the `SetNamespace` method succeeds. When serialized, the `PBDOM_DOCUMENT` looks like this:

```
<root xmlns:pre1="http://www.pre.com">  
    <child1 pre1:b="456" pre1:a="123" />  
</root>
```

### Usage

This method sets this `PBDOM_ATTRIBUTE` object's namespace based on the input prefix and URI. The input prefix can be an empty string, but the input URI cannot be an empty string unless the prefix is also an empty string.

If the input prefix and URI are both empty strings, the `PBDOM_ATTRIBUTE` has no namespace. The `bVerifyNamespace` parameter tells the method whether to search for an in-scope namespace declaration that matches the input namespace prefix and URI.

As required by the W3C specification on “Namespaces in XML,” if the current `PBDOM_ATTRIBUTE` has an owner `PBDOM_ELEMENT` that contains an existing `PBDOM_ATTRIBUTE` that has the same name as the current `PBDOM_ATTRIBUTE` and the same namespace URI as is to be set for the current `PBDOM_ATTRIBUTE`, the `EXCEPTION_INVALID_NAME` exception is thrown.

### See also

- `GetName`
- `GetNamespacePrefix`
- `GetNamespaceUri`
- `GetQualifiedName`
- `SetName`

## SetOwnerElementObject

Description	Sets the input PBDOM_ELEMENT as the owner of the current PBDOM_ATTRIBUTE.						
Syntax	<i>pbdom_attribute_name</i> .SetOwnerElementObject( <i>pbdom_element pbdom_element_ref</i> )						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Argument</th> <th style="text-align: left;">Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_attribute_name</i></td> <td>The name of the PBDOM_ATTRIBUTE</td> </tr> <tr> <td><i>pbdom_element_ref</i></td> <td>The PBDOM_ELEMENT to be set as the owner of this current PBDOM_ATTRIBUTE</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE	<i>pbdom_element_ref</i>	The PBDOM_ELEMENT to be set as the owner of this current PBDOM_ATTRIBUTE
Argument	Description						
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE						
<i>pbdom_element_ref</i>	The PBDOM_ELEMENT to be set as the owner of this current PBDOM_ATTRIBUTE						
Return value	PBDOM_ATTRIBUTE. This PBDOM_ATTRIBUTE itself modified and returned.						
Throws	<p>EXCEPTION_INVALID_ARGUMENT – The input PBDOM_ELEMENT is invalid. This can happen if it has not been initialized properly or is a null object reference.</p> <p>EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – The internal implementation of the PBDOM_ATTRIBUTE object or the input PBDOM_ELEMENT object is null. The occurrence of this exception is rare but can take place if severe memory corruption occurs.</p> <p>EXCEPTION_PBDOM_OBJECT_ALREADY_HAS_OWNER – This PBDOM_ATTRIBUTE already has an owner Element.</p> <p>EXCEPTION_USE_OF_UNNAMED_PBDOM_OBJECT – The input PBDOM_ELEMENT has not been named.</p> <p>EXCEPTION_INVALID_NAME – The input PBDOM_ELEMENT already contains an attribute that has the same name and that belongs to the same namespace as this current PBDOM_ATTRIBUTE.</p>						
Examples	<p>This example moves the positions of two PBDOM_ATTRIBUTE objects from one element to another.</p> <p>In the string strXML from which a PBDOM_DOCUMENT is created, the abc root element contains a namespace declaration and two attributes. My_Attr belongs to no namespace, and pre:My_Attr_NS belongs to the http://www.pre.com namespace.</p>						

The example obtains handles for the two attributes and the data element, then detaches both attributes from abc and sets data as their new owner:

```
PBDOM_BUILDER      pbdombuilder_new
PBDOM_DOCUMENT     pbdom_doc
PBDOM_ATTRIBUTE    pbdom_attr
PBDOM_ATTRIBUTE    pbdom_attr_ns
PBDOM_ELEMENT      pbdom_elem_data
string strXML = "<abc My_Attr=~\"Attribute Value~\"
pre:My_Attr_NS=~\"Attribute Value NS~\"
xmlns:pre=~\"http://www.pre.com~\"><data>Data</data></ab
c>"
```

TRY

```
pbdombuilder_new = Create PBDOM_Builder
pbdom_doc = pbdombuilder_new.BuildFromString(strXML)

pbdom_attr = pbdom_doc.GetRootElement(). &
    GetAttribute("My_Attr")
pbdom_attr_ns = pbdom_doc.GetRootElement(). &
    GetAttribute("My_Attr_NS", "pre", &
    "http://www.pre.com")
pbdom_elem_data = pbdom_doc.GetRootElement(). &
    GetChildElement("data")

pbdom_attr.Detach()
pbdom_attr.SetOwnerElementObject (pbdom_elem_data)

pbdom_attr_ns.Detach()
pbdom_attr_ns.SetOwnerElementObject (pbdom_elem_data)

pbdom_doc.SaveDocument("setownerelementobject.xml")

Destroy pbdombuilder_new
Destroy pbdom_doc
```

```
CATCH (PBDOM_Exception except)
    MessageBox ("Exception Occurred", except.Text)
END TRY
```

When the document is serialized, the XML looks like this:

```
<abc xmlns:pre="http://www.pre.com">
<data pre:My_Attr_NS="Attribute Value NS"
My_Attr="Attribute Value">Data</data>
</abc>
```

Usage	According to the “Namespace in XML” specifications, an element cannot contain two attributes with the same local name and namespace URI. This is true even if the prefixes of the two attributes are different. An exception is thrown if this rule is violated when <code>SetOwnerElementObject</code> is invoked.
See also	<code>GetOwnerElementObject</code>

## SetRealValue

Description	Sets the text value of a <code>PBDOM_ATTRIBUTE</code> object. The <code>SetRealValue</code> method creates this text value by serializing the provided real value into a string.						
Syntax	<code><i>pbdom_attribute_name</i>.SetRealValue(real <i>realValue</i>)</code>						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><i>pbdom_attribute_name</i></code></td> <td>The name of the <code>PBDOM_ATTRIBUTE</code></td> </tr> <tr> <td><code><i>realValue</i></code></td> <td>A real value to be set for the <code>PBDOM_ATTRIBUTE</code></td> </tr> </tbody> </table>	Argument	Description	<code><i>pbdom_attribute_name</i></code>	The name of the <code>PBDOM_ATTRIBUTE</code>	<code><i>realValue</i></code>	A real value to be set for the <code>PBDOM_ATTRIBUTE</code>
Argument	Description						
<code><i>pbdom_attribute_name</i></code>	The name of the <code>PBDOM_ATTRIBUTE</code>						
<code><i>realValue</i></code>	A real value to be set for the <code>PBDOM_ATTRIBUTE</code>						
Return value	<code>PBDOM_ATTRIBUTE</code> . The <code>PBDOM_ATTRIBUTE</code> from which the <code>SetRealValue</code> method was invoked.						
See also	<code>GetRealValue</code>						

## SetText

Description	Sets the string value of a <code>PBDOM_ATTRIBUTE</code> object.						
Syntax	<code><i>pbdom_attribute_name</i>.SetText(string <i>strText</i>)</code>						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code><i>pbdom_attribute_name</i></code></td> <td>The name of the <code>PBDOM_ATTRIBUTE</code></td> </tr> <tr> <td><code><i>strText</i></code></td> <td>The string value to be set in the <code>PBDOM_ATTRIBUTE</code></td> </tr> </tbody> </table>	Argument	Description	<code><i>pbdom_attribute_name</i></code>	The name of the <code>PBDOM_ATTRIBUTE</code>	<code><i>strText</i></code>	The string value to be set in the <code>PBDOM_ATTRIBUTE</code>
Argument	Description						
<code><i>pbdom_attribute_name</i></code>	The name of the <code>PBDOM_ATTRIBUTE</code>						
<code><i>strText</i></code>	The string value to be set in the <code>PBDOM_ATTRIBUTE</code>						
Return value	<code>PBDOM_ATTRIBUTE</code> .						
Usage	This method returns the current <code>PBDOM_ATTRIBUTE</code> with the input string value set.  This method is the counterpart of the JDOM <code>setValue</code> method.						
See also	<code>GetText</code> <code>GetTextNormalize</code> <code>GetTextTrim</code>						

## SetTimeValue

**Description** Sets the text value of a PBDOM\_ATTRIBUTE object. The SetTimeValue method creates this text value by serializing the provided time value into a string.

**Syntax** *pbdom\_attribute\_name*.SetTimeValue(time *timeValue*, string *strTimeFormat*)

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>timeValue</i>	A time value to be set for the PBDOM_ATTRIBUTE
<i>strTimeFormat</i>	The format in which the time value is to be set for the PBDOM_ATTRIBUTE, for example, HH:MM:SS

The value of the *strTimeFormat* parameter can use slashes or colons as delimiters. The following table illustrates characters that have special meaning in *strTimeFormat*.

Character	Meaning	Example
H	Hour number with no leading zero	5
HH	Hour number with leading zero, if applicable	05
M	Minutes number with no leading zero	5
MM	Minutes number with leading zero, if applicable	05
S	Seconds number with no leading zero	5
SS	Seconds number with leading zero, if applicable	55

**Return value** PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE from which the SetTimeValue method was invoked.

**See also** GetTimeValue

## SetUIntValue

**Description** Sets the text value of a PBDOM\_ATTRIBUTE object. The SetUIntValue method creates this text value by serializing the provided uint value into a string.

**Syntax** `pbdom_attribute_name.SetUIntValue(unsignedinteger uintValue)`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>uintValue</i>	A uint value to be set for the PBDOM_ATTRIBUTE

**Return value** PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE from which the SetUIntValue method was invoked.

**See also** GetUIntValue

## SetUlongValue

**Description** Sets the text value of a PBDOM\_ATTRIBUTE object. The SetUlongValue method creates this text value by serializing the provided ulong value into a string.

**Syntax** `pbdom_attribute_name.SetUlongValue(unsignedlong ulongValue)`

Argument	Description
<i>pbdom_attribute_name</i>	The name of the PBDOM_ATTRIBUTE
<i>ulongValue</i>	A ulong value to be set for the PBDOM_ATTRIBUTE

**Return value** PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE from which the SetUlongValue method was invoked.

**See also** GetUlongValue





About this chapter

This chapter describes the PBDOM\_BUILDER class.

## **PBDOM\_BUILDER**

Description

The PBDOM\_BUILDER class serves as a DOM factory that creates a PBDOM\_DOCUMENT from various input sources, such as a string and a DataStore. A PBDOM\_BUILDER class is not a PBDOM\_OBJECT. There are no DOM objects to which you can map a PBDOM\_BUILDER class.

The PBDOM\_BUILDER methods can be contrasted with the PBDOM\_DOCUMENT NewDocument methods (overloaded with several versions) that are intended to be used to build a PBDOM\_DOCUMENT from scratch.

Methods

PBDOM\_BUILDER has the following methods:

- BuildFromDataStore
- BuildFromFile
- BuildFromString
- GetParseErrors

## BuildFromDataStore

Description Builds a PBDOM\_DOCUMENT from the referenced DataStore object.

Syntax `pbdom_builder_name.BuildFromDataStore(datastore datastore_ref)`

Argument	Description
<code>pbdom_builder_name</code>	The name of a PBDOM_BUILDER object
<code>datastore_ref</code>	A DataStore object

Return value PBDOM\_DOCUMENT.

Throws EXCEPTION\_INVALID\_ARGUMENT – The input DataStore object is invalid. This can happen if it has not been initialized properly or is a null object reference.

Examples The following PowerScript code fragment demonstrates how to use the BuildFromDataStore method with a referenced DataStore object.

```
PBDOM_Builder pbdom_bldr
pbdom_document pbdom_doc
datastore ds

ds = Create datastore
ds.DataObject = "d_customer"
ds.SetTransObject (SQLCA)
ds.Retrieve ()

pbdom_doc = pbdom_bldr.BuildFromDataStore (ds)
```

In this example, a DataStore object *ds* is created and populated with data, and then passed to the BuildFromDataStore method. The BuildFromDataStore method causes the DataStore to export the data to XML, using the most current XML template for the DataStore, and then it uses the XML to build a PBDOM\_DOCUMENT. The PBDOM\_DOCUMENT object is assigned to *pbdom\_doc*.

Usage This method creates a temporary file in the directory pointed to by the user's TMP environment variable. If this directory is invalid, the temporary file is created in the *Windows\temp* directory.

The encoding specified in the XML export template has no effect on the encoding of the document created using BuildFromDataStore. It always has UTF-16LE encoding.

See also BuildFromFile  
BuildFromString

## BuildFromFile

**Description** Builds a PBDOM\_DOCUMENT from the file pointed to by the input URL string. The URL can be a local file path.

**Syntax** *pbdom\_builder\_name*.BuildFromFile (string *strURL*)

Argument	Description
<i>pbdom_builder_name</i>	The name of a PBDOM_BUILDER object
<i>strURL</i>	A string that indicates the URL of the file from which to build a PBDOM_DOCUMENT

**Return value** PBDOM\_DOCUMENT.

**Throws** EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – If there is insufficient memory to create a PBDOM\_DOCUMENT object.

**Examples** Suppose the file *c:\pbdom\_doc\_1.xml* contains the following XML string:

```
<!DOCTYPE abc [<!ENTITY text "Some Text" >]>
<abc>
  <data>
    <child_data>Child Data Text</child_data>
    <child_data An_Attribute="Some Attribute Value"/>
    &text;
    <!--Comment String-->
    <![CDATA[Some CDATA String]]>
  </data>
</abc>
```

The file contains a Document Type Declaration that indicates that `<abc>` is the root element, and a declaration for the text entity that expands to "Some Text":

The root element `abc` contains a child element `data`, which contains five child PBDOM\_OBJECTs: two PBDOM\_ELEMENT objects, and PBDOM\_TEXT, PBDOM\_COMMENT, and PBDOM\_CDATA objects.

The first `child_data` element contains a PBDOM\_TEXT with the string "Child Data Text". The second `child_data` element contains no child PBDOM\_OBJECTs but it does contain a PBDOM\_ATTRIBUTE, `An_Attribute`, that contains the value "Some Attribute Value".

This example creates a PBDOM\_DOCUMENT called pbdom\_doc from *c:\pbdom\_doc\_1.xml*, tests the content of pbdom\_doc, then saves the DOM tree contained within pbdom\_doc into a separate file, *c:\pbdom\_doc\_2.xml*. The input and output files should be identical.

```
PBDOM_Builder      pbdom_bldr
PBDOM_Document   pbdom_doc
PBDOM_Object      pbdom_obj_array[]
PBDOM_Element     pbdom_elem
integer iFileNum1
long l = 0

// Create a PBDOM_DOCUMENT from the XML file
pbdom_bldr = Create PBDOM_Builder
pbdom_doc = pbdom_bldr.BuildFromFile &
            ("c:\pbdom_doc_1.xml")

// Test the contents of the PBDOM_DOCUMENT
// First test the PBDOM_DOCTYPE in the document
MessageBox ("PBDOM_DOCTYPE GetName()", &
            pbdom_doc.GetDocType().GetName())
MessageBox ("PBDOM_DOCTYPE GetInternalSubset()", &
            pbdom_doc.GetDocType().GetInternalSubset())

// Test the root element
MessageBox ("PBDOM_DOC Root Element Name", &
            pbdom_doc.GetRootElement().GetName())

// test the root element's child element
MessageBox ("PBDOM_DOC <data> Element Name", &
            pbdom_doc.GetRootElement().GetChildElement &
            ("data").GetName())

// Collect all the child PBDOM_OBJECTs of the
// <data> element
pbdom_doc.GetRootElement().GetChildElement &
            ("data").GetContent(pbdom_obj_array)

// Display the class name, the name and the text
// contained
// within each PBDOM_OBJECT array item
for l = 1 to UpperBound(pbdom_obj_array)
    MessageBox ("Child Object " + string(l) + " Class",&
                pbdom_obj_array[l].GetObjectClassString())
    MessageBox ("Child Object " + string(l) + " Name",&
                pbdom_obj_array[l].GetName())
```

```

        MessageBox ("Child Object " + string(l) + " Text",&
            pbdom_obj_array[l].GetText())
    next

    // Retrieve and display the name and text value of the
    // "An_Attribute" attribute from the <child_data>
    element
        pbdom_elem = pbdom_obj_array[2]
        MessageBox ("child_data Attribute name", &
            pbdom_elem.GetAttribute("An_Attribute").GetName())
        MessageBox ("child_data Attribute value", &
            pbdom_elem.GetAttribute("An_Attribute").GetText())

    // save the DOM Tree contained within pbdom_doc into
    // a separate file "c:\pbdom_doc_2.xml"
    pbdom_doc.SaveDocument ("c:\pbdom_doc_2.xml")

    Destroy pbdom_bldr

    CATCH (PBDOM_Exception except)
        MessageBox ("Exception Occurred", except.Text)
    END TRY

```

**Usage**

The input URL string can be a local file path.

The encoding specified in the XML export template determines the encoding of the document created using BuildFromFile.

**See also**

BuildFromDataStore  
BuildFromString

## BuildFromString

Description Builds a PBDOM\_DOCUMENT from a string.

Syntax *pbdom\_builder\_name*.BuildFromString(string *strXMLStream*)

Argument	Description
<i>pbdom_builder_name</i>	The name of a PBDOM_BUILDER object
<i>strXMLStream</i>	A string containing XML

Return value PBDOM\_DOCUMENT.

Throws EXCEPTION\_INVALID\_ARGUMENT – The input string is invalid. This can happen if it has not been initialized properly or is a null object reference.  
EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

Examples The following PowerScript code fragment demonstrates how to use the BuildFromString method with an input string. A string containing XML is passed to the BuildFromString method and the return value is assigned to a PBDOM\_DOCUMENT.

```
PBDOM_Builder pbdom_bldr
pbdom_document pbdom_doc
string strXML

strXML = "<Music:abc xmlns:ZMusic="
strXML += "~"http://www.ZMusic.com~">"
strXML += "Root Element Data<data>ABC Data"
strXML += "<inner_data>My Inner Data</inner_data>"
strXML += "My Data</data></abc>"

pbdom_bldr = Create PBDOM_Builder
pbdom_doc = pbdom_bldr.BuildFromString (strXML)
```

Usage The encoding specified in the XML export template determines the encoding of the document created using BuildFromString.

See also BuildFromDataStore  
BuildFromFile

## GetParseErrors

**Description** Obtains a list of parsing errors detected during document parsing.

**Syntax** `pbdom_builder_name.GetParseErrors(ref string strErrorMessageArray[])`

Argument	Description
<code>pbdom_builder_name</code>	The name of a PBDOM_BUILDER object
<code>strErrorMessageArray</code>	An unbounded array of strings, each of which will be filled with a formatted string containing a parse error.

**Return value** Boolean. Returns true if a list of parse errors has been retrieved and false otherwise. Also returns false if there are no parse errors.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – The input string array is invalid. This can happen if it has not been initialised properly or is a null object reference.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

**Examples** The code in this example attempts to create a PBDOM\_DOCUMENT based on the following XML:

```
<!DOCTYPE root
[
<!ELEMENT root ANY>
<!ELEMENT data (#PCDATA)>
<!ENTITY text "Some Text">
]
>
<root><abc/><def/></root>
```

This XML is well formed but is not valid, because the element root contains two child elements abc and def that are not declared in the DOCTYPE. When GetParseErrors is called, it returns the value true, indicating that at least one parse error has occurred, and generates the following list of errors:

```
"1,103,Unknown element 'abc'"
"1,109,Unknown element 'def'"
```

The 1 in both error messages indicates that the error occurred in line 1 of the XML string, and the 103 and 109 indicate columns 103 and 109, respectively.

```
PBDOM_BUILDER pbdom_buildr
PBDOM_DOCUMENT pbdom_doc
long l = 0
string strXML = "<!DOCTYPE root [<!ELEMENT root
ANY><!ELEMENT data (#PCDATA)> <!ENTITY text ~"Some
Text~">]> <root><abc/><def/></root>"
string strParseErrors[]
BOOLEAN bRetTemp = FALSE

try
    pbdom_buildr = Create PBDOM_BUILDER
    pbdom_doc = pbdom_buildr.BuildFromString (strXML)
    bRetTemp = &
        pbdom_buildr.GetParseErrors(strParseErrors)

    if bRetTemp = true then
        for l = 1 to UpperBound(strParseErrors)
            MessageBox ("Parse Error", strParseErrors[l])
        next
    end if
catch (PBDOM_EXCEPTION pbdom_except)
    MessageBox ("PBDOM_EXCEPTION", &
        pbdom_except.GetMessage())
end try
```

#### Usage

This method retrieves a list of errors detected during the last parse operation performed by this PBDOM\_BUILDER. Each string in the array has the following format:

*[Line Number],[Column Number],[Error Message]*

where *Line Number* and *Column Number* indicate the line number and column number in the XML document where the error was encountered. *Error Message* is the parse error message.



About this chapter

This chapter describes the PBDOM\_CDATA class.

## PBDOM\_CDATA

Description

The PBDOM\_CDATA class represents an XML DOM CDATA section. The PBDOM\_CDATA class is derived from PBDOM\_TEXT, which inherits from the PBDOM\_CHARACTERDATA class.

A PBDOM\_CDATA object is used to hold text that contains characters that are prohibited in text objects, such as “<” and “&”, without using entity references. For example, consider the following PBDOM\_CDATA object:

```
<some_text>
  <![CDATA[ (x < y) & (y < z) => x < z ]]>
</some_text>
```

A PBDOM\_TEXT object with the same text content must be written like this:

```
<some_text>
  (x &lt; y) &amp; (y &lt; z) =&gt; x &lt; z
</some_text>
```

However, although the PBDOM\_CDATA class is derived from PBDOM\_TEXT, a PBDOM\_CDATA object cannot always be inserted in the same context as a PBDOM\_TEXT. For example, a PBDOM\_TEXT object can be added as a child of a PBDOM\_ATTRIBUTE, but a PBDOM\_CDATA object cannot.

Methods

Some of the inherited methods from PBDOM\_OBJECT serve no meaningful objective, and only default or trivial functionalities result. These are described in the following table:

<b>Method</b>	<b>Always returns</b>
AddContent	current PBDOM_CDATA
GetContent	false
GetName	a string "#cdata"
HasChildren	false
InsertContent	current PBDOM_CDATA
IsAncestorObjectOf	false
RemoveContent	false
SetContent	current PBDOM_CDATA
SetName	false

PBDOM\_CDATA has the following non-trivial methods:

- Append
- Clone
- Detach
- Equals
- GetObjectClass
- GetObjectClassString
- GetOwnerDocumentObject
- GetParentObject
- GetText
- GetTextNormalize
- GetTextTrim
- SetParentObject
- SetText

## Append

**Description** Appends the input string or the input text data of the PBDOM\_CHARACTERDATA object to the text content that already exists within the current PBDOM\_CDATA object.

**Syntax**

```
pbdom_cdata_name.Append(string strAppend)
pbdom_cdata_name.Append(pbdom_characterdata
pbdom_characterdata_ref)
```

Argument	Description
<i>pbdom_cdata_name</i>	The name of a PBDOM_CDATA
<i>strAppend</i>	The string you want appended to the existing text of the current PBDOM_CDATA object
<i>pbdom_characterdata_ref</i>	The referenced PBDOM_CHARACTERDATA object whose text data is to be appended to the existing text of the current PBDOM_CDATA object

**Return value** PBDOM\_CHARACTERDATA.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA (applies to second syntax).

## Clone

**Description** Creates and returns a clone of the current PBDOM\_CDATA.

**Syntax**

```
pbdom_cdata_name.Clone(boolean bDeep)
```

Argument	Description
<i>pbdom_cdata_name</i>	The name of a PBDOM_CDATA.
<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are <code>true</code> for a deep clone and <code>false</code> for a shallow clone. This argument is currently ignored.

**Return value** PBDOM\_OBJECT. The return value is a clone of the current PBDOM\_CDATA housed in a PBDOM\_OBJECT.

## Examples

This example tests the following characteristics of a cloned PBDOM\_CDATA object:

- The contents of an original and cloned PBDOM\_CDATA object are exactly the same
- A cloned PBDOM\_CDATA initially has no parent object
- A cloned PBDOM\_CDATA is initially contained within the same owner document as the original

```
PBDOM_BUILDER      pbdom_buildr
PBDOM_DOCUMENT    pbdom_doc
PBDOM_CDATA       pbdom_cdat
PBDOM_OBJECT      pbdom_obj_array[]
string strXML = "<![DOCTYPE root [<![ELEMENT root
(#PCDATA)]>]><root><![CDATA[This is a CDATA Section.]]></root>"

try
  // Build a PBDOM_DOCUMENT based on strXML.
  pbdom_buildr = Create PBDOM_BUILDER
  pbdom_doc = pbdom_buildr.BuildFromString (strXML)

  // Get the contents of the root element.
  pbdom_doc.GetRootElement().GetContent(pbdom_obj_array)

  // Test if the root element contains only one child object.
  if (UpperBound(pbdom_obj_array) = 1) then
    MessageBox ("Pass", "Root Element has only one child.")
  else
    MessageBox ("Fail", "Root Element must have only one child.")
  end if

  // Make a clone of the only child of the root element.
  pbdom_cdat = pbdom_obj_array[1].Clone(true)

  // Test if the clone is a PBDOM_CDATA object.
  if (pbdom_cdat.GetObjectClassString() = "pbdom_cdata") then
    MessageBox ("Pass", &
      "The first child, after being cloned, is indeed a PBDOM_CDATA object.")
  else
    MessageBox ("Fail", "The first child, after being cloned, " &
      + "is found to be a " + pbdom_cdat.GetObjectClassString() + " object.")
  end if

  // Test if the clone is a CDATA section.
  if (pbdom_cdat.GetText() = "This is a CDATA Section.") then
    MessageBox ("Pass", "The text contents of the clone is correct.")
```

```

else
    MessageBox ("Fail", "The text contents of the clone is : [" &
        + pbdom_cdat.GetText() + "]. This is incorrect.")
end if

// Test that the clone has no parent.
if (Not IsValid(pbdom_cdat.GetParentObject())) then
    MessageBox ("Pass", "The clone has no parent.")
else
    MessageBox ("Fail", "The clone should have no parent.")
end if

// Test that the clone's owner document is the same
// as the original's owner document.
if (pbdom_cdat.GetOwnerDocumentObject() = pbdom_doc) then
    MessageBox ("Pass", "The clone's owner document is correct.")
else
    MessageBox ("Fail", "The clone's owner document is incorrect.")
end if

catch (PBDOM_EXCEPTION pbdom_except)
    MessageBox ("PBDOM_EXCEPTION", pbdom_except.GetMessage())
end try

```

**Usage** The Clone method creates a new PBDOM\_CDATA object that is a duplicate of, and a separate object from, the original. The clone of a PBDOM\_CDATA is always identical to its original whether deep or shallow cloning is invoked, because a PBDOM\_CDATA object does not contain any subtree of child PBDOM\_OBJECTs.

A PBDOM\_CDATA clone has no parent. However, the clone resides in the same PBDOM\_DOCUMENT as its original, and if the original PBDOM\_CDATA is standalone, the clone is standalone.

## Detach

**Description** Detaches a PBDOM\_CDATA from its parent PBDOM\_OBJECT.

**Syntax** *pbdom\_cdata\_name*.Detach()

Argument	Description
<i>pbdom_cdata_name</i>	The name of a PBDOM_CDATA

**Return value** PBDOM\_OBJECT. The current PBDOM\_CDATA detached from its parent.

**Usage** If the current PBDOM\_CDATA object has no parent, no modifications occur.

## Equals

**Description** Tests for the equality of the current PBDOM\_CDATA and a referenced PBDOM\_OBJECT.

**Syntax** *pbdom\_cdata\_name*.Equals(*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_cdata_name</i>	The name of a PBDOM_CDATA
<i>pbdom_object_ref</i>	A PBDOM_OBJECT to test for equality with the current PBDOM_CDATA

**Return value** Boolean. Returns true if the current PBDOM\_CDATA object is equivalent to the referenced PBDOM\_OBJECT and false otherwise.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not a reference to an object derived from PBDOM\_OBJECT.

**Usage** True is returned only if the referenced PBDOM\_OBJECT is also a derived PBDOM\_CDATA object and refers to the same DOM object as the current PBDOM\_CDATA. Two separately created PBDOM\_CDATA objects, for example, can contain exactly the same text but not be equal.

## GetObjectClass

**Description** Returns a long integer code that indicates the class of the current PBDOM\_OBJECT.

**Syntax** *pbdom\_object\_name*.GetObjectClass()

Argument	Description
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

**Return value** Long. GetObjectClass returns a long integer code that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_CDATA object, the returned value is 8.

**See also** GetObjectClassString

## GetObjectClassString

**Description** Returns a string form of the class of the PBDOM\_OBJECT.

**Syntax** *pbdom\_object\_name*.GetObjectClassString()

Argument	Description
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

**Return value** String. GetObjectClassString returns a string that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_CDATA object, the returned string is “pbdom\_cdata”.

**See also** GetObjectClass

## GetOwnerDocumentObject

**Description** Returns the owning PBDOM\_DOCUMENT of the current PBDOM\_CDATA.

**Syntax** *pbdom\_cdata\_name*.GetOwnerDocumentObject()

Argument	Description
<i>pbdom_cdata_name</i>	The name of a PBDOM_CDATA

**Return value** PBDOM\_OBJECT.

**Usage** If there is no owning PBDOM\_DOCUMENT, null is returned.

**See also** GetParentObject  
SetParentObject

## GetParentObject

**Description** Returns the parent PBDOM\_OBJECT of the PBDOM\_CDATA. If there is no parent, null is returned.

**Syntax** *pbdom\_cdata\_name*.GetParentObject()

Argument	Description
<i>pbdom_cdata_name</i>	The name of a PBDOM_CDATA

**Return value** PBDOM\_OBJECT.

**See also** GetOwnerDocumentObject  
SetParentObject

## GetText

**Description** Returns the text data that is contained within the current PBDOM\_CDATA object.

**Syntax** `pbdom_cdata_name.GetText()`

<b>Argument</b>	<b>Description</b>
<code>pbdom_cdata_name</code>	The name of a PBDOM_CDATA

**Return value** String. The textual content of the current PBDOM\_CDATA object.

**See also** `GetTextNormalize`  
`GetTextTrim`  
`SetText`

## GetTextNormalize

**Description** Returns the text data that is contained within the current PBDOM\_CDATA object, with all surrounding whitespace characters removed and internal whitespace characters normalized to a single space.

**Syntax** `pbdom_cdata_name.GetTextNormalize()`

<b>Argument</b>	<b>Description</b>
<code>pbdom_cdata_name</code>	The name of a PBDOM_CDATA

**Return value** String.

**Usage** If no textual value exists for the current PBDOM\_OBJECT, or if only whitespace characters exist, an empty string is returned.

**See also** `GetText`  
`GetTextTrim`  
`SetText`



## GetTextTrim

**Description** Returns the textual content of the current PBDOM\_CDATA object with all surrounding whitespace characters removed.

**Syntax** `pbdom_cdata_name.GetTextTrim()`

Argument	Description
<i>pbdom_cdata_name</i>	The name of a PBDOM_CDATA

**Return value** String.

**Usage** If no textual value exists for the current PBDOM\_CDATA, or if only whitespace characters exist, an empty string is returned.

**See also** GetText  
GetTextNormalize  
SetText

## SetParentObject

**Description** Sets the referenced PBDOM\_OBJECT to be the parent of the current PBDOM\_CDATA.

**Syntax** `pbdom_cdata_name.SetParentObject(pbdom_object pbdom_object_ref)`

Argument	Description
<i>pbdom_cdata_name</i>	The name of a PBDOM_CDATA
<i>pbdom_object_ref</i>	A PBDOM_OBJECT to be set as the parent of this PBDOM_CDATA object

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not a reference to an object derived from PBDOM\_OBJECT.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – If the current PBDOM\_CDATA already has a parent.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT is of a class that does not have a legal parent-child relationship with the PBDOM\_CDATA class.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT requires a user-defined name and it has not been named.

**Usage** The PBDOM\_OBJECT that you set to be the parent of the current PBDOM\_CDATA must have a legal parent-child relationship. If it does not, an exception is thrown. Only a PBDOM\_ELEMENT object can be set as the parent of a PBDOM\_CDATA object.

**See also** GetParentObject

## SetText

**Description** Sets the input string to be the text content of the current PBDOM\_CDATA object.

**Syntax** *pbdom\_cdata\_name*.SetText(string *strSet*)

Argument	Description
<i>pbdom_cdata_name</i>	The name of a PBDOM_CDATA
<i>strSet</i>	The string you want set as the text of the PBDOM_CDATA

**Return value** PBDOM\_CHARACTERDATA. This PBDOM\_CDATA modified and returned as a PBDOM\_CHARACTERDATA object.

**See also** GetText  
GetTextNormalize  
GetTextTrim

# PBDOM\_ENTITYREFERENCE Class

About this chapter

This chapter describes the PBDOM\_ENTITYREFERENCE class.

## PBDOM\_ENTITYREFERENCE

Description

The PBDOM\_ENTITYREFERENCE class defines behavior for an XML Entity reference Node. It allows you to insert entity references within element nodes as well as attribute nodes. The PBDOM\_ENTITYREFERENCE class is derived from PBDOM\_OBJECT.

Methods

Some of the inherited methods from PBDOM\_OBJECT currently serve no meaningful objective, and only default or trivial functionalities result. These are described in the following table:

Method	Always returns
AddContent	current PBDOM_ENTITYREFERENCE
GetContent	false
GetText	an empty string
GetTextNormalize	an empty string
GetTextTrim	an empty string
HasChildren	false
InsertContent	current PBDOM_ENTITYREFERENCE
IsAncestorObjectOf	false
RemoveContent	false
SetContent	current PBDOM_ENTITYREFERENCE

PBDOM\_ENTITYREFERENCE has the following non-trivial methods:

Clone	GetName	GetParentObject
Detach	GetObjectClass	SetName
Equals	GetObjectClassString	SetParentObject
	GetOwnerDocumentObject	

## Clone

**Description** Creates and returns a clone of the current PBDOM\_ENTITYREFERENCE object.

**Syntax** `pbdom_entityref_name.Clone(boolean bDeep)`

Argument	Description
<i>pbdom_entityref_name</i>	The name of a PBDOM_ENTITYREFERENCE object.
<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are true for a deep clone and false for a shallow clone. This parameter is currently ignored.

**Return value** PBDOM\_OBJECT. A clone of the current PBDOM\_ENTITYREFERENCE object housed in a PBDOM\_OBJECT.

**Examples** This example creates a PBDOM\_DOCUMENT based on a string that contains an XML document, and creates a PBDOM\_ENTITYREFERENCE object to reference the ENTITY my\_er defined in the DOCTYPE. The DOCTYPE also indicates that the root element must contain zero or more child elements named child, and that each child can contain only parsed character data.

The FOR loop creates ten child elements and inserts a new clone of pbdom\_er into each child element. You must use a clone, because the same object cannot be inserted as a child of more than one parent:

```
PBDOM_BUILDER          pbdom_buildr
PBDOM_DOCUMENT         pbdom_doc
PBDOM_ENTITYREFERENCE  pbdom_er
string strXML = "<!DOCTYPE root [<!ELEMENT root
(child)*><!ELEMENT child (#PCDATA)><!ENTITY my_er ~"MY
ENTITY~">]><root/>"
long l = 0
```

```
TRY
  pbdom_buildr = Create PBDOM_BUILDER
  pbdom_doc = pbdom_buildr.BuildFromString(strXML)
  pbdom_er = Create PBDOM_ENTITYREFERENCE
  pbdom_er.SetName("my_er")

// Create 10 child elements for the root element
for l = 1 to 10
  PBDOM_ELEMENT pbdom_elem_child

  pbdom_elem_child = Create PBDOM_ELEMENT
  pbdom_elem_child.SetName("child")
  // Add a clone of pbdom_er as content
```

```

        pbdom_elem_child.AddContent (pbdom_er.Clone (true))

        pbdom_doc.GetRootElement(). &
            AddContent (pbdom_elem_child)
    next

    pbdom_doc.SaveDocument ("clone_er.xml")
CATCH (PBDOM_EXCEPTION pbdom_e)
    MessageBox ("PBDOM_EXCEPTION", pbdom_e.GetMessage ())
END TRY

```

When the PBDOM\_DOCUMENT object is serialized, it produces the following XML document :

```

<!DOCTYPE root
[
<!ELEMENT root (child)*>
<!ELEMENT child (#PCDATA)*>
<!ENTITY my_er "MY ENTITY">
]
>
<root> <child>MY ENTITY</child>
<child>MY ENTITY</child>
<child>MY ENTITY</child>
<child>MY ENTITY</child>
<child>MY ENTITY</child>
<child>MY ENTITY</child>
<child>MY ENTITY</child>
<child>MY ENTITY</child>
<child>MY ENTITY</child>
<child>MY ENTITY</child>
</root>

```

#### Usage

The Clone method creates a new PBDOM\_ENTITYREFERENCE object which is a duplicate of the original. A PBDOM\_ENTITYREFERENCE object cannot contain any child PBDOM\_OBJECTs, so there is no subtree beneath a PBDOM\_ENTITYREFERENCE object. A shallow clone is therefore structurally no different than a deep clone of a PBDOM\_ENTITYREFERENCE object.

This method allows you to use an entity reference node more than once. You cannot add a PBDOM\_ENTITYREFERENCE object as the child of more than one PBDOM\_OBJECT, but you can clone it and then add the clone as the child of another PBDOM\_OBJECT.

A PBDOM\_ENTITYREFERENCE clone does not have any parent. However, the clone resides in the same PBDOM\_DOCUMENT as its original. If the original PBDOM\_ENTITYREFERENCE object is standalone, the clone is also standalone.

## Detach

**Description** Detaches a PBDOM\_ENTITYREFERENCE object from its parent PBDOM\_OBJECT.

**Syntax** *pbdom\_entityref\_name*.Detach()

Argument	Description
<i>pbdom_entityref_name</i>	The name of a PBDOM_ENTITYREFERENCE object

**Return value** PBDOM\_OBJECT. The current PBDOM\_ENTITYREFERENCE object detached from its parent.

**Usage** If the current PBDOM\_ENTITYREFERENCE object has no parent, no modifications occur.

## Equals

**Description** Tests for the equality of the current PBDOM\_ENTITYREFERENCE object and a referenced PBDOM\_OBJECT.

**Syntax** *pbdom\_entityref\_name*.Equals(*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_entityref_name</i>	The name of a PBDOM_ENTITYREFERENCE object

**Return value** Boolean. Returns true if the current PBDOM\_ENTITYREFERENCE object is equivalent to the input PBDOM\_OBJECT, and false otherwise.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not an object derived from PBDOM\_OBJECT.

**Usage** This method returns true only if the referenced PBDOM\_OBJECT is also a derived PBDOM\_ENTITYREFERENCE object *and* it refers to the same DOM object as the current PBDOM\_ENTITYREFERENCE object. Two separately created PBDOM\_COMMENTS, for example, can contain exactly the same text but not be equal.

## GetName

Description Obtains the name of the current PBDOM\_ENTITYREFERENCE object.

Syntax *pbdom\_entityref\_name*.GetName()

Argument	Description
<i>pbdom_entityref_name</i>	The name of a PBDOM_ENTITYREFERENCE object

Return value String.

See also SetName

## GetObjectClass

Description Returns a long integer code that indicates the class of the current PBDOM\_OBJECT.

Syntax *pbdom\_object\_name*.GetObjectClass()

Argument	Description
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

Return value Long. A code that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_ENTITYREFERENCE object, the returned value is 11.

See also GetObjectClassString

## GetObjectClassString

Description Returns a string form of the class of the PBDOM\_OBJECT.

Syntax *pbdom\_object\_name*.GetObjectClassString()

Argument	Description
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

Return value String. A string that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_ENTITYREFERENCE object, the returned string is “pbdom\_entityreference”.

See also GetObjectClass

## GetOwnerDocumentObject

**Description** The GetOwnerDocumentObject method returns the owning PBDOM\_DOCUMENT of the current PBDOM\_ENTITYREFERENCE object.

**Syntax** *pbdom\_entityref\_name*.GetOwnerDocumentObject()

<b>Argument</b>	<b>Description</b>
<i>pbdom_entityref_name</i>	The name of a PBDOM_ENTITYREFERENCE object

**Return value** PBDOM\_DOCUMENT.

**Usage** If there is no owning PBDOM\_DOCUMENT, null is returned.

**See also** GetParentObject  
SetParentObject

## GetParentObject

**Description** The GetParentObject method returns the parent PBDOM\_OBJECT of the current PBDOM\_ENTITYREFERENCE object.

**Syntax** *pbdom\_entityref\_name*.GetParentObject()

<b>Argument</b>	<b>Description</b>
<i>pbdom_entityref_name</i>	The name of a PBDOM_ENTITYREFERENCE object

**Return value** PBDOM\_OBJECT.

**Usage** The GetParentObject method returns the parent PBDOM\_OBJECT of the current PBDOM\_ENTITYREFERENCE object. If the PBDOM\_ENTITYREFERENCE object has no parent, null is returned.

**See also** GetOwnerDocumentObject  
SetParentObject



## SetName

**Description** Changes the name of the PBDOM\_ENTITYREFERENCE object, effectively making it refer to another DOM entity object.

**Syntax** `pbdom_entityref_name.SetName(string strName)`

Argument	Description
<i>pbdom_entityref_name</i>	The name of a PBDOM_ENTITYREFERENCE object
<i>strName</i>	The new name you want to set for the current PBDOM_ENTITYREFERENCE object

**Return value** Boolean. Returns true if the name of the current PBDOM\_ENTITYREFERENCE object was changed, and false if it was not.

**See also** GetName

## SetParentObject

**Description** The SetParentObject method sets the referenced PBDOM\_OBJECT to be the parent of the current PBDOM\_ENTITYREFERENCE object.

**Syntax** `pbdom_entityref_name.SetParentObject(pbdom_object pbdom_object_ref)`

Argument	Description
<i>pbdom_entityref_name</i>	The name of a PBDOM_ENTITYREFERENCE object
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to be set as the parent of the current PBDOM_ENTITYREFERENCE object

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not an object derived from PBDOM\_OBJECT.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – If the current PBDOM\_ENTITYREFERENCE object already has a parent.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT is of a class that does not have a legal parent-child relationship with the PBDOM\_ENTITYREFERENCE class.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT requires a user-defined name and it has not been named, or the name of the entity reference object has not been set.

Usage	This method sets the input PBDOM_OBJECT to be the parent of this PBDOM_ENTITYREFERENCE object. The caller is responsible for ensuring that the current PBDOM_ENTITYREFERENCE object and the input PBDOM_OBJECT can have a legal parent-child relationship. Currently only a PBDOM_ELEMENT or a PBDOM_ATTRIBUTE can be set as the parent of a PBDOM_ENTITYREFERENCE object.
See also	GetOwnerDocumentObject GetParentObject

# PBDOM\_CHARACTERDATA Class

About this document

This chapter describes the PBDOM\_CHARACTERDATA class.

## PBDOM\_CHARACTERDATA

Description

The PBDOM\_CHARACTERDATA class represents character-based content (not markup) within an XML document. It extends the PBDOM\_OBJECT class with a set of methods specifically intended for manipulating character data in the DOM.

The PBDOM\_CHARACTERDATA class is the parent class of three other PBDOM classes:

- PBDOM\_TEXT
- PBDOM\_CDATA
- PBDOM\_COMMENT

The PBDOM\_CHARACTERDATA class, like its parent class PBDOM\_OBJECT, is a “virtual” class (similar to a virtual C++ class) in that it is not expected to be directly instantiated and used.

For example, in the following code, the attempt to set the text of pbdom\_chrdata raises an exception:

```
PBDOM_CHARACTERDATA pbdom_chrdata
pbdom_chrdata = CREATE PBDOM_CHARACTERDATA
pbdom_chrdata.SetText ("character string");//error
```

In this example, the attempt to set the text of pbdom\_chrdata succeeds because pbdom\_chrdata is declared as a PBDOM\_CHARACTERDATA but instantiated as a PBDOM\_TEXT:

```
PBDOM_CHARACTERDATA pbdom_chrdata
pbdom_chrdata = CREATE PBDOM_TEXT
pbdom_chrdata.SetText ("character string");//success
```

Methods

Some of the inherited methods from PBDOM\_OBJECT serve no meaningful objective and only default or trivial functionalities result. These are described in the following table:

Method	Always returns
AddContent	current PBDOM_CHARACTERDATA
GetContent	false
InsertContent	current PBDOM_CHARACTERDATA
RemoveContent	false
SetContent	current PBDOM_CHARACTERDATA
SetName	false

PBDOM\_CHARACTERDATA has the following non-trivial methods:

Append	GetParentObject
Clone	GetText
Detach	GetTextNormalize
Equals	GetTextTrim
GetName	HasChildren
GetObjectClass	IsAncestorObjectOf
GetObjectClassString	SetParentObject
GetOwnerDocumentObject	SetText

## Append

Description

The Append method is overloaded:

- Syntax 1 appends an input string to the text content that already exists within the current PBDOM\_CHARACTERDATA object.
- Syntax 2 appends the text data of a PBDOM\_CHARACTERDATA object to the text content that already exists within the current PBDOM\_CHARACTERDATA object.

Syntax

For this syntax	See
Append(string <i>strAppend</i> )	Append Syntax 1
Append(pbdom_characterdata <i>pbdom_characterdata_ref</i> )	Append Syntax 2

## Append Syntax 1

**Description** Appends an input string to the text content that already exists within the current PBDOM\_CHARACTERDATA object.

**Syntax** `pbdom_text_name.Append(string strAppend)`

Argument	Description
<code>pbdom_text_name</code>	The name of a PBDOM_CHARACTERDATA object
<code>strAppend</code>	The string you want appended to the existing text of the current PBDOM_CHARACTERDATA object

**Return value** PBDOM\_CHARACTERDATA. The current PBDOM\_CHARACTERDATA modified and returned as a PBDOM\_CHARACTERDATA object.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA object is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Examples** In this example, the PowerScript code builds a PBDOM\_DOCUMENT based on the following DOM Tree:

```
<abc>
  <data>
    <child_1>
      My Text
    </child_1>
    <child_2>
      <!--My Comment-->
    </child_2>
    <child_3>
      <![CDATA[My CDATA]]>
    </child_3>
  </data>
</abc>
```

The root element `abc` has a child element, `data`, that has three child elements. `child_1` contains a child PBDOM\_TEXT with the string “My Text”. `child_2` contains a child PBDOM\_COMMENT with the string “My Comment”. `child_3` contains a child PBDOM\_CDATA with the string “My CDATA”.

In the following PowerScript code, the single statement that follows the comment `// obtain the child PBDOM_TEXT of child_1` does the following:

- 1 Obtains the root element of the PBDOM\_DOCUMENT `pbdom_doc` using `GetRootElement`. A new PBDOM\_ELEMENT representing the root element `abc` is created in memory and returned.

- 2 Calls the GetChildElement method on the returned root abc PBDOM\_ELEMENT using data as the parameter to single out the data child element. A PBDOM\_ELEMENT representing the data element is created in memory and returned.
- 3 Calls the GetChildElement on the returned data PBDOM\_ELEMENT, using child\_1 as the parameter to single out the child\_1 child element. A PBDOM\_ELEMENT representing the child\_1 element is created in memory and returned.
- 4 Calls the GetContent method on the returned child\_1 PBDOM\_ELEMENT, supplying a reference to the unbounded array pbdom\_chardata\_array.

You can supply PBDOM\_CHARACTERDATA array instead of a PBDOM\_OBJECT array because PBDOM\_CHARACTERDATA is a subclass of PBDOM\_OBJECT. However, GetContent fails if child\_1 contains any objects other than PBDOM\_CHARACTERDATA objects.

Because child\_1 holds only the PBDOM\_TEXT containing the string “My Text”, this statement returns an array that has only one array item. The next statement appends another string to the array item. The example then repeats these steps for child\_2 and child\_3 and saves pbdom\_doc to a file:

```
PBDOM_Builder          pbdombuilder_new
pbdom_document         pbdom_doc
PBDOM_CHARACTERDATA   pbdom_chardata_array[]

string strXML = "<abc><data><child_1>My
Text</child_1><child_2><!--My Comment--
></child_2><child_3><![CDATA[My
CDATA]]></child_3></data></abc>"

TRY
    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString (strXML)
```

```

// obtain the child PBDOM_TEXT of child_1
pbdom_doc.GetRootElement().GetChildElement("data").&
  GetChildElement("child_1"). &
  GetContent(pbdom_chardata_array)

// append the string "Now Appended" to the text
// returned by the call to GetContent
pbdom_chardata_array[1].Append (" Now Appended")

// repeat for child_2 and child_3
pbdom_doc.GetRootElement().GetChildElement("data").&
  GetChildElement("child_2"). &
  GetContent(pbdom_chardata_array)
pbdom_chardata_array[1].Append (" Now Appended")

pbdom_doc.GetRootElement().GetChildElement("data").&
  GetChildElement("child_3"). &
  GetContent(pbdom_chardata_array)
pbdom_chardata_array[1].Append (" Now Appended")

// save pbdom_doc to a file
pbdom_doc.SaveDocument ("c:\pbdom_doc_1.xml")

Destroy pbdombuilder_new

CATCH (PBDOM_Exception except)
  MessageBox ("Exception Occurred", except.Text)
END TRY

```

The saved file contains the following:

```

<abc>
  <data>
    <child_1>
      My Text Now Appended
    </child_1>
    <child_2>
      <!--My Comment Now Appended-->
    </child_2>
    <child_3>
      <![CDATA[My CDATA Now Appended]]>
    </child_3>
  </data>
</abc>

```

## Append Syntax 2

**Description** Appends the text data of a PBDOM\_CHARACTERDATA object to the text content that already exists within the current PBDOM\_CHARACTERDATA object.

**Syntax** *pbdom\_text\_name*.Append(*pbdom\_characterdata pbdom\_characterdata\_ref*)

<b>Argument</b>	<b>Description</b>
<i>pbdom_text_name</i>	The name of a PBDOM_CHARACTERDATA
<i>pbdom_characterdata_ref</i>	The referenced PBDOM_CHARACTERDATA object whose text data is to be appended to the existing text of the current PBDOM_CHARACTERDATA object

**Return value** PBDOM\_CHARACTERDATA. The current PBDOM\_CHARACTERDATA modified and returned as a PBDOM\_CHARACTERDATA object.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the current PBDOM\_CHARACTERDATA or the input PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Usage** Note that JDOM does not define an Append method for its CHARACTERDATA class. Because PBDOM implements its Append method in the base PBDOM\_CHARACTERDATA class, a PBDOM\_TEXT object, a PBDOM\_CDATA object, and a PBDOM\_TEXT object can append their internal text data to each other because they are all PBDOM\_CHARACTERDATA-derived objects.



## Clone

**Description** Creates and returns a clone of the current PBDOM\_CHARACTERDATA.

**Syntax** `pbdom_chardata_name.Clone(boolean bDeep)`

Argument	Description
<code>pbdom_chardata_name</code>	The name of a PBDOM_CHARACTERDATA.
<code>bDeep</code>	A boolean specifying whether a deep or shallow clone is returned. Values are <code>true</code> for a deep clone and <code>false</code> for a shallow clone. This argument is currently ignored.

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Examples** This example creates a PBDOM\_DOCUMENT based on the following DOM tree:

```
<abc>
  <data>Data</data>
</abc>
```

The PowerScript code obtains the data element of the root element as a PBDOM\_ELEMENT and obtains an array of its children. The array has only one item, the PBDOM\_TEXT containing the string “data”:

```
PBDOM_BUILDER pbdombuilder_new
PBDOM_DOCUMENT pbdom_doc
PBDOM_ELEMENT pbdom_elem
PBDOM_CHARACTERDATA pbdom_chardata_1
PBDOM_CHARACTERDATA pbdom_chardata_2
PBDOM_CHARACTERDATA pbdom_chardata_3
PBDOM_OBJECT pbdom_obj_array[]
string strXML = "<abc><data>Data</data></abc>"

TRY
    pbdombuilder_new = CREATE PBDOM_BUILDER
    pbdom_doc = pbdombuilder_new.BuildFromString
    (strXML)

    // get the data element, store in pbdom_elem,
    // and get an array of its children
    pbdom_elem = pbdom_doc.GetRootElement(). &
        GetChildElement("data")
    pbdom_elem.GetContent(pbdom_obj_array)
```

This PBDOM\_TEXT is assigned into a PBDOM\_CHARACTERDATA object, pbdom\_chardata\_1. Calling GetObjectClassString on pbdom\_chardata\_1 returns the class name of the actual object contained within it, pbdom\_text. Calling GetText on it returns the string Data:

```
pbdom_chardata_1 = pbdom_obj_array[1]
MessageBox ("Class", &
    pbdom_chardata_1.GetObjectClassString())
MessageBox ("Text", pbdom_chardata_1.GetText())
```

Calling Clone on pbdom\_chardata\_1 creates a new PBDOM\_CHARACTERDATA object. However, because the actual object referenced by pbdom\_chardata\_1 is a PBDOM\_TEXT, the clone is a PBDOM\_TEXT object.

Calling GetObjectClassString and GetText on the clone have the same result as for pbdom\_chardata\_1. The clone and the original object are separate objects and a call to Equals returns false:

```
pbdom_chardata_2 = pbdom_chardata_1.Clone(TRUE)
MessageBox ("Class", &
    pbdom_chardata_2.GetObjectClassString())
MessageBox ("Text", pbdom_chardata_2.GetText())
if (pbdom_chardata_1.Equals(pbdom_chardata_2)) then
    MessageBox ("Equals", &
        "pbdom_chardata_1 equals pbdom_chardata_2")
else
    MessageBox ("Equals", &
        "pbdom_chardata_1 NOT equals pbdom_chardata_2")
end if
```

However, a call to Equals returns true if the object being compared to pbdom\_chardata\_1 is a reference to pbdom\_chardata\_1:

```
pbdom_chardata_3 = pbdom_chardata_1
if (pbdom_chardata_1.Equals(pbdom_chardata_3)) then
    MessageBox ("Equals", &
        "pbdom_chardata_1 equals pbdom_chardata_3")
else
    MessageBox ("Equals", &
        "pbdom_chardata_1 NOT equals pbdom_chardata_3")
end if
```

```
DESTROY pbdombuilder_new
```

```
CATCH (PBDOM_Exception except)
    MessageBox ("Exception Occurred", except.Text)
END TRY
```

**Usage** The Clone method creates a new PBDOM\_CHARACTERDATA object which is a duplicate of, and a separate object from, the original. Calling Equals using these two objects returns false.

The clone of a PBDOM\_CHARACTERDATA object is always identical to its original whether *bDeep* is true or false, because a PBDOM\_CHARACTERDATA object contains no subtree of child PBDOM\_OBJECTs.

A PBDOM\_CHARACTERDATA clone has no parent, but it resides in the same PBDOM\_DOCUMENT as its original, and if the original PBDOM\_CHARACTERDATA is standalone, the clone is standalone.

## Detach

**Description** Detaches a PBDOM\_CHARACTERDATA object from its parent.

**Syntax** `pbdom_chardata_name.Detach()`

Argument	Description
<i>pbdom_chardata_name</i>	The name of a PBDOM_CHARACTERDATA object

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Examples** This example creates a PBDOM\_DOCUMENT based on the following DOM tree:

```
<abc>
  <data>Data</data>
</abc>
```

The PowerScript code obtains the root element, uses it to obtain the child element, and then obtains an array of the child element's own children. This array has a single item, the PBDOM\_TEXT object with the text Data. The array can be cast to a PBDOM\_CHARACTERDATA object because it does not contain any objects that are not derived from PBDOM\_CHARACTERDATA.

Calling Detach separates the PBDOM\_TEXT object from its parent PBDOM\_OBJECT, data.

```
PBDOM_Builder          pbdombuilder_new
pbdom_document         pbdom_doc
pbdom_document         pbdom_owner_doc
PBDOM_CHARACTERDATA   pbdom_chardata
PBDOM_OBJECT           pbdom_obj_array[]
string strXML = "<abc><data>Data</data></abc>"

TRY
    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString
                (strXML)

    pbdom_doc.GetRootElement(). &
                GetChildElement("data"). &
                GetContent(pbdom_obj_array)

    pbdom_chardata = pbdom_obj_array[1]
    pbdom_chardata.Detach()
    pbdom_doc.SaveDocument("c:\pbdom_doc_1.xml")
    Destroy pbdombuilder_new
CATCH (PBDOM_Exception except)
    MessageBox ("Exception Occurred", except.Text)
END TRY
```

When the document is saved to a file, the file's contents are as follows, because the PBDOM\_TEXT object was removed from data:

```
<abc>
  <data/>
</abc>
```

**Usage**

Nothing occurs if the PBDOM\_CHARACTERDATA object has no parent.

## Equals

**Description** Tests for the equality of the current PBDOM\_CHARACTERDATA and a referenced PBDOM\_OBJECT.

**Syntax** `pbdom_chardata_name.Equals(pbdom_object pbdom_object_ref)`

Argument	Description
<i>pbdom_chardata_name</i>	The name of a PBDOM_CHARACTERDATA object
<i>pbdom_object_ref</i>	A reference to a PBDOM_OBJECT to test for equality with the current PBDOM_CHARACTERDATA object

**Return value** Boolean. Returns true if the current PBDOM\_CHARACTERDATA is equivalent to the input PBDOM\_OBJECT and false otherwise.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Usage** True is returned only if the referenced PBDOM\_OBJECT is also a derived PBDOM\_CHARACTERDATA object and refers to the same DOM object as the current PBDOM\_CHARACTERDATA. Two separately created PBDOM\_COMMENTS, for example, can contain exactly the same text but are not equal.

**See also** Clone

## GetOwnerDocumentObject

**Description** The GetOwnerDocumentObject method returns the owning PBDOM\_DOCUMENT of the current PBDOM\_CHARACTERDATA.

**Syntax** *pbdom\_chardata\_name*.GetOwnerDocumentObject()

Argument	Description
<i>pbdom_chardata_name</i>	The name of a PBDOM_CHARACTERDATA object

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not associated with a derived PBDOM\_CHARACTERDATA class.

**Examples** **Example 1** This example creates a PBDOM\_DOCUMENT based on the following DOM tree:

```
<abc>
  <data>Data</data>
</abc>
```

The PowerShell code obtains the root element, uses it to obtain the child element, and then obtains an array of the child element's own children. This array has a single item, the PBDOM\_TEXT object with the text Data. The array can be cast to a PBDOM\_CHARACTERDATA object because it does not contain any objects that are not derived from PBDOM\_CHARACTERDATA,

The call to GetOwnerDocumentObject returns a PBDOM\_OBJECT, which is stored in a PBDOM\_DOCUMENT called *pbdom\_owner\_doc*. The call to Equals tests whether the owner document of the "Data" PBDOM\_TEXT and the main document, referenced using *pbdom\_doc*, refer to the same document.

```
PBDOM_Builder      pbdombuilder_new
pbdom_document     pbdom_doc
pbdom_document     pbdom_owner_doc
pbdom_element      pbdom_elem
PBDOM_CHARACTERDATA pbdom_chardata
PBDOM_OBJECT       pbdom_obj_array[]
string strXML = "<abc><data>Data</data></abc>"

TRY
    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString
    (strXML)

    pbdom_elem = pbdom_doc.GetRootElement(). &
```

```

        GetChildElement("data")
        pbdom_elem.GetContent(pbdom_obj_array)

        pbdom_chardata = pbdom_obj_array[1]

        pbdom_owner_doc = &
            pbdom_chardata.GetOwnerDocumentObject()

        if (pbdom_doc.Equals(pbdom_owner_doc)) then
            MessageBox("Equals", &
                "pbdom_doc Equals pbdom_owner_doc")
        else
            MessageBox("Equals", &
                "pbdom_doc Not Equals pbdom_owner_doc")
        end if

        Destroy pbdombuilder_new

    CATCH (PBDOM_Exception except)
        MessageBox("Exception Occurred", except.Text)
    END TRY

```

**Example 2** This example creates a PBDOM\_DOCUMENT based on the same DOM tree as example 1. It creates a PBDOM\_TEXT, stores it in the PBDOM\_CHARACTERDATA variable pbdom\_chardata, and assigns it some text. Objects created in this way are standalone objects—they have no owner document or parent. Calling GetOwnerDocumentObject on pbdom\_chardata returns null.

The code then adds pbdom\_chardata as a child to the data element. This implicitly imports pbdom\_chardata into the original document. pbdom\_chardata now has an owner document and a parent (the data element). Calling GetOwnerDocumentObject on pbdom\_chardata returns the original document. When the returned PBDOM\_DOCUMENT has been assigned into pbdom\_owner\_doc, a call to Equals to compare pbdom\_doc with pbdom\_owner\_doc returns true:

```

PBDOM_Builder          pbdombuilder_new
pbdom_document        pbdom_doc
pbdom_document        pbdom_owner_doc
PBDOM_CHARACTERDATA  pbdom_chardata
string strXML = "<abc><data>Data</data></abc>"

TRY
    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString (strXML)

```

```
pbdom_chardata = Create PBDOM_TEXT
pbdom_chardata.SetText(" Some Text")

if (IsValid (pbdom_chardata.GetOwnerDocumentObject())) then
    MessageBox ("Owner Document", &
        "PBDOM_TEXT (~'Some Text~') has an owner document.")
else
    MessageBox ("Owner Document", &
        "PBDOM_TEXT (~'Some Text~') has NO owner document.")
end if

pbdom_doc.GetRootElement().GetChildElement("data"). &
    AddContent(pbdom_chardata)

pbdom_owner_doc = pbdom_chardata.GetOwnerDocumentObject()

if (pbdom_doc.Equals(pbdom_owner_doc)) then
    MessageBox ("Equals", "pbdom_doc Equals pbdom_owner_doc")
else
    MessageBox ("Equals", "pbdom_doc Not Equals pbdom_owner_doc")
end if

Destroy pbdombuilder_new
Destroy pbdom_chardata

CATCH (PBDOM_Exception except)
    MessageBox ("Exception Occurred", except.Text)
END TRY
```

**Usage**                      If there is no owning PBDOM\_DOCUMENT, null is returned.

**See also**                      GetParentObject  
                                 SetParentObject



## GetName

**Description** The GetName method allows you to obtain the name of the current PBDOM\_CHARACTERDATA.

**Syntax** `pbdom_chardata_name.GetName()`

Argument	Description
<i>pbdom_chardata_name</i>	The name of a PBDOM_CHARACTERDATA object

**Return value** String.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Usage** The returned string depends on the specific type of DOM object that is contained within PBDOM\_CHARACTERDATA.

---

### Note

A PBDOM\_CHARACTERDATA is abstract and is not to be instantiated into an object of its own. Thus, there is no name returned as “#characterdata”.

---

The following table lists the return values based on the type of DOM Object contained within PBDOM\_CHARACTERDATA.

DOM Object	Return Value
PBDOM_CDATA	"#cdata-section"
PBDOM_COMMENT	"#comment"
PBDOM_TEXT	"#text"

## GetObjectClass

**Description** The GetObjectClass method returns a long integer code that indicates the class of the current PBDOM\_OBJECT.

**Syntax** *pbdom\_object\_name*.GetObjectClass()

<b>Argument</b>	<b>Description</b>
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

**Return value** Long. GetObjectClass returns a long integer value that indicates the class of the current PBDOM\_OBJECT.

The possible return values for classes inherited from PBDOM\_CHARACTERDATA are:

- 7 for PBDOM\_TEXT
- 8 for PBDOM\_CDATA
- 9 for PBDOM\_COMMENT

The PBDOM\_CHARACTERDATA class itself cannot be instantiated, so the class ID 6, for PBDOM\_CHARACTERDATA, is never returned.

**See also** GetObjectClassString

## GetObjectClassString

**Description** The GetObjectClassString method returns a string form of the class of the PBDOM\_OBJECT.

**Syntax** *pbdom\_object\_name*.GetObjectClassString()

<b>Argument</b>	<b>Description</b>
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

**Return value** String. GetObjectClassString returns a string that indicates the class of the current PBDOM\_OBJECT.

The possible return values for classes inherited from PBDOM\_CHARACTERDATA are:

- pbdom\_text
- pbdom\_cdata
- pbdom\_comment

The PBDOM\_CHARACTERDATA class itself cannot be instantiated, so the string “pbdom\_characterdata” is never returned.

#### Examples

This example creates a PBDOM\_DOCUMENT based on the following DOM tree:

```
<abc>
  <data>
    Data with a &lt; character
    <!-- Comment with a &lt; character -->
    <![CDATA[ CDATA with an actual > character and
      an entity reference &lt; ]]>
  </data>
</abc>
```

The PowerScript code obtains the root element, uses it to obtain the child element, and then obtains an array of the child element’s own children. This is an array of three PBDOM\_OBJECTs, each of which is a child node of data. This array provides the ability to access and manipulate the child nodes, but to illustrate the virtual nature of the PBDOM\_CHARACTERDATA class and the calling of methods of the PBDOM\_CHARACTERDATA class, the example defines an array of PBDOM\_CHARACTERDATA objects.

Each array item of the pbdom\_obj\_array is assigned to the pbdom\_chardata array, so you can call the methods of each array item without needing to know what subclass the item belongs to.

---

#### Children must be subclasses of PBDOM\_CHARACTERDATA

If the data element contained a child that was not a subclass of PBDOM\_CHARACTERDATA, the FOR loop to assign each pbdom\_obj\_array item to a corresponding pbdom\_chardata array item would fail when it reached that item.

---

The MessageBox calls illustrate how the entity reference &lt; is handled by the different PBDOM\_CHARACTERDATA subclasses. In the PBDOM\_TEXT object, it is expanded. In the PBDOM\_COMMENT and PBDOM\_CDATA objects, it is not. The character to which the entity reference refers, “>”, can also be included in a PBDOM\_CDATA object.

```
PBDOM_Builder          pbdombuilder_new
pbdom_document        pbdom_doc
pbdom_element         pbdom_elem
PBDOM_CHARACTERDATA  pbdom_chardata[]
PBDOM_OBJECT          pbdom_obj_array[]
long l = 0
string strXML = "<abc><data>Data with a &lt;
```

```

character<!-- Comment with a &lt; character --
><![CDATA[ CDATA with an actual > character and an
entity reference &lt; ]]></data></abc>"

TRY
    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString
    (strXML)

    pbdom_elem = pbdom_doc.GetRootElement(). &
        GetChildElement("data")
    pbdom_elem.GetContent(pbdom_obj_array)

// populate an array of PBDOM_CHARACTERDATA objects
for l = 1 to UpperBound(pbdom_obj_array)
    pbdom_chardata[l] = pbdom_obj_array[l]
next
for l = 1 to UpperBound(pbdom_chardata)
    MessageBox ("Class", &
        pbdom_chardata[l].GetObjectClassString())
    MessageBox ("Text", pbdom_chardata[l].GetText())
next

Destroy pbdombuilder_new

CATCH (PBDOM_Exception except)
    MessageBox ("Exception Occurred", except.Text)
END TRY

```

See also [GetObjectClass](#)

## GetParentObject

**Description** The GetParentObject method returns the parent PBDOM\_OBJECT of the current PBDOM\_CHARACTERDATA.

**Syntax** *pbdom\_chardata\_name*.GetParentObject()

Argument	Description
<i>pbdom_chardata_name</i>	The name of a PBDOM_CHARACTERDATA object

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

## Examples

This example creates a PBDOM\_DOCUMENT based on the following DOM tree and demonstrates how a PBDOM\_CHARACTERDATA INSTANCE can be detached from its parent:

```
<abc>
  <data>Data</data>
</abc>
```

The PowerScript code obtains the root element, uses it to obtain the child element, and then obtains an array of the child element's own children. This array has a single item, the PBDOM\_TEXT object with the text Data. The array can be cast to a PBDOM\_CHARACTERDATA object, because it does not contain any objects that are not derived from PBDOM\_CHARACTERDATA.

The parent of pbdom\_chardata\_1 is the data element. The following steps detach it from its parent:

- Create a PBDOM\_COMMENT in the PBDOM\_CHARACTERDATA object pbdom\_chardata\_2 and assign to it the text "Some Comments".
- Set pbdom\_chardata\_2 as an array item of pbdom\_obj\_array.
- Call SetContent on the parent of pbdom\_chardata\_1 (the data element).

Calling SetContent resets the contents of data, which can cause its original contents (including pbdom\_chardata\_1) to be removed, depending on what is stored inside pbdom\_obj\_array. Because pbdom\_obj\_array contains only the newly created PBDOM\_COMMENT, pbdom\_chardata\_2, data will have only this PBDOM\_COMMENT as its child.

pbdom\_chardata\_1 will have no parent, because it has been silently detached from it. Calling GetParentObject on it will return null:

```
PBDOM_Builder          pbdombuilder_new
pbdom_document         pbdom_doc
pbdom_document        pbdom_owner_doc
PBDOM_CHARACTERDATA   pbdom_chardata_1
PBDOM_CHARACTERDATA   pbdom_chardata_2
PBDOM_OBJECT          pbdom_obj_array[]
string strXML = "<abc><data>Data</data></abc>"
```

TRY

```
pbdombuilder_new = Create PBDOM_Builder
pbdom_doc = pbdombuilder_new.BuildFromString (strXML)
```

```
pbdom_doc.GetRootElement(). &
  GetChildElement("data"). &
  GetContent(pbdom_obj_array)
```

```
pbdom_chardata_1 = pbdom_obj_array[1]

pbdom_chardata_2 = Create PBDOM_COMMENT
pbdom_chardata_2.SetText ("Some Comments")

pbdom_obj_array[1] = pbdom_chardata_2

pbdom_chardata_1.GetParentObject(). &
  SetContent (pbdom_obj_array)

if (IsValid(pbdom_chardata_1.GetParentObject())) then
  MessageBox ("Has Parent Object", &
    "PBDOMTEXT (~'Data~') has a parent")
else
  MessageBox ("Has Parent Object", &
    "PBDOMTEXT (~'Data~') has NO parent")
end if

pbdom_doc.SaveDocument ("c:\pbdom_doc_1.xml")

Destroy pbdombuilder_new
Destroy pbdom_chardata_2

CATCH (PBDOM_Exception except)
  MessageBox ("Exception Occurred", except.Text)
END TRY
```

When the resulting PBDOM\_DOCUMENT is saved to a file, it looks like this:

```
<abc>
  <data>
    <!-- Some Comments -->
  </data>
</abc>
```

### Usage

The parent is also an object derived from PBDOM\_CHARACTERDATA. If the PBDOM\_OBJECT has no parent, null is returned.

### See also

SetParentObject

## GetText

**Description** Calling the GetText method allows you to obtain text data that is contained within the current PBDOM\_CHARACTERDATA.

**Syntax** `pbdom_chardata_name.GetText()`

Argument	Description
<code>pbdom_chardata_name</code>	The name of a PBDOM_CHARACTERDATA object

**Return value** String. The text of the current PBDOM\_CHARACTERDATA-derived object.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Usage** The following table lists the return values based on the type of DOM Object contained within PBDOM\_CHARACTERDATA.

DOM Object	Return Value
PBDOM_TEXT	<p>The text data contained within the PBDOM_TEXT object itself.</p> <p>For example, suppose you have the following element:</p> <pre>&lt;abc&gt;MY TEXT&lt;/abc&gt;</pre> <p>If you have a PBDOM_TEXT object to represent the TEXT NODE “MY TEXT”, then calling GetText on the PBDOM_TEXT returns the string MY TEXT.</p>
PBDOM_CDATA	<p>The string data that is contained within the CDATA section itself. For example, suppose you have the following CDATA:</p> <pre>&lt;![CDATA[ They're saying "x &lt; y" &amp; that "z &gt; y" so I guess that means that z &gt; x ]]&gt;</pre> <p>If there is a PBDOM_CDATA to represent the above CDATA section, then calling GetText returns the string:</p> <pre>They're saying "x &lt; y" &amp; that "z &gt; y" so I guess that means that z &gt; x</pre>
PBDOM_COMMENT	<p>The comment itself. For example, suppose you have the following comment:</p> <pre>&lt;!--This is a comment. --&gt;</pre> <p>Calling GetText on the comment returns the string:</p> <pre>This is a comment.</pre>

**See also** GetTextNormalize  
GetTextTrim  
SetText

## GetTextNormalize

**Description** The GetTextNormalize method allows you to obtain the text data that is contained within the current PBDOM\_CHARACTERDATA object, with all surrounding whitespace characters removed and internal whitespace characters normalized to a single space.

**Syntax** `pbdom_chardata_name.GetTextNormalize()`

Argument	Description
<code>pbdom_chardata_name</code>	The name of a PBDOM_CHARACTERDATA object

**Return value** String. The following table lists the return values, based on the type of DOM object contained within PBDOM\_CHARACTERDATA.

DOM Object	Return Value
PBDOM_TEXT	<p>Suppose you have the following element:</p> <pre>&lt;abc&gt; MY TEXT &lt;/abc&gt;</pre> <p>If there is a PBDOM_TEXT object to represent the TEXT NODE "MY TEXT", then calling GetTextNormalize on the PBDOM_TEXT returns the string MY TEXT.</p>
PBDOM_CDATA	<p>Suppose there is the following CDATA:</p> <pre>&lt;![CDATA] They're saying "x &lt; y" &amp; that "z &gt; y" so I guess that means that z &gt; x ]]&gt;</pre> <p>If there is a PBDOM_CDATA to represent the above CDATA section, then calling GetTextNormalize on it returns the string:</p> <pre>They're saying " x &lt; y " &amp; that "z &gt; y" so I guess that means that z &gt; x</pre> <p>Note that the initial spaces before "They're" and the trailing space after the last "x" are removed. Additionally, the spaces between the words "guess" and "that" are reduced to just one space.</p>
PBDOM_COMMENT	<p>Suppose there is the following comment:</p> <pre>&lt;!--This is a comment --&gt;</pre> <p>Calling GetTextNormalize on this comment returns:</p> <pre>This is a comment</pre>

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.



## Examples

This example demonstrates:

- 1 Using an external general parsed entity.
- 2 Using a single line statement to obtain the children PBDOM\_OBJECTs of an element.
- 3 Obtaining the text of the three separate types of PBDOM\_CHARACTERDATA objects : PBDOM\_TEXT, PBDOM\_COMMENT, and PBDOM\_CDATA.
- 4 Obtaining the normalized text of the same three separate types of PBDOM\_CHARACTERDATA objects.
- 5 The difference between the two types of text retrieved in 3 and 4.

Suppose the file *C:\entity\_text.txt* contains the following string:

```
&#9;&#32;Some&#32;External&#32;&#32;&#9;&#32;Text&#32;
&#9;
```

The example creates a PBDOM\_DOCUMENT `pbdom_doc` based on the following DOM tree, which is in the file *C:\inputfile.txt*:

```
<!DOCTYPE abc [<!ENTITY text1 SYSTEM
"c:\entity_text.txt" >]>
<abc>
  <data>
    &text1;
    <!-- &text1;-->
    <![CDATA[&text1;]]>
  </data>
</abc>
```

The Document Type Declaration defines an external general parsed entity `text1`.

The example obtains the root element, uses it to obtain the data child element, and then obtains an array of the child element's own children. PBDOM collects all the PBDOM\_OBJECTs that are the children of data and stores them in the PBDOM\_OBJECT array `pbdom_obj_array`.

Next, the FOR loop iterates through all the items in `pbdom_obj_array` and stores each item in the `PBDOM_CHARACTERDATA` array `pbdom_chardata`. This step is not required—the `pbdom_obj_array` can be used to manipulate the data element's children. It is done to demonstrate that you can cast each item into a `PBDOM_CHARACTERDATA` object by assigning it into a `PBDOM_CHARACTERDATA` array. This is possible if and only if each `PBDOM_OBJECT` is also derived from `PBDOM_CHARACTERDATA`. If a `PBDOM_OBJECT` is not derived from `PBDOM_CHARACTERDATA`, the PowerBuilder VM throws an exception.

The next FOR loop iterates through all the items of the `pbdom_chardata` array and calls the `GetText` and `GetTextNormalize` methods on each. Each of the returned strings from `GetText` and `GetTextNormalize` is delimited by “[” and “]” characters so that the complete text content displays clearly in the message boxes.

The first child of data is the `PBDOM_TEXT &text1;`, which has been declared as an external general parsed entity whose content is the content of the file `c:\entity_text.txt`. The `&text1;` entity reference and the entity references it contains are expanded by the parser. The call to `GetTextNormalize` strips away the whitespace characters.

The second child of data is the `PBDOM_COMMENT <!-- &text1;-->` and the third child is the `PBDOM_CDATA <![CDATA[&text1;]]>`. Entity references within comments and CDATA sections are never expanded. Both `GetText` and `GetTextNormalize` return `&text1;`.

```
PBDOM_Builder      pbdombuilder_new
pbdom_document     pbdom_doc
PBDOM_CHARACTERDATA pbdom_chardata []
PBDOM_OBJECT       pbdom_obj_array []
integer            iFileNum1
long               l = 0

TRY
  pbdombuilder_new = Create PBDOM_Builder
  pbdom_doc = pbdombuilder_new.BuildFromFile &
    ("C:\inputfile.txt")

  pbdom_doc.GetRootElement(). &
    GetChildElement("data"). &
    GetContent(pbdom_obj_array)

  for l = 1 to UpperBound(pbdom_obj_array)
    pbdom_chardata[l] = pbdom_obj_array[l]
  next
```

```
for l = 1 to UpperBound(pbdom_chardata)
  MessageBox(pbdom_chardata[l]. &
    GetObjectClassString() + "GetText()", &
    "[" + pbdom_chardata[l].GetText() + "]")
  MessageBox(pbdom_chardata[l]. &
    GetObjectClassString() + " GetTextNormalize()", &
    "[" + pbdom_chardata[l].GetTextNormalize() + "]")
next

Destroy pbdombuilder_new

CATCH (PBDOM_Exception except)
  MessageBox ("Exception Occurred", except.Text)
END TRY
```

**Usage** If no textual value exists for the current PBDOM\_OBJECT, or if only whitespace characters exist, an empty string is returned.

**See also** [GetText](#)  
[GetTextTrim](#)  
[SetText](#)

## GetTextTrim

**Description** The GetTextTrim method returns the textual content of the current PBDOM\_CHARACTERDATA object with all surrounding whitespace characters removed.

**Syntax** `pbdom_chardata_name.GetTextTrim()`

Argument	Description
<code>pbdom_chardata_name</code>	The name of a PBDOM_CHARACTERDATA

**Return value** String.

DOM Object	Return Value
PBDOM_TEXT	<p>The text data contained within the PBDOM_TEXT object itself with surrounding whitespace characters removed.</p> <p>For example, suppose there is the following element:</p> <pre>&lt;abc&gt; MY TEXT &lt;/abc&gt;</pre> <p>If there is a PBDOM_TEXT object to represent the TEXT NODE “MY TEXT”, then calling GetTextTrim on the PBDOM_TEXT returns the string MY TEXT.</p>
PBDOM_CDATA	<p>The string data that is contained within the CDATA section itself with surrounding whitespace characters removed. For example, suppose there is the following CDATA:</p> <pre>&lt;![CDATA[ They're saying "x &lt; y" &amp; that "z &gt; y" so I guess that means that z &gt; x ]]&gt;</pre> <p>If there is a PBDOM_CDATA to represent the above CDATA section, then calling GetTextTrim on it returns the string:</p> <pre>They're saying " x &lt; y " &amp; that "z &gt; y" so I guess that means that z &gt; x</pre> <p>Note that the initial spaces before “They’re” and the trailing space after the last “x” are removed.</p>
PBDOM_COMMENT	<p>Suppose there is the following comment:</p> <pre>&lt;!-- This is a comment --&gt;</pre> <p>Calling GetTextTrim on this comment returns:</p> <pre>This is a comment</pre> <p>Note that the spaces between the individual words in the comment are preserved. Only the surrounding whitespace characters are removed.</p>

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Examples** This example demonstrates:

- 1 Using an External DTD.
- 2 Using a parameter entity.
- 3 Using a single line statement to obtain the children PBDOM\_OBJECTs of an element.
- 4 Obtaining the text of the three separate types of PBDOM\_CHARACTERDATA objects : PBDOM\_TEXT, PBDOM\_COMMENT, and PBDOM\_CDATA.
- 5 Obtaining the trimmed text of the same three separate types of PBDOM\_CHARACTERDATA objects.
- 6 The difference between the two types of text retrieved in 4 and 5.

The PowerScript code saves a string into an external file, then creates a PBDOM\_DOCUMENT pbdom\_doc based on the following DOM tree:

```
<!DOCTYPE abc SYSTEM "c:\external_entity.dtd">
<abc>
  <data>
    &text1;
    <!-- &text1;-->
    <![CDATA[&text1;]]>
  </data>
</abc>
```

*c:\external\_entity.dtd* is an external Document Type Definition file. Its contents are the external subset of the Document Type Definition. The first line declares a PARAMETER entity `param_entity_ref` that contains the following replacement text:

```
&#32; &#32; &#32; PARAMETER ENTITY REFERENCE&#9; &#9; &#9;
```

The next line declares a general entity `text1` that contains the following replacement text:

```
%param_entity_ref;
```

When the entity `text1` is used in an XML document, it is expanded to the contents of the PARAMETER entity `param_entity_ref`.

The PowerScript code then obtains the root element, uses it to obtain the data child element, and then obtains an array of the child element's own children. PBDOM collects all the PBDOM\_OBJECTs that are the children of data and stores them in the PBDOM\_OBJECT array `pbdom_obj_array`.

Next, the FOR loop iterates through all the items in `pbdom_obj_array` and stores each item in the PBDOM\_CHARACTERDATA array `pbdom_chardata`. This step is not required—the `pbdom_obj_array` can be used to manipulate the data element's children. It is done to demonstrate that you can cast each item into a PBDOM\_CHARACTERDATA object by assigning it into a PBDOM\_CHARACTERDATA array.

This is possible if and only if each PBDOM\_OBJECT is also derived from PBDOM\_CHARACTERDATA. If a PBDOM\_OBJECT is not derived from PBDOM\_CHARACTERDATA, the PowerBuilder VM throws an exception.

The next FOR loop iterates through all the items of the `pbdom_chardata` array and calls the `GetText` and `GetTextTrim` methods on each. Each of the returned strings from `GetText` and `GetTextTrim` is delimited by “[“ and “]” characters so that the complete text content displays clearly in the message boxes.

The first child of data is the PBDOM\_TEXT `&text1;`, which expands to the string in `param_entity_ref`. The entity references within this string are also expanded and the Tab and Space characters display when `GetText` is called. When `GetTextTrim` is called, PBDOM removes the beginning and trailing whitespace characters and the resulting string is simply `PARAMETER ENTITY REFERENCE`.

The second child of data is the PBDOM\_COMMENT `<!-- &text1;-->`, and the third child is the PBDOM\_CDATA `<![CDATA[&text1;]]>`. The string `&text1;` is not considered to be an entity reference by PBDOM because W3C DOM comments and CDATA sections cannot hold any entity references. Both `GetText` and `GetTextTrim` return the string `&text1;`. There are no leading or trailing spaces to remove.

```
PBDOM_CHARACTERDATA      pbdom_chardata[]
PBDOM_OBJECT             pbdom_obj_array[]
integer                  iFileNum1
long                     l = 0
string strExternalDTD = "<!ENTITY % param_entity_ref
~"&#32;&#32;&#32;PARAMETER ENTITY
REFERENCE&#9;&#9;&#9;~"><!ENTITY text1
~"%param_entity_ref;~">"
string strXML = "<!DOCTYPE abc SYSTEM
~"c:\external_entity.dtd~"><abc><data>&text1;<!--
&text1;--><![CDATA[&text1;]]></data></abc>"
```

```

TRY
    iFileNum1 = FileOpen("c:\external_entity.dtd", &
        StreamMode!, Write!, LockWrite!, Replace!)
    FileWrite(iFileNum1, strExternalDTD)
    FileClose(iFileNum1)

    pbdombuilder_new = Create PBDOM_Builder
    pbdom_doc = pbdombuilder_new.BuildFromString (strXML)

    pbdom_doc.GetRootElement(). &
        GetChildElement("data"). &
        GetContent(pbdom_obj_array)

    for l = 1 to UpperBound(pbdom_obj_array)
        pbdom_chardata[l] = pbdom_obj_array[l]
    next

    for l = 1 to UpperBound(pbdom_chardata)
        MessageBox (pbdom_chardata[l]. &
            GetObjectClassString() + " GetText()", &
            "[" + pbdom_chardata[l].GetText() + "]")
        MessageBox (pbdom_chardata[l]. &
            GetObjectClassString() + " GetTextTrim()" , &
            "[" + pbdom_chardata[l].GetTextTrim() + "]")
    next

    Destroy pbdombuilder_new

CATCH (PBDOM_Exception except)
    MessageBox ("Exception Occurred", except.Text)
END TRY

```

**Usage** If no textual value exists for the current PBDOM\_CHARACTERDATA, or if only whitespace characters exist, an empty string is returned.

**See also** [GetText](#)  
[GetTextNormalize](#)  
[SetText](#)

## HasChildren

**Description** This method returns true if this PBDOM\_CHARACTERDATA has at least one child PBDOM\_OBJECT. If this PBDOM\_CHARACTERDATA has no children, false is returned.

**Syntax** *pbdom\_chardata\_name*.HasChildren()

Argument	Description
<i>pbdom_chardata_name</i>	The name of a PBDOM_CHARACTERDATA.

**Return value** Boolean.

Value	Description
true	The current PBDOM_CHARACTERDATA has at least one child PBDOM_OBJECT
false	The current PBDOM_CHARACTERDATA has no child PBDOM_OBJECTs

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Usage** If the PBDOM\_CHARACTERDATA has at least one child PBDOM\_OBJECT, true is returned. False is returned if there are no children. Currently, false is always returned because no subclasses of PBDOM\_CHARACTERDATA contain child nodes.



## IsAncestorObjectOf

**Description** The IsAncestorObjectOf method determines whether the current PBDOM\_CHARACTERDATA is the ancestor of another PBDOM\_OBJECT.

**Syntax** *pbdom\_chardata\_name*.IsAncestorObjectOf(*pbdom\_object* *pbdom\_object\_ref*)

Argument	Description
<i>pbdom_chardata_name</i>	The name of a PBDOM_CHARACTERDATA
<i>pbdom_object_ref</i>	A PBDOM_OBJECT to check against

**Return value** Boolean. Returns true if the current PBDOM\_CHARACTERDATA is the ancestor of the referenced PBDOM\_OBJECT, and false otherwise.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Usage** Currently, false is always returned because no subclasses of PBDOM\_CHARACTERDATA contain child nodes. Therefore, they cannot be ancestors of a PBDOM\_OBJECT.

## SetParentObject

**Description** The SetParentObject method sets the referenced PBDOM\_OBJECT to be the parent of the current PBDOM\_CHARACTERDATA.

**Syntax** *pbdom\_chardata\_name*.SetParentObject(*pbdom\_object* *pbdom\_object\_ref*)

Argument	Description
<i>pbdom_chardata_name</i>	The name of a PBDOM_CHARACTERDATA
<i>pbdom_object_ref</i>	A PBDOM_OBJECT to be set as the parent of this PBDOM_CHARACTERDATA object

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA. This exception also occurs if the input PBDOM\_OBJECT is not a reference to an object derived from PBDOM\_OBJECT.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – If the current PBDOM\_CHARACTERDATA already has a parent.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT is of a class that does not have a proper parent-child relationship with the class of this PBDOM\_CHARACTERDATA.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT requires a user-defined name, and it has not been named.

Examples

This example creates a PBDOM\_DOCUMENT based on the following DOM tree:

```
<abc>
  <data>
    <child_1/>
    <child_2/>
    <child_3/>
  </data>
</abc>
```

The code creates three separate types of PBDOM\_CHARACTERDATA objects and stores them in the pbdom\_chardata array. It then obtains the root element, uses it to obtain the data child element, and then uses that to obtain the first child element, which it sets as the parent of the first item in the pbdom\_chardata array.

The text of the array item is set to Comment. You can set the string content of any PBDOM\_CHARACTERDATA object after you have set it as the child of a parent.

The same process is repeated for the text and CDATA objects:

```
PBDOM_Builder          pbdombuilder_new
pbdom_document         pbdom_doc
PBDOM_CHARACTERDATA   pbdom_chardata[]
PBDOM_ELEMENT         pbdom_elem
long                   = 0
string strXML =
"<abc><data><child_1/><child_2/><child_3/></data></abc>"
```

```
TRY
  pbdombuilder_new = Create PBDOM_Builder
  pbdom_doc = pbdombuilder_new.BuildFromString (strXML)

  pbdom_chardata[1] = Create PBDOM_COMMENT
  pbdom_chardata[2] = Create PBDOM_TEXT
  pbdom_chardata[3] = Create PBDOM_CDATA

  pbdom_elem = pbdom_doc.GetRootElement(). &
```

```

    GetChildElement("data").GetChildElement("child_1")
    pbdom_chardata[1].SetParentObject (pbdom_elem)
    pbdom_chardata[1].SetText ("Comment")

    pbdom_elem = pbdom_doc.GetRootElement(). &
        GetChildElement("data").GetChildElement("child_2")
    pbdom_chardata[2].SetParentObject (pbdom_elem)
    pbdom_chardata[2].SetText ("Text")

    pbdom_elem = pbdom_doc.GetRootElement(). &
        GetChildElement("data").GetChildElement("child_3")
    pbdom_chardata[3].SetParentObject (pbdom_elem)
    pbdom_chardata[3].SetText ("CDATA")

    pbdom_doc.SaveDocument ("c:\pbdom_doc_1.xml")

    Destroy pbdombuilder_new

    CATCH (PBDOM_Exception except)
        MessageBox ("Exception Occurred", except.Text)
    END TRY

```

When the PBDOM\_DOCUMENT is saved to a file, the output DOM tree looks like this:

```

<abc>
  <data>
    <child_1>
      <!--Comment-->
    </child_1>
    <child_2>
      Text
    </child_2>
    <child_3>
      <![CDATA[CDATA]]>
    </child_3>
  </data>
</abc>

```

#### Usage

The PBDOM\_OBJECT that you set to be the parent of the current PBDOM\_CHARACTERDATA must have a legal parent-child relationship. If it does not, an exception is thrown.

#### See also

GetParentObject

## SetText

**Description** The SetText method sets the input string to be the text content of the current PBDOM\_CHARACTERDATA object.

**Syntax** *pbdom\_chardata\_name*.SetText(string *strSet*)

<b>Argument</b>	<b>Description</b>
<i>pbdom_chardata_name</i>	The name of a PBDOM_CHARACTERDATA
<i>strSet</i>	The string you want set as the text of the PBDOM_CHARACTERDATA

**Return value** PBDOM\_CHARACTERDATA. The current PBDOM\_CHARACTERDATA object modified.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_CHARACTERDATA is not a reference to an object derived from PBDOM\_CHARACTERDATA.

**Usage** The SetText method sets the input string to be the text content of the current PBDOM\_CHARACTERDATA object.

**See also** GetText  
GetTextNormalize  
GetTextTrim

About this chapter

This chapter describes the PBDOM\_COMMENT class.

## PBDOM\_COMMENT

### Description

The PBDOM\_COMMENT class represents a DOM Comment Node within an XML document. The PBDOM\_COMMENT class is derived from the PBDOM\_CHARACTERDATA class and is intended to extend the PBDOM\_CHARACTERDATA class with a set of methods intended specifically for manipulating DOM comment nodes.

You can use comments to annotate an XML document with user-readable information.

In PBDOM, when a document is parsed, any comments found within the document persist as part of the resultant DOM tree in memory. A PBDOM\_COMMENT created at runtime also becomes part of the DOM tree. However, an XML comment does not usually form part of the content model of a document.

The presence or absence of comments has no bearing on a document's validity. There is no requirement that comments must be predeclared in a DTD.

### Methods

Some of the inherited methods from PBDOM\_OBJECT serve no meaningful objective, and only default or trivial functionalities result. These are described in the following table:

<b>Method</b>	<b>Always returns</b>
AddContent	current PBDOM_COMMENT
GetContent	false
GetName	a string "#comment"
HasChildren	false
InsertContent	current PBDOM_COMMENT
IsAncestorObjectOf	false
RemoveContent	false

Method	Always returns
SetContent	current PBDOM_COMMENT
SetName	false

PBDOM\_COMMENT has the following non-trivial methods:

- Append
- Clone
- Detach
- Equals
- GetObjectClass
- GetObjectClassString
- GetOwnerDocumentObject
- GetParentObject
- GetText
- GetTextNormalize
- GetTextTrim
- SetParentObject
- SetText

## Append

Description

The Append method is overloaded:

- Syntax 1 appends an input string to the text content that already exists within the current PBDOM\_COMMENT object.
- Syntax 2 appends the text data of a PBDOM\_CHARACTERDATA object to the text content that already exists within the current PBDOM\_COMMENT object.

Syntax

For this syntax	See
Append(string <i>strAppend</i> )	Append Syntax 1
Append(pbdom_characterdata <i>pbdom_characterdata_ref</i> )	Append Syntax 2

## Append Syntax 1

**Description** Appends an input string to the text content that already exists within the current PBDOM\_COMMENT object.

**Syntax** `pbdom_comment_name.Append(string strAppend)`

Argument	Description
<code>pbdom_comment_name</code>	The name of a PBDOM_COMMENT
<code>strAppend</code>	The string you want to append to the existing text of the current PBDOM_COMMENT object

**Return value** PBDOM\_CHARACTERDATA. The current PBDOM\_COMMENT modified and returned as a PBDOM\_CHARACTERDATA object.

## Append Syntax 2

**Description** Appends the text data of a PBDOM\_CHARACTERDATA object to the text content that exists within the current PBDOM\_COMMENT object.

**Syntax** `pbdom_comment_name.Append(pbdom_characterdata pbdom_characterdata_ref)`

Argument	Description
<code>pbdom_comment_name</code>	The name of a PBDOM_COMMENT
<code>pbdom_characterdata_ref</code>	The referenced PBDOM_CHARACTERDATA object whose text data is to be appended to the existing text of the current PBDOM_COMMENT object

**Return value** PBDOM\_CHARACTERDATA. The current PBDOM\_COMMENT modified and returned as a PBDOM\_CHARACTERDATA object.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_CHARACTERDATA is not a reference to a PBDOM\_CHARACTERDATA-derived object.

**Usage** Note that JDOM does not define an Append method for its COMMENT class. Because PBDOM implements its Append method in the base PBDOM\_CHARACTERDATA class, a PBDOM\_TEXT object, a PBDOM\_CDATA object, and a PBDOM\_COMMENT object can append their internal text data to each other because they are all PBDOM\_CHARACTERDATA-derived objects.

## Clone

Description Creates and returns a clone of the current PBDOM\_COMMENT.

Syntax *pbdom\_comment\_name.Clone*(boolean *bDeep*)

Argument	Description
<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT
<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are true for a deep clone and false for a shallow clone

Return value PBDOM\_OBJECT.

Examples This example creates an XML document that, when serialized, appears as follows :

```
<!DOCTYPE root
[
<!ELEMENT root (level_1)*>
<!ELEMENT level_1 (level_2)*>
<!ELEMENT level_2 (#PCDATA)*>
]>
<root>
  <level_1>
    <!--Element at level : 1-->
    <level_2>
      <!--Element at level : 2-->
    </level_2>
  </level_1>
</root>
```

The definition of the DTD shows that the document is required to have the following composition:

- The document contains a root element with the name root.
- The root element contains zero or more occurrences of level\_1 elements.
- A level\_1 element contains zero or more level\_2 elements.
- A level\_2 element is expected to contain text.



The following PowerScript code supplies annotations within the document by including comments to mark level\_1 and level\_2 elements. The sample code creates a PBDOM\_DOCUMENT from an XML string that contains a DTD and a minimal root element. Then, it creates a comment that serves as a template. The template comment is then cloned, and instance-specific text is added for each element:

```

PBDOM_COMMENT pbdom_comm
PBDOM_COMMENT pbdom_comm_clone
PBDOM_ELEMENT pbdom_elem
PBDOM_DOCUMENT pbdom_doc
PBDOM_BUILDER pbdom_buildr
string strXML = "<!DOCTYPE root [<!ELEMENT root
(level_1)*><!ELEMENT level_1 (level_2)*><!ELEMENT
level_2 (#PCDATA)>]><root/>"

try
    // Create a PBDOM_DOCUMENT from the XML string that
    // contains a DTD and a minimal root element.
    pbdom_buildr = Create PBDOM_BUILDER
    pbdom_doc = pbdom_buildr.BuildFromString(strXML)

    // Create a template comment that can be reused.
    pbdom_comm = Create PBDOM_COMMENT
    pbdom_comm.SetText ("Element at level : ")

    // Create a level_1 element.
    pbdom_elem = Create PBDOM_ELEMENT
    pbdom_elem.SetName("level_1")

    // Clone the template comment, append instance-
    // specific text, and add it to the level_1 element.
    pbdom_comm_clone = pbdom_comm.Clone(true)
    pbdom_elem.AddContent(pbdom_comm_clone.Append("1"))

    // Add a level_1 element into the root element
    // as stipulated by the DTD.
    pbdom_doc.GetRootElement().AddContent(pbdom_elem)

    // Create a level_2 element.
    pbdom_elem = Create PBDOM_ELEMENT
    pbdom_elem.SetName("level_2")

    // Clone the template comment, append instance-
    // specific text, and add it to the level_2 element.
    pbdom_comm_clone = pbdom_comm.Clone(true)

```

```

pbdom_elem.AddContent (pbdom_comm_clone.Append ("2"))

// Add a level_2 element into the level_1 element
// as stipulated by the DTD.
pbdom_doc.GetRootElement().GetChildElement &
    ("level_1").AddContent (pbdom_elem)

// Finally, serialize the document.
pbdom_doc.SaveDocument ("sample.xml")

catch (PBDOM_EXCEPTION pbdom_e)
    MessageBox ("PBDOM_EXCEPTION", pbdom_e.GetMessage())
end try

```

Usage

The Clone method creates a new PBDOM\_COMMENT object that is a duplicate of, and a separate object from, the original. Whether true or false is supplied, the clone is always identical to its original, because a PBDOM\_COMMENT does not contain a subtree of child PBDOM\_OBJECTs.

A PBDOM\_COMMENT clone has no parent. However, the clone resides in the same PBDOM\_DOCUMENT as its original, and if the original is standalone, the clone is standalone.

## Detach

Description

Detaches a PBDOM\_COMMENT from its parent PBDOM\_OBJECT.

Syntax

*pbdom\_comment\_name*.Detach()

Argument	Description
<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT

Return value

PBDOM\_OBJECT.

The current PBDOM\_COMMENT is detached from its parent.

Usage

If the current PBDOM\_COMMENT object has no parent, no modifications occur.

## Equals

**Description** Tests for the equality of the current PBDOM\_COMMENT and a referenced PBDOM\_OBJECT.

**Syntax** `pbdom_comment_name.Equals(pbdom_object pbdom_object_ref)`

Argument	Description
<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT.
<i>pbdom_object_ref</i>	A PBDOM_OBJECT to test for equality with the current PBDOM_COMMENT

**Return value** Boolean. Returns true if the current PBDOM\_COMMENT is equivalent to the input PBDOM\_OBJECT, and false otherwise.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the referenced PBDOM\_OBJECT is not a reference to an object derived from a PBDOM\_OBJECT object.

**Usage** True is returned only if the referenced PBDOM\_OBJECT is also a derived PBDOM\_COMMENT object and refers to the same DOM object as the current PBDOM\_COMMENT. Two separately created PBDOM\_COMMENTS, for example, can contain exactly the same text but are not equal.

## GetObjectClass

**Description** Returns a long integer code that indicates the class of the current PBDOM\_OBJECT.

**Syntax** `pbdom_object_name.GetObjectClass()`

Argument	Description
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

**Return value** Long. GetObjectClass returns a long integer code that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_COMMENT, the returned value is 9.

**See also** GetObjectClassString

## GetObjectClassString

Description Returns a string form of the class of the PBDOM\_OBJECT.

Syntax *pbdom\_object\_name*.GetObjectClassString()

Argument	Description
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

Return value String. GetObjectClassString returns a string that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_COMMENT, the returned string is “pbdom\_comment”.

See also GetObjectClass

## GetOwnerDocumentObject

Description Returns the owning PBDOM\_DOCUMENT of the current PBDOM\_COMMENT.

Syntax *pbdom\_comment\_name*.GetOwnerDocumentObject()

Argument	Description
<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT

Return value PBDOM\_OBJECT.

Usage If there is no owning PBDOM\_DOCUMENT, null is returned.

## GetParentObject

Description Returns the parent PBDOM\_OBJECT of the current PBDOM\_COMMENT.

Syntax *pbdom\_comment\_name*.GetParentObject()

Argument	Description
<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT

Return value PBDOM\_OBJECT.

Usage The GetParentObject method returns the parent PBDOM\_OBJECT of the current PBDOM\_COMMENT. If the PBDOM\_COMMENT has no parent, null is returned.

See also SetParentObject

## GetText

Description	Allows you to obtain the text data that is contained within the current PBDOM_COMMENT object.				
Syntax	<i>pbdom_comment_name</i> .GetText() <table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_comment_name</i></td> <td>The name of a PBDOM_COMMENT</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT
Argument	Description				
<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT				
Return value	String. The textual content of the current PBDOM_COMMENT object.				
Examples	If you have the comment <!--A COMMENT-->, the GetText method returns the string A COMMENT.				
See also	GetTextNormalize GetTextTrim SetText				

## GetTextNormalize

Description	Allows you to obtain the text data that is contained within the current PBDOM_COMMENT object, with all surrounding whitespace characters removed and internal whitespace characters normalized to a single space.				
Syntax	<i>pbdom_comment_name</i> .GetTextNormalize() <table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_comment_name</i></td> <td>The name of a PBDOM_COMMENT</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT
Argument	Description				
<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT				
Return value	String.				
Examples	If you have the comment <!-- A COMMENT -->, which has three spaces before and after the text and between the two words, the GetTextNormalize method returns the string A COMMENT, which has a single space between the words.				
Usage	This method allows the caller to obtain the text data that is contained within the current PBDOM_COMMENT with all surrounding whitespace characters removed and internal whitespace characters normalized to single spaces. If no textual value exists for the current PBDOM_COMMENT, or if only whitespace characters exist, an empty string is returned.				
See also	GetText GetTextTrim SetText				

## GetTextTrim

**Description** Returns the textual content of the current PBDOM\_COMMENT object with all surrounding whitespace characters removed.

**Syntax** `pbdom_comment_name.GetTextTrim()`

<b>Argument</b>	<b>Description</b>
<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT

**Return value** String.

**Examples** If you have the comment `<!-- A COMMENT -->`, which has three spaces before and after the text and between the two words, the `GetTextTrim` method returns the string `A COMMENT`. The whitespace characters between the words are preserved.

**Usage** This method allows the caller to obtain the text data that is contained within the current PBDOM\_COMMENT with all surrounding whitespace characters removed. Internal whitespace characters are preserved. If no textual value exists for the current PBDOM\_COMMENT, or if only whitespace characters exist, an empty string is returned.

**See also** `GetText`  
`GetTextNormalize`  
`SetText`

## SetParentObject

**Description** Sets the referenced PBDOM\_OBJECT to be the parent of the current PBDOM\_COMMENT.

**Syntax** `pbdom_comment_name.SetParentObject(pbdom_object pbdom_object_ref)`

<b>Argument</b>	<b>Description</b>
<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT
<i>pbdom_object_ref</i>	A PBDOM_OBJECT to be set as the parent of the current PBDOM_COMMENT

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not a reference to an object derived from PBDOM\_OBJECT.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – If the current PBDOM\_COMMENT already has a parent.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT is of a class that does not have a proper parent-child relationship with the PBDOM\_COMMENT class.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT requires a user-defined name, and it has not been named.

## Usage

This method sets the input PBDOM\_OBJECT as the parent of this PBDOM\_COMMENT. The caller is responsible for ensuring that the current PBDOM\_COMMENT and the input PBDOM\_OBJECT can have a legal parent-child relationship. Currently, only a PBDOM\_ELEMENT and a PBDOM\_DOCUMENT can be set as the parent of a PBDOM\_COMMENT.

The PBDOM\_COMMENT SetParentObject method differs from the JDOM Comment setParent method in two ways:

- JDOM defines a setParent method for several specific classes, including Element, Comment, and CDATA. PBDOM implements the SetParentObject method in the base PBDOM\_OBJECT class to allow for polymorphism.
- The JDOM Comment's setParent method takes only an Element class object as a parameter:

```
COMMENT::setParent(Element parent)
```

To set a Document as the parent owner of a Comment using JDOM, you use the setDocument method:

```
COMMENT::setDocument(Document document)
```

In PBDOM, SetParentObject takes a reference to a PBDOM\_OBJECT, so that both a PBDOM\_ELEMENT and a PBDOM\_DOCUMENT can be set as a parent.

## See also

GetOwnerDocumentObject  
GetParentObject

**SetText**

## Description

Sets the input string to be the text content of the current PBDOM\_COMMENT object.

## Syntax

```
pbdom_comment_name.SetText(string strSet)
```

<b>Argument</b>	<b>Description</b>
<i>pbdom_comment_name</i>	The name of a PBDOM_COMMENT
<i>strSet</i>	The string you want set as the text of the PBDOM_COMMENT

Return value

String.

See also

GetText  
GetTextNormalize  
GetTextTrim



About this chapter

This chapter describes the PBDOM\_DOCTYPE class.

## PBDOM\_DOCTYPE

Description

The PBDOM\_DOCTYPE class represents the Document Type Declaration Object of an XML DOM Document. The PBDOM\_DOCTYPE class provides access to the name of the root element that is constrained within the DOCTYPE as well as the internal subset, system, and public IDs.

Methods

Some of the inherited methods from PBDOM\_OBJECT serve no meaningful objective and only default or trivial functionalities result. These are described in the following table:

<b>Method</b>	<b>Always returns</b>
AddContent	The current PBDOM_DOCTYPE
GetContent	false
GetText	Empty string
GetTextNormalize	Empty string
GetTextTrim	Empty string
HasChildren	false
InsertContent	The current PBDOM_DOCTYPE
IsAncestorObjectOf	false
RemoveContent	false
SetContent	The current PBDOM_DOCTYPE

PBDOM\_DOCTYPE has the following non-trivial methods:

Clone	GetObjectClassString	SetInternalSubset
Detach	GetOwnerDocumentObject	SetName
Equals	GetParentObject	SetParentObject
GetInternalSubset	GetPublicID	SetPublicID
GetName	GetSystemID	SetSystemID
GetObjectClass	SetDocument	

## Clone

**Description** Creates and returns a clone of the current PBDOM\_DOCTYPE.

**Syntax** *pbdom\_doctype\_name*.Clone(boolean *bDeep*)

Argument	Description
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object.
<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are TRUE for a deep clone and FALSE for a shallow clone. This argument is currently ignored.

**Return value** PBDOM\_OBJECT. A deep clone of the current PBDOM\_DOCTYPE housed in a PBDOM\_OBJECT.

**Usage** A PBDOM\_DOCTYPE clone (whether shallow or deep) is always an exact copy of its original. This is because a PBDOM\_DOCTYPE does not contain any subtree of child PBDOM\_OBJECTs.

A PBDOM\_DOCTYPE clone has no parent. However, the clone resides in the same PBDOM\_DOCUMENT as its original. If the original PBDOM\_DOCTYPE is standalone, the clone is standalone.

## Detach

**Description** Detaches a PBDOM\_DOCTYPE object from its parent PBDOM\_DOCUMENT object. The detached PBDOM\_DOCTYPE object is still part of the PBDOM\_DOCUMENT object in which it resided before the Detach method was invoked, but it no longer has a parent PBDOM\_DOCUMENT object.

**Syntax** *pbdom\_doctype\_name*.Detach()

Argument	Description
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object

**Return value** PBDOM\_OBJECT. The PBDOM\_DOCTYPE object modified and returned as a PBDOM\_OBJECT object.

## Equals

**Description** Tests for the equality of the current PBDOM\_DOCTYPE and a referenced PBDOM\_OBJECT.

**Syntax** `pbdom_doctype_name.Equals(pbdom_object_ref)`

Argument	Description
<code>pbdom_doctype_name</code>	The name of a PBDOM_DOCTYPE object
<code>pbdom_object_ref</code>	A PBDOM_OBJECT to test for equality with the current PBDOM_DOCTYPE

**Return value** Boolean. Returns true if the current PBDOM\_DOCTYPE is equivalent to the input PBDOM\_OBJECT, and false otherwise.

**Usage** True is returned only if the referenced PBDOM\_OBJECT is also a PBDOM\_DOCTYPE and refers to the same DOM Doctype object as the current PBDOM\_DOCTYPE.

## GetInternalSubset

**Description** Returns the internal subset data of the DOCTYPE.

**Syntax** `pbdom_doctype_name.GetInternalSubset()`

Argument	Description
<code>pbdom_doctype_name</code>	The name of a PBDOM_DOCTYPE object

**Return value** String.

**See also** SetInternalSubset

## GetName

**Description** Allows you to obtain the name of the root element that is being constrained within the current PBDOM\_DOCTYPE.

**Syntax** `pbdom_doctype_name.GetName()`

Argument	Description
<code>pbdom_doctype_name</code>	The name of a PBDOM_DOCTYPE object

**Return value** String.

Examples                      If you have the following DOCTYPE declaration, the GetName method returns abc.

```
<!DOCTYPE abc [<!-- internal subset -->
<!ELEMENT abc (#PCDATA)> <!ELEMENT data (#PCDATA)>
<!ELEMENT inner_data (#PCDATA)>]>
```

## GetObjectClass

Description                      Returns a long integer code that indicates the class of the current PBDOM\_OBJECT.

Syntax                              *pbdom\_object\_name*.GetObjectClass()

Argument	Description
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

Return value                      Long. A long integer code that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_DOCTYPE, the returned value is 4.

## GetObjectClassString

Description                      Returns a string form of the class of the PBDOM\_OBJECT.

Syntax                              *pbdom\_object\_name*.GetObjectClassString()

Argument	Description
<i>pbdom_object_name</i>	The name of your PBDOM_OBJECT

Return value                      String. A string that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_DOCTYPE, the returned string is “pbdom\_doctype”.

## GetOwnerDocumentObject

Description	Returns the owning PBDOM_DOCUMENT of the current PBDOM_DOCTYPE.				
Syntax	<i>pbdom_doctype_name</i> .GetOwnerDocumentObject()				
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_doctype_name</i></td> <td>The name of a PBDOM_DOCTYPE object</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object
Argument	Description				
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object				
Return value	PBDOM_OBJECT.				
Usage	If there is no owning PBDOM_DOCUMENT, null is returned.				

## GetParentObject

Description	Returns the parent PBDOM_OBJECT of the current PBDOM_DOCTYPE.				
Syntax	<i>pbdom_doctype_name</i> .GetParentObject()				
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_doctype_name</i></td> <td>The name of a PBDOM_DOCTYPE object</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object
Argument	Description				
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object				
Return value	PBDOM_OBJECT.				
Usage	The parent is also a PBDOM_DOCUMENT object. If the PBDOM_OBJECT has no parent, null is returned.				

## GetPublicID

Description	Retrieves the public ID of an externally reference DTD declared in the DOCTYPE.				
Syntax	<i>pbdom_doctype_name</i> .GetPublicID()				
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_doctype_name</i></td> <td>The name of a PBDOM_DOCTYPE object</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object
Argument	Description				
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object				
Return value	String. If no public ID is referenced, an empty string is returned.				
Examples	Suppose you have the following DTD declaration:				

```
<!DOCTYPE Books PUBLIC "-//MyCompany//DTD//EN"
"http://mycompany.com/dtd/mydoctype.dtd">
```

The following PowerShell code displays the public and system IDs in message boxes:

```

pbdom_doctype pbdom_doctype_1
pbdom_document pbdom_doc

pbdom_doctype_1 = pbdom_doc.GetDocType()
MessageBox ("DocType Public ID", &
    pbdom_doctype_1.GetPublicID())
MessageBox ("DocType System ID", &
    pbdom_doctype_1.GetSystemID())

```

The returned strings from the calls to `GetPublicID` and `GetSystemID` are:

```

"- //MyCompany//DTD//EN"
"http://mycompany.com/dtd/mydoctype.dtd"

```

See also

[GetSystemID](#)  
[SetPublicID](#)  
[SetSystemID](#)

## GetSystemID

**Description** Retrieves the system ID of an externally referenced DTD declared in the DOCTYPE.

**Syntax** `pbdom_doctype_name.GetSystemID()`

Argument	Description
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object

**Return value** String. If no system ID is referenced, an empty string is returned.

**Examples** See [GetPublicID](#).

See also

[GetPublicID](#)  
[SetPublicID](#)  
[SetSystemID](#)

## SetDocument

**Description** Sets the owning PBDOM\_DOCUMENT of the current PBDOM\_DOCTYPE.

**Syntax** `pbdom_doctype_name.SetDocument(pbdom_document  
pbdom_document_ref)`

Argument	Description
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object
<i>pbdom_document_ref</i>	A PBDOM_DOCUMENT object to be set as the owner document of this PBDOM_DOCTYPE object

**Return value** PBDOM\_DOCTYPE. The current PBDOM\_DOCTYPE modified to be the DOCTYPE of the referenced PBDOM\_DOCUMENT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – if the input PBDOM\_DOCUMENT object is invalid for use in any way.  
  
EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – if this current PBDOM\_DOCTYPE already has a parent PBDOM\_OBJECT. In this case, this PBDOM\_DOCTYPE is already the DOCTYPE of some document.

**Usage** A DOM DOCTYPE object can have no owner document, or it can have an owner document but no parent node. A DOCTYPE that has an owner document as well as a parent node is the actual DOCTYPE of the owner document.

**See also** SetParentObject

## SetInternalSubset

**Description** Sets the data for the internal subset of the PBDOM\_DOCTYPE.

**Syntax** `pbdom_doctype_name.SetInternalSubset()`

Argument	Description
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object

**Return value** PBDOM\_DOCTYPE. The current PBDOM\_DOCTYPE with the new internal subset.

**Examples** Suppose you have the following DTD declaration:

```
<!DOCTYPE abc [<!ELEMENT abc (#PCDATA)> <!ELEMENT data
(#PCDATA)> <!ELEMENT inner_data (#PCDATA)>]>
```

The following code displays the internal subset in a message box:

```
string strInternalSubset
pbdom_document pbdom_doc

strInternalSubset = pbdom_doc.GetDocType().GetInternalSubset()
strInternalSubset += "<!ELEMENT another_data(#PCDATA)>"
pbdom_doc.GetDocType().SetInternalSubset (strInternalSubset)
MessageBox ("Get Internal Subset", &
    pbdom_doc.GetDocType().GetInternalSubset())
```

The returned string from the call to `GetInternalSubset` is:

```
"<!-- internal subset --> <!ELEMENT abc (#PCDATA)>
<!ELEMENT data (#PCDATA)> <!ELEMENT inner_data
(#PCDATA)> <!ELEMENT another_data (#PCDATA)>"
```

The new `ELEMENT` declaration for “another\_data” is included in the final internal subset.

See also [GetInternalSubset](#)

## SetName

**Description** The `SetName` method sets the name of the root element that is declared by this `PBDOM_DOCTYPE`.

**Syntax** `pbdom_doctype_name.SetName(string strName)`

Argument	Description
<code>pbdom_doctype_name</code>	The name of a <code>PBDOM_DOCTYPE</code> object
<code>strName</code>	The new name you want to set for the root element that is declared by the current <code>PBDOM_DOCTYPE</code>

**Return value** Boolean. Returns true if the name of the root element was changed and false otherwise.



## SetParentObject

**Description** The SetParentObject method sets the referenced PBDOM\_OBJECT to be the parent of the current PBDOM\_OBJECT and so sets the DOCTYPE represented by this PBDOM\_DOCTYPE to be the DOCTYPE of the referenced PBDOM\_DOCUMENT.

**Syntax** *pbdom\_doctype\_name*.SetParentObject(*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object
<i>pbdom_object_ref</i>	A PBDOM_OBJECT to be set as the parent of the current PBDOM_DOCTYPE

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – If this PBDOM\_DOCTYPE already has a parent.

EXCEPTION\_MULTIPLE\_DOCTYPE – If the input PBDOM\_OBJECT is a PBDOM\_DOCUMENT object and already has a doctype.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT is not a PBDOM\_DOCUMENT.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not associated with a derived PBDOM\_OBJECT.

**Usage** This method sets the input PBDOM\_OBJECT as the parent of the current PBDOM\_OBJECT. The input PBDOM\_OBJECT must be a PBDOM\_DOCUMENT. If it is not, an exception is thrown.

In PBDOM, calling SetParentObject is equivalent to setting the input PBDOM\_DOCUMENT as the owner document and parent node of the current PBDOM\_DOCTYPE. This has the effect of setting the DOCTYPE in PBDOM\_DOCTYPE as the DOCTYPE of the document.

A DOM DOCTYPE object can have no owner document, or it can have an owner document but no parent node. A DOCTYPE that has an owner document as well as a parent node is the actual DOCTYPE of the owner document.

This method is exactly the same as the SetDocument method.

**See also** SetDocument

## SetPublicID

Description Sets the public ID of an externally referenced DTD.

Syntax *pbdom\_doctype\_name*.SetPublicID(string *strPublicID*)

Argument	Description
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object
<i>strPublicID</i>	A string that specifies the new public ID

Return value PBDOM\_DOCTYPE.

Examples Suppose you have the following DTD declaration:

```
<!DOCTYPE abc [<!ELEMENT abc (#PCDATA)> <!ELEMENT data (#PCDATA)> <!ELEMENT inner_data (#PCDATA)>]>
```

The following PowerScript sets the public ID, and then gets it and displays it in a message box:

```
PBDOM_DOCUMENT pbdom_doc

pbdom_doc.GetDocType().SetPublicID &
  ("-//MyCompany//DTD//EN")
MessageBox ("Get Public ID", &
  pbdom_doc.GetDocType().GetPublicID())
```

The returned string from the GetPublicID call is:

```
"-//MyCompany//DTD//EN"
```

The final DOCTYPE definition in the document is:

```
<!DOCTYPE abc PUBLIC "-//MyCompany//DTD//EN"
[<!ELEMENT abc (#PCDATA)> <!ELEMENT data (#PCDATA)>
<!ELEMENT inner_data (#PCDATA)>]>
```

---

### About Public ID

The PUBLIC ID is usually accompanied by a SYSTEM ID, so the DOCTYPE declaration in this example (with a PUBLIC ID but no SYSTEM ID) might be considered invalid by some parsers.

---

See also  
GetPublicID  
GetSystemID  
SetSystemID

## SetSystemID

**Description** Sets the system ID of an externally referenced DTD.

**Syntax** *pbdom\_doctype\_name*.SetSystemID(*strSystemID*)

Argument	Description
<i>pbdom_doctype_name</i>	The name of a PBDOM_DOCTYPE object
<i>strSystemID</i>	A string that specifies the new system ID

**Return value** PBDOM\_DOCTYPE.

**Examples** Suppose you have the following DTD declaration:

```
<!DOCTYPE abc [<!ELEMENT abc (#PCDATA)> <!ELEMENT data
(#PCDATA)> <!ELEMENT inner_data (#PCDATA)>]>
```

The following PowerScript sets the system ID and then gets it and returns it in a message box:

```
PBDOM_DOCUMENT pbdom_doc
pbdom_doc.GetDocType().SetSystemID &
("http://www.sybase.com/dtd/datadef.dtd")
MessageBox ("Get System ID", &
pbdom_doc.GetDocType().GetSystemID())
```

The returned string from the GetSystemID call is:

```
"http://www.sybase.com/dtd/datadef.dtd"
```

The final DOCTYPE definition in the document is:

```
<!DOCTYPE abc SYSTEM
"http://www.sybase.com/dtd/datadef.dtd" [<!ELEMENT abc
(#PCDATA)> <!ELEMENT data (#PCDATA)> <!ELEMENT
inner_data (#PCDATA)>]>
```

**See also** GetPublicID  
GetSystemID  
SetPublicID



About this chapter

This chapter describes the PBDOM\_DOCUMENT class.

## PBDOM\_DOCUMENT

Description

The PBDOM\_DOCUMENT class defines behavior for an XML DOM document. Methods allow access to the root element, processing instructions, and other document-level information.

The PBDOM\_DOCUMENT class inherits from a PBDOM\_OBJECT and so provides specialized implementations for most of the PBDOM\_OBJECT class methods.

Methods

Some of the inherited methods from PBDOM\_OBJECT serve no meaningful objective and only default or trivial functionalities result. These are described in the following table:

<b>Method</b>	<b>Always returns</b>
Detach	The current PBDOM_DOCUMENT
GetName	The string "#document"
GetOwnerDocumentObject	null
GetParentObject	null
GetText	An empty string
GetTextNormalize	An empty string
GetTextTrim	An empty string
SetName	false
SetParentObject	The current PBDOM_DOCUMENT

PBDOM\_DOCUMENT has the following non-trivial methods:

AddContent	HasRootElement
Clone	InsertContent
DetachRootElement	IsAncestorObjectOf
Equals	NewDocument
GetContent	RemoveContent
GetDocType	SaveDocument
GetElementsByTagName	SaveDocumentIntoString
GetObjectClass	SetContent
GetObjectClassString	SetDocType
GetRootElement	SetRootElement
HasChildren	

## AddContent

Description

Allows you to add a new PBDOM\_OBJECT into the current PBDOM\_DOCUMENT object.

Syntax

*pbdom\_document\_name*.AddContent(*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to add

Return value

PBDOM\_OBJECT. The return value is the newly modified PBDOM\_DOCUMENT object returned as a PBDOM\_OBJECT.

Throws

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – The input PBDOM\_OBJECT is nameable, but it currently has no name.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – The input PBDOM\_OBJECT object is not associated with a derived PBDOM\_OBJECT class object.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – Adding the input PBDOM\_OBJECT is inappropriate. See description section below on the valid PBDOM\_OBJECTs that can be added to a PBDOM\_DOCUMENT object.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – If the PBDOM\_OBJECT to be added already has a parent PBDOM\_OBJECT.

EXCEPTION\_MULTIPLE\_ROOT\_ELEMENT – If a PBDOM\_ELEMENT is to be added and this document already has a root element.

EXCEPTION\_MULTIPLE\_DOCTYPE – If a PBDOM\_DOCTYPE is to be added and this document already has a DOCTYPE.

### Examples

The document `pbdom_doc1` is created with three elements: `pbdom_elem_1`, `pbdom_elem_2`, and `pbdom_elem_3`. `pbdom_elem_2` and `pbdom_elem_3` are set as children of `pbdom_element_1`.

`pbdom_doc1.GetRootElement().Detach()` detaches the root element from `pbdom_doc1`. `pbdom_elem_1` is added as a child of `pbdom_doc1` with `pbdom_doc1.AddContent(pbdom_elem_1)`.

```

TRY
    PBDOM_ELEMENT pbdom_elem_1
    PBDOM_ELEMENT pbdom_elem_2
    PBDOM_ELEMENT pbdom_elem_3
    PBDOM_DOCUMENT pbdom_doc1

    pbdom_doc1 = Create PBDOM_DOCUMENT
    pbdom_elem_1 = Create PBDOM_ELEMENT
    pbdom_elem_2 = Create PBDOM_ELEMENT
    pbdom_elem_3 = Create PBDOM_ELEMENT

    pbdom_elem_1.SetName("pbdom_elem_1")
    pbdom_elem_2.SetName("pbdom_elem_2")
    pbdom_elem_3.SetName("pbdom_elem_3")

    pbdom_elem_1.AddContent(pbdom_elem_2)
    pbdom_elem_1.AddContent(pbdom_elem_3)

    pbdom_doc1.NewDocument("", "", "Root_Element", &
        "", "")
    pbdom_doc1.GetRootElement().Detach()
    pbdom_doc1.AddContent(pbdom_elem_1)
CATCH (pbdom_exception ex)
    MessageBox("Exception", ex.getMessage())
END TRY

```

The original root element `<Root_Element>` has been detached and replaced by `<pbdom_elem_1>`. The document is transformed to:

```

<!DOCTYPE Root_Element>
<pbdom_elem_1>
  <pbdom_elem_2/>
  <pbdom_elem_3/>
</pbdom_elem_1>

```

If the following root element detachment statement is omitted, an exception is thrown:

`pbdom_doc1.GetRootElement().Detach()`

Usage

The new PBDOM\_OBJECT becomes a child PBDOM\_OBJECT of the current PBDOM\_DOCUMENT. The following table lists the PBDOM\_OBJECTs that can be added to a PBDOM\_DOCUMENT object and the restrictions for their addition.

PBDOM_OBJECT	Restrictions
PBDOM_ELEMENT	<p>Allowed to be added only if this document currently does not contain any root element. Otherwise the exception EXCEPTION_MULTIPLE_ROOT_ELEMENT is thrown.</p> <p>The PBDOM_ELEMENT to be added must not already have a parent PBDOM_OBJECT. If it does, the exception EXCEPTION_PBDOM_OBJECT_ALREADY_HAS_PARENT is thrown.</p>
PBDOM_COMMENT	<p>Any number of PBDOM_COMMENT objects can be added to a document.</p> <p>The only restriction is that the PBDOM_COMMENT must not already have a parent. If so, the exception EXCEPTION_PBDOM_OBJECT_ALREADY_HAS_PARENT is thrown.</p>
PBDOM_PROCESSINGINSTRUCTION	<p>Any number of PBDOM_PROCESSINGINSTRUCTION objects can be added to a document.</p> <p>The only restriction is that the PBDOM_PROCESSINGINSTRUCTION must not already have a parent. If so, the exception EXCEPTION_PBDOM_OBJECT_ALREADY_HAS_PARENT is thrown.</p>
PBDOM_DOCTYPE	<p>Allowed to be added only if this document currently does not contain any DOCTYPE node. Otherwise the exception EXCEPTION_MULTIPLE_DOCTYPE is thrown.</p> <p>The PBDOM_DOCTYPE to be added must not already have a parent PBDOM_OBJECT. If it does, the exception EXCEPTION_PBDOM_OBJECT_ALREADY_HAS_PARENT is thrown.</p>

See also

GetContent, InsertContent, RemoveContent, SetContent



## Clone

**Description** Creates a clone of the current PBDOM\_DOCUMENT object.

**Syntax** `pbdom_document_name.Clone(boolean bDeep)`

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object
<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are true for a deep clone and false for a shallow clone.

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – The internal implementation of the PBDOM\_DOCUMENT object is null. The occurrence of this exception is rare but can happen if severe memory corruption occurs.

**Usage** If you specify a deep clone, the Clone method creates a deep clone of the current PBDOM\_DOCUMENT object as a PBDOM\_OBJECT. The method recursively clones the subtree under the PBDOM\_DOCUMENT object, where the subtree consists of all legal children of the PBDOM\_DOCUMENT object.

If a shallow clone is requested, this method clones only the PBDOM\_DOCUMENT object and returns a completely empty PBDOM\_DOCUMENT object as a PBDOM\_OBJECT.

## DetachRootElement

**Description** Detaches the root element of this document and returns it.

**Syntax** `pbdom_document_name.DetachRootElement()`

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object

**Return value** PBDOM\_ELEMENT.

**Throws** EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

**See also** GetRootElement  
HasRootElement  
SetRootElement

## Equals

**Description** Tests for the equality of the current PBDOM\_DOCUMENT object and a referenced PBDOM\_OBJECT.

**Syntax** *pbdom\_document\_name*.Equals(*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_OBJECT
<i>pbdom_object_ref</i>	A PBDOM_OBJECT to test for equality with the current PBDOM_OBJECT

**Return value** Boolean. Returns true if the current PBDOM\_DOCUMENT object is equivalent to the input PBDOM\_OBJECT, and false otherwise.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – The input PBDOM\_OBJECT is not associated with a derived PBDOM\_OBJECT class object.

EXCEPTION\_INVALID\_ARGUMENT – The input PBDOM\_OBJECT is invalid. This can happen if the object has not been initialized properly or is a null object reference.

**Usage** True is returned only if the referenced PBDOM\_OBJECT is also a PBDOM\_DOCUMENT object and refers to the same DOM document as the current PBDOM\_DOCUMENT object.

## GetContent

**Description** Returns all child content of the current PBDOM\_DOCUMENT object.

**Syntax** *pbdom\_document\_name*.GetContent(*ref pbdom\_object pbdom\_object\_array[ ]*)

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object
<i>pbdom_object_array</i>	The referenced name of an array of PBDOM_OBJECTs that receives PBDOM_OBJECTs

**Return value** Boolean. Returns true for success and false for failure.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object is not associated with a derived PBDOM\_OBJECT class object.

**Examples** Assume a PBDOM\_DOCUMENT object called *pbdom\_doc* contains the following XML document.

```

<Root>
  <Element_1>
    <Element_1_1/>
    <Element_1_2/>
    <Element_1_3/>
  </Element_1>
  <Element_2/>
  <Element_3/>
</Root>

```

In the following PowerScript code fragment, the array `pbdom_obj_array` contains just one `PBDOM_ELEMENT` which represents the element `Root`:  
`pbdom_obj_array[1] - <Root>`:

```

PBDOM_DOCUMENT pbdom_doc
PBDOM_OBJECT pbdom_obj_array[]
...
pbdom_doc.GetContent(pbdom_obj_array)
pbdom_doc.GetRootElement().GetContent(pbdom_obj_array)

```

The call to `GetRootElement` in the last line of the previous code fragment yields an array that contains:

```

pbdom_obj_array[1] - <Element_1>
pbdom_obj_array[2] - <Element_2>
pbdom_obj_array[3] - <Element_3>

```

The returned `PBDOM_OBJECT` array can be manipulated. For example, the following statement causes `Element_2` to contain the Text node “Element 2 Text”:

```

pbdom_obj_array[2].AddContent ("Element 2 Text")

```

After this call, the tree is as follows:

```

<Root>
  Element_1>
    Element_1_1/>
    Element_1_2/>
    Element_1_3/>
  /Element_1>
  Element_2>Element 2 Text<Element_2/>
  Element_3/>
</Root>

```

- Usage** The returned array is passed by reference, with items in the same order in which they appear in the `PBDOM_DOCUMENT` object. Any changes to any item of the array affect the actual item to which it refers.
- See also** `AddContent`, `InsertContent`, `RemoveContent`, `SetContent`

## GetDocType

**Description** Allows you to retrieve the DOCTYPE declaration of the current XML DOM document.

**Syntax** `pbdom_document_name.GetDocType()`

Argument	Description
<code>pbdom_document_name</code>	The name of a PBDOM_DOCUMENT object

**Return value** PBDOM\_DOCTYPE.

**Throws** EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

**Usage** The DOCTYPE declaration is housed in a PBDOM\_OBJECT.

## GetElementsByTagName

**Description** Retrieves all the elements in the XML document that have the specified TagName.

**Syntax** `pbdom_object_name.GetElementsByTagName(string strTagName, ref pbdom_element pbdom_element_array[])`

Argument	Description
<code>strTagName</code>	The TagName of the elements to be searched for
<code>pbdom_element_array[]</code>	A reference to a PBDOM_ELEMENT object array that has the specified TagName

**Return value** Boolean. GetElementsByTagName returns true for success and false if an exception occurs.

**Examples** Assume a PBDOM\_DOCUMENT contains the following XML fragment:

```
<book>
  <title>The Winter's Tale</title>
  <author>William Shakespeare</author>
  <price>7.95</price>
  <quantity>1</quantity>
</book>
<book>
  <title>Le Lecon</title>
  <author>Eugene Ionesco</author>
  <price>10.95</price>
  <quantity>1</quantity>
</book>
```

```

<book>
  <title>Deutsches Tempo</title>
  <author>Kurt Tucholsky</author>
  <price>13.95</price>
  <quantity>1</quantity>
</book>

```

The following statements extract the list of titles from the document and display it in a multilineedit control:

```

pbdom_document doc
pbdom_element element[]

// doc contains role elements
boolean bb_bool

bb_bool = doc.getelementsbytagname("title",element[])

integer ii_bound, i

ii_bound = upperbound(element)
for i = 1 to ii_bound
  mle_1.text += element[i].gettext() + "~r~n"
next

```

## GetObjectClass

Description	Returns a long integer code that indicates the class of the current PBDOM_OBJECT.				
Syntax	<i>pbdom_object_name</i> .GetObjectClass()				
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_object_name</i></td> <td>The name of a PBDOM_OBJECT</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT
Argument	Description				
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT				
Return value	Long. GetObjectClass returns a long integer code that indicates the class of the current PBDOM_OBJECT. If <i>pbdom_object_name</i> is a PBDOM_DOCUMENT object, the returned value is 2.				

## GetObjectClassString

Description Returns a string form of the class of the PBDOM\_OBJECT.

Syntax *pbdom\_object\_name*.GetObjectClassString()

Argument	Description
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

Return value String. GetObjectClassString returns a string that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_DOCUMENT object, the returned string is “pbdom\_document”.

## GetRootElement

Description Retrieves the root element of the current XML DOM document.

Syntax *pbdom\_document\_name*.GetRootElement()

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object

Return value PBDOM\_ELEMENT. The root element of the PBDOM\_DOCUMENT object housed in a PBDOM\_ELEMENT object.

Throws EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

Usage The return value is the root element encapsulated in a PBDOM\_ELEMENT object.

See also DetachRootElement  
HasRootElement  
SetRootElement

## HasChildren

**Description** Returns true if the current PBDOM\_DOCUMENT object has at least one child PBDOM\_OBJECT, and false if it has none.

**Syntax** `pbdom_document_name.HasChildren()`

Argument	Description
<code>pbdom_document_name</code>	The name of a PBDOM_DOCUMENT object

**Return value** Boolean. Returns true if the current PBDOM\_DOCUMENT object has at least one child PBDOM\_OBJECT, and false otherwise.

## HasRootElement

**Description** Returns true if this document has a root element.

**Syntax** `pbdom_document_name.HasRootElement()`

Argument	Description
<code>pbdom_document_name</code>	The name of a PBDOM_DOCUMENT object

**Return value** Boolean. Returns true if the current PBDOM\_DOCUMENT object has a root element, and false otherwise.

**See also** DetachRootElement  
GetRootElement  
SetRootElement

## InsertContent

**Description** Inserts a new PBDOM\_OBJECT into the current PBDOM\_DOCUMENT object.

**Syntax** `pbdom_document_name.InsertContent(pbdom_object pbdom_object_new, pbdom_object pbdom_object_ref)`

Argument	Description
<code>pbdom_document_name</code>	The name of a PBDOM_DOCUMENT object
<code>pbdom_object_new</code>	The PBDOM_OBJECT to insert
<code>pbdom_object_ref</code>	The PBDOM_OBJECT in front of which the new PBDOM_OBJECT will be inserted

Return value	PBDOM_OBJECT. The modified PBDOM_DOCUMENT object returned as a PBDOM_OBJECT.
Throws	<p>EXCEPTION_INVALID_ARGUMENT – The input PBDOM_OBJECT to insert is invalid. This can happen if it has not been initialized properly or is a null object reference.</p> <p>EXCEPTION_USE_OF_UNNAMED_PBDOM_OBJECT – The input PBDOM_OBJECT to insert has not been given a user-defined name. The same exception is thrown if the reference PBDOM_OBJECT is also not given a user-defined name, unless the reference PBDOM_OBJECT is specifically set to null.</p> <p>EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – The input PBDOM_OBJECT to insert is not associated with a derived PBDOM_OBJECT. The same exception is thrown if the reference PBDOM_OBJECT is also not associated with a derived PBDOM_OBJECT, unless the reference PBDOM_OBJECT is specifically set to null.</p> <p>EXCEPTION_PBDOM_OBJECT_ALREADY_HAS_PARENT – The input PBDOM_OBJECT to insert already as a parent.</p> <p>EXCEPTION_MULTIPLE_ROOT_ELEMENT – A PBDOM_ELEMENT is to be inserted, but this document already has a root element.</p> <p>EXCEPTION_MULTIPLE_DOCTYPE – A PBDOM_DOCTYPE is to be inserted, but this document already has a DOCTYPE.</p> <p>EXCEPTION_HIERARCHY_ERROR – Inserting the PBDOM_OBJECT adversely affects how well-formed the document is.</p> <p>EXCEPTION_INAPPROPRIATE_USE_OF_PBDOM_OBJECT – An invalid PBDOM_OBJECT is to be inserted. See AddContent on page 200 for information on the valid PBDOM_OBJECTs that can be added to a PBDOM_DOCUMENT object.</p> <p>EXCEPTION_WRONG_PARENT_ERROR – The reference PBDOM_OBJECT is not a child of this PBDOM_DOCUMENT object.</p>
Examples	<p>A PBDOM_DOCUMENT object is created from an XML string. The PBDOM_ELEMENT pbdom_elem_1 is also created and set as Elem_1. The PBDOM_DOCTYPE pbdom_doctype_1 and the root element pbdom_root_elem are set.</p> <p>The root element is detached from its parent, which is also the PBDOM_DOCUMENT object itself. This makes it possible to insert pbdom_elem_1 into the document specifically before pbdom_doctype_1.</p>



```

pbdom_builder pbdom_builder_1
pbdom_document pbdom_doc
pbdom_doctype pbdom_doctype_1
pbdom_element pbdom_elem_1
pbdom_element pbdom_elem_root
string strXML

strXML = "<!DOCTYPE abc [<!-- internal subset -->"
strXML += "<!ELEMENT abc (#PCDATA)> "
strXML += "<!ELEMENT data&(#PCDATA)> "
strXML += "<!ELEMENT inner_data (#PCDATA)>]><abc>"
strXML += "Root Element Data<data>ABC Data<inner_data>"
strXML += "My Inner Data</inner_data>My Data</data>"
strXML += " now with extra& info</abc>"

pbdom_builder_1 = Create PBDOM_Builder
pbdom_elem_1 = Create PBDOM_Element

pbdom_doc = pbdom_builder_1.BuildFromString (strXML)
pbdom_elem_1.SetName ("Elem_1")
pbdom_doctype_1 = pbdom_doc.GetDocType()
pbdom_elem_root = pbdom_doc.GetRootElement()

pbdom_elem_root.Detach()
pbdom_doc.InsertContent(pbdom_elem_1, pbdom_doctype_1

```

The result is the following document, which is not well-formed:

```

<Elem_1/>
<!DOCTYPE abc [<!-- internal subset -->
<!ELEMENT abc (#PCDATA)*> <!ELEMENT data (#PCDATA)*>
<!ELEMENT inner_data (#PCDATA)*>]>

```

#### Usage

When a new PBDOM\_OBJECT is inserted into the current PBDOM\_DOCUMENT object, the new PBDOM\_OBJECT becomes a child node of the current PBDOM\_DOCUMENT object. Also, the new PBDOM\_OBJECT is to be positioned specifically before another PBDOM\_OBJECT, denoted using the second parameter.

If the second PBDOM\_OBJECT is specified as null, then the new PBDOM\_OBJECT is to be inserted at the end of the list of children of the current PBDOM\_DOCUMENT object.

#### See also

AddContent  
 GetContent  
 RemoveContent  
 SetContent

## IsAncestorObjectOf

**Description** The IsAncestorObjectOf method determines whether the current PBDOM\_DOCUMENT object is the ancestor of another PBDOM\_OBJECT.

**Syntax** *pbdom\_document\_name*.IsAncestorObjectOf(*pbdom\_object*, *pbdom\_object\_ref*)

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to check against

**Return value** Boolean. Returns true if the current PBDOM\_DOCUMENT object is the ancestor of the referenced PBDOM\_OBJECT, and false otherwise.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – The input PBDOM\_OBJECT is invalid. This can happen if it has not been initialized properly or is a null object reference.

## NewDocument

**Description** The NewDocument method is overloaded:

- Syntax 1 creates a new XML DOM document using the name of the root element to be contained within the new DOM document.
- Syntax 2 creates a new XML DOM document using the name and namespace URI of the root element to be contained in the new DOM document, and also the external subset public and system identifiers.

**Syntax**

For this syntax	See
NewDocument(string <i>strRootElementName</i> )	NewDocument Syntax 1
NewDocument( string <i>strRootElementNamespacePrefix</i> , string <i>strRootElementNamespaceURI</i> , string <i>strRootElementName</i> , string <i>strDocTypePublicId</i> , string <i>strDocTypeSystemId</i> )	NewDocument Syntax 2

## NewDocument Syntax 1

Description	Creates a new XML DOM document from scratch.						
Syntax	<i>pbdom_document_name</i> .NewDocument( <i>strRootElementName</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_document_name</i></td> <td>The name of a PBDOM_DOCUMENT object</td> </tr> <tr> <td><i>strRootElementName</i></td> <td>The name of the root element to be contained in the DOM document</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object	<i>strRootElementName</i>	The name of the root element to be contained in the DOM document
Argument	Description						
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object						
<i>strRootElementName</i>	The name of the root element to be contained in the DOM document						
Return value	Boolean. Returns true if a new document is successfully created and false otherwise.						
Throws	EXCEPTION_INVALID_ARGUMENT – The input string is invalid, which can occur if the string was set to null by means of the PowerScript SetNull method.  EXCEPTION_MEMORY_ALLOCATION_FAILURE – Insufficient memory was encountered while executing this method.						
Usage	The parameter <i>strRootElementName</i> becomes the name of the root element.						
See also	SaveDocument						

## NewDocument Syntax 2

Description	Creates a new XML DOM document from scratch.														
Syntax	<i>pbdom_document_name</i> .NewDocument(string <i>strRootElementNamespacePrefix</i> , string <i>strRootElementNamespaceURI</i> , string <i>strRootElementName</i> , string <i>strDocTypePublicId</i> , string <i>strDocTypeSystemId</i> )														
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_document_name</i></td> <td>The name of a PBDOM_DOCUMENT object.</td> </tr> <tr> <td><i>strRootElementNamespacePrefix</i></td> <td>The namespace prefix of the root element to be contained in the DOM document. This can be an empty string.</td> </tr> <tr> <td><i>strRootElementNamespaceURI</i></td> <td>The namespace URI of the root element to be contained in the DOM document. This can be an empty string.</td> </tr> <tr> <td><i>strRootElementName</i></td> <td>The name of the root element to be contained in the DOM document.</td> </tr> <tr> <td><i>strDocTypePublicId</i></td> <td>The external subset public identifier.</td> </tr> <tr> <td><i>strDocTypeSystemId</i></td> <td>The external subset system identifier.</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object.	<i>strRootElementNamespacePrefix</i>	The namespace prefix of the root element to be contained in the DOM document. This can be an empty string.	<i>strRootElementNamespaceURI</i>	The namespace URI of the root element to be contained in the DOM document. This can be an empty string.	<i>strRootElementName</i>	The name of the root element to be contained in the DOM document.	<i>strDocTypePublicId</i>	The external subset public identifier.	<i>strDocTypeSystemId</i>	The external subset system identifier.
Argument	Description														
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object.														
<i>strRootElementNamespacePrefix</i>	The namespace prefix of the root element to be contained in the DOM document. This can be an empty string.														
<i>strRootElementNamespaceURI</i>	The namespace URI of the root element to be contained in the DOM document. This can be an empty string.														
<i>strRootElementName</i>	The name of the root element to be contained in the DOM document.														
<i>strDocTypePublicId</i>	The external subset public identifier.														
<i>strDocTypeSystemId</i>	The external subset system identifier.														

Return value	Boolean. Returns true if a new document is successfully created, and false otherwise.
Throws	<p>EXCEPTION_INVALID_ARGUMENT – One of the input strings is invalid. This can happen if the string has been set to null using the PowerScript SetNull method.</p> <p>EXCEPTION_MEMORY_ALLOCATION_FAILURE – Insufficient memory was encountered while executing this method.</p> <p>EXCEPTION_INVALID_NAME – The root element name, or the root element namespace prefix or URI, is invalid.</p> <p>EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – This PBDOM_DOCUMENT object's internal implementation is NULL. The occurrence of this exception is rare but can take place if severe memory corruption occurs.</p>

**Examples** **Example 1** This example attempts to create a PBDOM\_DOCUMENT object in which the root element belongs to no namespace, as indicated by the empty strings for the namespace prefix and URI arguments to NewDocument:

```
PBDOM_DOCUMENT pbdom_doc

try
  pbdom_doc = Create PBDOM_DOCUMENT
  pbdom_doc.NewDocument ("", "", "root", "public_id", &
    "system_id.dtd")

  pbdom_doc.SaveDocument &
    ("new_document_no_namespace.xml")

catch (PBDOM_EXCEPTION except)
  MessageBox ("PBDOM_EXCEPTION", except.GetMessage())
end try
```

When serialized, the XML document looks like the following :

```
<!DOCTYPE root PUBLIC "public_id" "system_id.dtd">
<root xmlns="" />
```

The namespace declaration attribute (xmlns="") present in the root element indicates that the root element belongs to no namespace.

**Example 2** This example attempts to create a PBDOM\_DOCUMENT object in which the root element belongs to a default namespace. The URI is <http://www.pre.com>, which means that the root element belongs to the namespace <http://www.pre.com>. The prefix is an empty string, which means that the root element belongs to the <http://www.pre.com> namespace by default:

```

PBDOM_DOCUMENT pbdom_doc

try
    pbdom_doc = Create PBDOM_DOCUMENT
    pbdom_doc.NewDocument ("", "http://www.pre.com", &
        "root", "public_id", "system_id.dtd")

    pbdom_doc.SaveDocument &
        ("new_document_default_namespace.xml")

catch (PBDOM_EXCEPTION except)
    MessageBox ("PBDOM_EXCEPTION", except.GetMessage())
end try

```

When serialized, the XML document looks like the following :

```

<!DOCTYPE root PUBLIC "public_id" "system_id.dtd">
<root xmlns="http://www.pre.com"/>

```

The namespace declaration attribute (`xmlns="http://www.pre.com"`) present in the root element indicates that the root element belongs to the default namespace `http://www.pre.com`. All child elements of root belong to this same namespace unless another in-scope namespace declaration is present and is used.

**Example 3** This example attempts to create a PBDOM\_DOCUMENT object in which the root element belong to a prefixed namespace. The namespace prefix is `pre` and the URI is `http://www.pre.com`. This means that the root element will belong to the namespace `http://www.pre.com`, and that the root element will have a namespace prefix of `pre`:

```

PBDOM_DOCUMENT pbdom_doc

try
    pbdom_doc = Create PBDOM_DOCUMENT
    pbdom_doc.NewDocument ("pre", "http://www.pre.com", &
        "root", "public_id", "system_id.dtd")

    pbdom_doc.SaveDocument &
        ("new_document_namespace.xml")

catch (PBDOM_EXCEPTION except)
    MessageBox ("PBDOM_EXCEPTION", except.GetMessage())
end try

```

When serialized, the XML document looks like the following :

```
<!DOCTYPE pre:root PUBLIC "public_id" "system_id.dtd">
<pre:root xmlns:pre="http://www.pre.com"/>
```

A namespace declaration attribute (`xmlns:pre="http://www.pre.com"`) is present in the root element. The root element also contains a `pre` prefix. This indicates that the root element belongs to the namespace `http://www.pre.com`.

However, the fact that the `http://www.pre.com` namespace is prefixed by `pre` indicates that the child elements of root belong to this same namespace only if their qualified names also contain the `pre` prefix and there is an in-scope namespace declaration for `http://www.pre.com` that is prefixed by `pre`.

**Usage** Using the five parameters available with this syntax provides more control over the DOCTYPE definition of the document.

**See also** [SaveDocument](#)

## RemoveContent

**Description** Removes a child PBDOM\_OBJECT from the current PBDOM\_DOCUMENT object.

**Syntax** `pbdom_document_name.RemoveContent(pbdom_object pbdom_object_ref)`

Argument	Description
<code>pbdom_document_name</code>	The name of a PBDOM_DOCUMENT object
<code>pbdom_object_ref</code>	The PBDOM_OBJECT to remove

**Return value** Boolean. Returns true if the content was removed, and false otherwise.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – The input PBDOM\_OBJECT to remove is invalid. This can happen if it has not been initialized properly or is a null object reference.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – The input PBDOM\_OBJECT is nameable, but it has not been assigned a name.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – The input PBDOM\_OBJECT is not associated with a derived PBDOM\_OBJECT class object.

EXCEPTION\_WRONG\_DOCUMENT\_ERROR – The input PBDOM\_OBJECT is not contained within the current PBDOM\_DOCUMENT object.

EXCEPTION\_WRONG\_PARENT\_ERROR – The input PBDOM\_OBJECT is not a child of the current PBDOM\_DOCUMENT object.

Usage	When a PBDOM_OBJECT is removed from the current PBDOM_DOCUMENT object, all children under the removed PBDOM_OBJECT are also removed.
See also	AddContent GetContent InsertContent SetContent

## SaveDocument

**Description** Saves the serialized XML string of the DOM tree contained within the PBDOM\_DOCUMENT object into a disk file.

**Syntax** *pbdom\_document\_name*.SaveDocument(string *strFileName*)

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object
<i>strFileName</i>	The name of the disk file to which the contents of the current PBDOM_DOCUMENT object is to be serialized

**Return value** Boolean. Returns true if a new document was successfully saved to a disk file, and false otherwise.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – The input string specifying the file name is invalid. This can happen if the string has been set to null using the PowerShell SetNull method.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

**Usage** A PBDOM\_DOCUMENT object that has been created from an existing XML document or string can differ from its original after it has been converted back to an XML string or document. This can occur even if no modifications have been made to the PBDOM\_DOCUMENT object using PowerShell.

This can occur if the original XML document or string referred to an external DTD that mandates the inclusion of default attributes. In this case, PBDOM complies with the rules of the DTD and inserts these required attributes into the relevant elements while building up the in-memory DOM tree.

When the PBDOM\_DOCUMENT object is saved and converted back to an XML document, these default attributes are saved in the document.

**See also** NewDocument

## SaveDocumentIntoString

**Description** Saves the serialized XML string of the DOM tree contained within the PBDOM\_DOCUMENT object into a string.

**Syntax** `pbdom_document_name.SaveDocumentIntoString()`

<b>Argument</b>	<b>Description</b>
<code>pbdom_document_name</code>	The name of a PBDOM_DOCUMENT object

**Return value** String. Returns a string containing the XML string of the PBDOM\_DOCUMENT.

**Examples** This code creates a new PBDOM\_DOCUMENT and saves it to the string *ls\_xml*:

```
PBDOM_DOCUMENT pbdom_doc
string ls_xml

try
    pbdom_doc = Create PBDOM_DOCUMENT
    pbdom_doc.NewDocument ("pre", "http://www.pre.com", &
        "root", "public_id", "system_id.dtd")
    ls_xml = pbdom_doc.SaveDocumentIntoString
catch (PBDOM_EXCEPTION except)
    MessageBox ("PBDOM_EXCEPTION", except.GetMessage())
end try
```

**See also** SaveDocument



## SetContent

**Description** Sets the entire content of the PBDOM\_DOCUMENT object, removing pre-existing children first.

**Syntax** `pbdom_document_name.SetContent(pbdom_object pbdom_object_array)`

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object
<i>pbdom_object_array</i>	An array of PBDOM_OBJECTs set as the contents of the PBDOM_DOCUMENT object

*pbdom\_object\_array* must contain only PBDOM\_OBJECT objects that can legally be set as the contents of a PBDOM\_DOCUMENT object. The SetContent method restricts the array to one PBDOM\_ELEMENT object to set as the root element of the PBDOM\_DOCUMENT object from which the method is invoked. The SetContent method also restricts the array to one PBDOM\_DOCTYPE object to set as the DOCTYPE of the PBDOM\_DOCUMENT object.

**Return value** PBDOM\_OBJECT. The modified PBDOM\_DOCUMENT object returned as a PBDOM\_OBJECT.

**Throws** EXCEPTION\_ILLEGAL\_PBOBJECT – An array item is not a valid PBDOM object. This can happen if the array item has not been initialized properly or is a null object reference.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – An array item is nameable and has not been given a user-defined name.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – An array item is not associated with a derived PBDOM\_OBJECT.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – An array item already has a parent PBDOM\_OBJECT.

EXCEPTION\_MULTIPLE\_ROOT\_ELEMENT – The array contains more than one PBDOM\_ELEMENT. The array must contain at most one PBDOM\_ELEMENT that is set as the root element of this document.

EXCEPTION\_MULTIPLE\_DOCTYPE – The array contains more than one PBDOM\_DOCTYPE. The array must contain at most one PBDOM\_DOCTYPE that is set as the DOCTYPE of this document.

EXCEPTION\_MULTIPLE\_XMLDECL – The array contains more than one PBDOM\_PROCESSINGINSTRUCTION that has been constructed into an XML Declaration.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – An array item is not allowed to be set as a document-level content.

**Usage** The supplied array contains PBDOM\_OBJECTs that can legally be set as the content of a PBDOM\_DOCUMENT object.

For example, a PBDOM\_DOCUMENT object accepts only an array that contains PBDOM\_ELEMENT, PBDOM\_COMMENT, PBDOM\_DOCTYPE, or PBDOM\_PROCESSINGINSTRUCTION objects. In addition, the array can contain at most one PBDOM\_ELEMENT object that it sets as its root element, at most one PBDOM\_DOCTYPE object that it sets as its DOCTYPE, and at most one XML declaration .PBDOM\_PROCESSINGINSTRUCTION.

In the event of an exception, the original contents of this PBDOM\_DOCUMENT object are unchanged, and the PBDOM\_OBJECTs contained in the supplied array are unaltered.

**See also** AddContent, GetContent, InsertContent, RemoveContent

## SetDocType

**Description** Sets the DOCTYPE declaration of this document.

**Syntax** *pbdom\_document\_name*.SetDocType(*pbdom\_doctype* *pbdom\_doctype\_ref*)

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object
<i>pbdom_doctype_ref</i>	A PBDOM_DOCTYPE object to be set as the DOCTYPE of this document

**Return value** PBDOM\_DOCUMENT. The same PBDOM\_DOCUMENT object with a modified DOCTYPE declaration.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – The input PBDOM\_DOCTYPE is invalid. This can happen if it has not been initialized properly or is a null object reference.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – The input PBDOM\_DOCTYPE is nameable and has not been given a user-defined name.

EXCEPTION\_WRONG\_DOCUMENT\_ERROR – The input PBDOM\_DOCTYPE already has an owner document.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – The input PBDOM\_DOCTYPE is already the DOCTYPE of another document.

**Usage** If this document already contains a DOCTYPE declaration, the new PBDOM\_DOCTYPE replaces it. The DOCTYPE of a PBDOM\_DOCUMENT object can be changed multiple times, and it is legal for a user to call the SetDocType method multiple times.

A DOM DOCTYPE object can have no owner document, or it can have an owner document but no parent node. A DOCTYPE that has an owner document as well as a parent node is the actual DOCTYPE of the owner document.

## SetRootElement

**Description** Sets the root element for this document.

**Syntax** *pbdom\_document\_name*.SetRootElement(*pbdom\_element pbdom\_element\_ref*)

Argument	Description
<i>pbdom_document_name</i>	The name of a PBDOM_DOCUMENT object
<i>pbdom_element_ref</i>	A PBDOM_ELEMENT object to be set as the root element for this document

**Return value** PBDOM\_DOCUMENT. The PBDOM\_DOCUMENT object with a modified root element.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – The input PBDOM\_ELEMENT is invalid. This can happen if it has not been initialized properly or is a null object reference.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – The input PBDOM\_ELEMENT is nameable and it has not been given a user-defined name.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – The input PBDOM\_ELEMENT already has a parent PBDOM\_OBJECT.

**Usage** If this document already has a root element, the existing root element is replaced. The root element of a PBDOM\_DOCUMENT object can be changed multiple times, and it is legal for a user to call the SetRootElement method multiple times.

**See also** DetachRootElement  
GetRootElement  
HasRootElement



About this chapter

This chapter describes the PBDOM\_ELEMENT class.

## **PBDOM\_ELEMENT**

Description

The PBDOM\_ELEMENT class defines the behavior for an XML element modeled in PowerScript. Methods allow the user to obtain the text content of an element, the attributes of an element, and the children of an element.

In PBDOM, an XML element's attributes are *not* its children. Attributes are properties of elements rather than having a separate identity from the elements with which they are associated. An element's PBDOM\_ATTRIBUTE objects do not have sibling relationships with each other in the same way as the element's children.

For more information on the relationships among PBDOM\_ELEMENT and PBDOM\_ATTRIBUTE objects, see the chapter on XML services in *Application Techniques*.

Methods

PBDOM\_ELEMENT has the following methods:

AddContent	GetTextTrim
AddNamespaceDeclaration	HasAttributes
Clone	HasChildElements
Detach	HasChildren
Equals	InsertContent
GetAttribute	IsAncestorObjectOf
GetAttributes	IsRootElement
GetAttributeValue	RemoveAttribute
GetChildElement	RemoveChildElement
GetChildElements	RemoveChildElements
GetContent	RemoveContent
GetName	RemoveNamespaceDeclaration
GetNamespacePrefix	SetAttribute
GetNamespaceUri	SetAttributes
GetObjectClass	SetContent
GetObjectClassString	SetDocument
GetOwnerDocumentObject	SetName
GetParentObject	SetNamespace
GetQualifiedName	SetParentObject
GetText	SetText
GetTextNormalize	

## AddContent

Description

The AddContent method is overloaded:

- Syntax 1 adds a new PBDOM\_OBJECT into a PBDOM\_ELEMENT object.
- Syntax 2 adds a new text string to the PBDOM\_ELEMENT object from which the method is invoked.

Syntax

For this syntax	See
AddContent(pbdom_object <i>pbdom_object_ref</i> )	AddContent Syntax 1
AddContent(string <i>strText</i> )	AddContent Syntax 2

## AddContent Syntax 1

**Description** Adds a new PBDOM\_OBJECT into a PBDOM\_ELEMENT object. The added PBDOM\_OBJECT becomes a child of the PBDOM\_ELEMENT object.

**Syntax** `pbdom_element_name.AddContent(pbdom_object pbdom_object_ref)`

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to add

**Return value** PBDOM\_OBJECT. The PBDOM\_ELEMENT object modified and returned as a PBDOM\_OBJECT.

**Throws** EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If an invalid PBDOM\_OBJECT is added. See description section below on the valid PBDOM\_OBJECTs that can be added to a PBDOM\_ELEMENT object. This exception is also thrown if the input PBDOM\_OBJECT is this PBDOM\_ELEMENT object itself.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT has not been given a user-defined name.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not associated with a derived PBDOM\_OBJECT.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – If the input PBDOM\_OBJECT already has a parent PBDOM\_OBJECT.

EXCEPTION\_HIERARCHY\_ERROR – If adding the input PBDOM\_OBJECT will cause the current PBDOM\_ELEMENT object to be no longer well-formed.

**Examples** The AddContent method is invoked for the Element\_2 PBDOM\_ELEMENT object in the following XML fragment:

```
<Element_1>
  <Element_1_1/>
  <Element_1_2/>
  <Element_1_3/>
</Element_1>
<Element_2>Element 2 Text</Element_2>
<Element_3/>
```

The AddContent is invoked from the following PowerScript code, where pbdom\_elem\_2 represents the Element\_2 object:

```
PBDOM_ELEMENT pbdom_elem
pbdom_elem = Create PBDOM_ELEMENT
pbdom_elem.SetName("Sub_Element")
pbdom_elem.AddContent("Sub Element Text")
pbdom_elem_2.AddContent (pbdom_elem)
```

The following XML fragment results:

```
<Element_1>
  <Element_1_1/>
  <Element_1_2/>
  <Element_1_3/>
</Element_1>
<Element_2>
  Element 2 Text
  <Sub_Element>
    Sub Element Text
  </Sub_Element>
<Element_2/>
<Element_3/>
```

### Usage

Only the following PBDOM\_OBJECT types can be validly added to a PBDOM\_ELEMENT object:

- PBDOM\_ELEMENT
- PBDOM\_CDATA
- PBDOM\_COMMENT
- PBDOM\_ENTITYREFERENCE
- PBDOM\_PROCESSINGINSTRUCTION
- PBDOM\_TEXT

### See also

AddContent Syntax 2  
GetContent  
InsertContent  
RemoveContent  
SetContent



## AddContent Syntax 2

**Description** Adds a new text string to the PBDOM\_ELEMENT object from which the method is invoked.

**Syntax** `pbdom_element_name.AddContent(string strText)`

Argument	Description
<code>pbdom_element_name</code>	The name of a PBDOM_ELEMENT object
<code>strText</code>	A string to be added to the PBDOM_ELEMENT object as new text content

**Return value** PBDOM\_OBJECT. The PBDOM\_ELEMENT object modified and returned as a PBDOM\_OBJECT.

**Examples** The AddContent method is invoked for the abc element of the following XML document:

```
<abc>
  Root Element Data
  <data>
    ABC Data
    <inner_data>My Inner Data</inner_data>
  </data>
</abc>
```

The AddContent method is invoked from the following PowerScript statement:

```
pbdom_doc.GetRootElement().AddContent(" And More !")
```

The following XML results:

```
<abc>
  Root Element Data
  <data>
    ABC Data
    <inner_data>My Inner Data</inner_data>
  </data>
  And More !
</abc>
```

**See also** AddContent Syntax 1  
GetContent  
InsertContent  
RemoveContent  
SetContent

## AddNamespaceDeclaration

**Description** Adds a new namespace declaration to this PBDOM\_ELEMENT object. The new namespace can apply to the PBDOM\_ELEMENT object itself if the namespace becomes the default namespace in the PBDOM\_ELEMENT object.

**Syntax** *pbdom\_element\_name*.AddNamespaceDeclaration(string *strNamespacePrefix*, string *strNamespaceUri*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strNamespacePrefix</i>	The prefix of the new namespace to be declared
<i>strNamespaceUri</i>	The URI of the new namespace to be declared

**Return value** PBDOM\_ELEMENT. The modified PBDOM\_ELEMENT object.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – If any of the input parameters is invalid (null).

EXCEPTION\_INVALID\_NAME – If the input Prefix is invalid, as, for example, if it contains a colon.

EXCEPTION\_INVALID\_STRING – If the input URI is invalid.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – If memory allocation failure occurred in this method.

**Examples** Consider the following element:

```
<Vehicle>
  <seats>4</seats>
  <color>Red</color>
  <engine>
    <capacity units="cc">1600</capacity>
  </engine>
</Vehicle>
```

Given a PBDOM\_ELEMENT object *elem\_vehicle* that represents the Vehicle element, consider the following statement:

```
elem_vehicle.AddNamespaceDeclaration("vehicle_specs", &
  "http://www.vehicle.com/specs")
```

It transforms the Vehicle element as follows:

```
<Vehicle
xmlns:vehicle_specs="http://www.vehicle.com/specs">
  <seats>4</seats>
  <color>Red</color>
  <engine>
    <capacity units="cc">1600</capacity>
  </engine>
</Vehicle>
```

Vehicle, seats, color, engine, and capacity are all unqualified (that is, they have no namespace prefix). Therefore, the vehicle\_specs namespace does not apply to any of them or their attributes or subelements.

However, consider the following statement:

```
elem_vehicle.AddNamespaceDeclaration("", &
    "http://www.vehicle.com/specs")
```

It transforms the Vehicle element as follows:

```
<Vehicle xmlns:"http://www.vehicle.com/specs">
  <seats>4</seats>
  <color>Red</color>
  <engine>
    <capacity units="cc">1600</capacity>
  </engine>
</Vehicle>
```

http://www.vehicle.com/specs is the default namespace and so Vehicle, seats, color, engine, and capacity are all part of this namespace. Note that the default namespace does *not* apply to the units attribute.

See also

- GetNamespacePrefix
- GetNamespaceUri
- GetQualifiedName
- RemoveNamespaceDeclaration
- SetNamespace

## Clone

Description Creates a clone of a PBDOM\_ELEMENT object.

Syntax *pbdom\_element\_name*.Clone(boolean *bDeep*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object.
<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are true for a deep clone and false for a shallow clone.

Return value PBDOM\_OBJECT. A clone of this PBDOM\_ELEMENT object returned as a PBDOM\_OBJECT.

Examples The Clone method is used to alter the following XML:

```
<Telephone_Book>
  <Entry>
    <Particulars>
      <Name>John Doe</Name>
      <Age>21</Age>
      <Phone_Number>1234567</Phone_Number>
    </Particulars>
  </Entry>
</Telephone_Book>
```

The Clone method is invoked from the following PowerScript code, where entry represents the Entry> element in the preceding XML:

```
PBDOM_ELEMENT elem_clone

elem_clone = entry.Clone(true)
pbdom_doc.AddContent(elem_clone)
```

The resulting XML contains two identical Entry> elements:

```
<Telephone_Book>
  <Entry>
    <Particulars>
      <Name>John Doe</Name>
      <Age>21</Age>
      <Phone_Number>1234567</Phone_Number>
    </Particulars>
  </Entry>
  <Entry>
    <Particulars>
      <Name>John Doe</Name>
      <Age>21</Age>
```

```

        <Phone_Number>1234567</Phone_Number>
    </Particulars>
</Entry>
</Telephone_Book>

```

**Usage**

This method creates and returns a duplicate of the current PBDOM\_ELEMENT object. If a shallow clone is requested, this method clones the PBDOM\_ELEMENT object together with its namespace information values and its PBDOM\_ATTRIBUTES and their subtrees. If a deep clone is requested, this method additionally recursively clones the subtree under the PBDOM\_ELEMENT object.

A PBDOM\_ELEMENT clone has no parent. However, the clone resides in the same PBDOM\_DOCUMENT as its original, and if the original PBDOM\_ELEMENT object is standalone, the clone is standalone.

**Detach****Description**

Detaches a PBDOM\_ELEMENT object from its parent PBDOM\_OBJECT.

**Syntax**

*pbdom\_element\_name*.Detach()

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object

**Return value**

PBDOM\_OBJECT. The PBDOM\_ELEMENT object detached from its parent object and returned as a PBDOM\_OBJECT. If the PBDOM\_ELEMENT object has no parent, the Detach method does nothing.

**Equals****Description**

Tests for equality between the PBDOM\_ELEMENT object from which the method is invoked and a PBDOM\_OBJECT indicated by the method parameter.

**Syntax**

*pbdom\_element\_name*.Equals(*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>pbdom_object_ref</i>	A PBDOM_OBJECT to be tested for equality with this PBDOM_ELEMENT object

**Return value**

Boolean. Returns true if the PBDOM\_ELEMENT object is equivalent to the referenced PBDOM\_OBJECT and false otherwise.

Examples

The Equals method is invoked from the following PowerScript code, in which pbdom\_doc represents a PBDOM\_DOCUMENT object containing a root element:

```
PBDOM_ELEMENT pbdom_elem_1
PBDOM_ELEMENT pbdom_elem_2
PBDOM_OBJECT pbdom_obj
PBDOM_DOCUMENT pbdom_doc

pbdom_elem_1 = pbdom_doc.GetRootElement()
pbdom_elem_2 = pbdom_doc.GetRootElement()

IF pbdom_elem_1.Equals(pbdom_elem_2) THEN
    MessageBox ("Equals", "The objects are equal")
ELSE
    MessageBox ("Equals", "The objects are NOT equal")
END IF

pbdom_obj = Create PBDOM_ELEMENT
pbdom_obj.SetName ("An_Element")

IF pbdom_elem_1.Equals(pbdom_obj) THEN
    MessageBox ("Equals", "The objects are equal")
ELSE
    MessageBox ("Equals", "The objects are NOT equal")
END IF
```

Because pbdom\_elem\_1 and pbdom\_elem\_2 refer to the same root element, a message box reports that the objects are equal.

## GetAttribute

- Description
- The GetAttribute method is overloaded:
- Syntax 1 returns the PBDOM\_ATTRIBUTE object for a PBDOM\_ELEMENT object using the name of the PBDOM\_ATTRIBUTE.
  - Syntax 2 returns the PBDOM\_ATTRIBUTE object for a PBDOM\_ELEMENT object with the name provided and within the namespace specified by the prefix and URI provided.

### Syntax

For this syntax	See
GetAttribute(string <i>strName</i> )	GetAttribute Syntax 1
GetAttribute(string <i>strName</i> , string <i>strNamespacePrefix</i> , string <i>strNamespaceUri</i> )	GetAttribute Syntax 2

## GetAttribute Syntax 1

Description Returns the PBDOM\_ATTRIBUTE object for a PBDOM\_ELEMENT object.

Syntax *pbdom\_element\_name*.GetAttribute(string *strName*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strName</i>	The name of the PBDOM_ATTRIBUTE to be returned

Return value PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE object matching the name specified in the method parameter. If no such PBDOM\_ATTRIBUTE object exists, the GetAttribute method returns a value of null.

Throws EXCEPTION\_INVALID\_NAME – If the supplied name is a qualified name that contains a namespace prefix.

Examples The GetAttribute method is invoked for the following XML document:

```
<MyMusic:abc
xmlns:MyMusic="http://www.MyMusic_records.com"
My_Attr="My MyMusic Attribute">Root Element
Data</MyMusic:abc>
```

The GetAttribute method is invoked from the following PowerScript statement:

```
pbdom_attr = &
    pbdom_doc.GetRootElement().GetAttribute("My_Attr")
```

The GetAttribute method returns the PBDOM\_ATTRIBUTE object My\_Attr.

**Usage** If the PBDOM\_ATTRIBUTE name specified in the method parameter is a qualified name, an exception is thrown. A qualified name appears in the following form: [*namespace\_prefix*]:[*local\_name*].

**See also** GetAttribute Syntax 2  
 GetAttributes  
 GetAttributeValue  
 HasAttributes  
 SetAttribute  
 SetAttributes

## GetAttribute Syntax 2

**Description** Returns the PBDOM\_ATTRIBUTE object for a PBDOM\_ELEMENT object with the name provided and within the namespace specified by the prefix and URI provided.

**Syntax** *pbdom\_element\_name*.GetAttribute(string *strName*, string *strNamespacePrefix*, string *strNamespaceUri*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strName</i>	The name of the PBDOM_ATTRIBUTE to be returned
<i>strNamespacePrefix</i>	The prefix of the namespace of the PBDOM_ATTRIBUTE to return
<i>strNamespaceUri</i>	The URI of the namespace of the PBDOM_ATTRIBUTE to return

**Return value** PBDOM\_ATTRIBUTE. The PBDOM\_ATTRIBUTE object matching the name, namespace prefix, and URI specified in the method parameters. If no such PBDOM\_ATTRIBUTE object exists, the GetAttribute method returns a value of null.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – If any of the arguments is invalid, for example, null.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – If there was any memory allocation failure during the running of this method.

**See also** GetAttribute Syntax 1  
 GetAttributes  
 GetAttributeValue  
 HasAttributes  
 SetAttribute, SetAttributes



## GetAttributes

Description	Returns the complete set of PBDOM_ATTRIBUTE objects for a PBDOM_ELEMENT object.  If there are no PBDOM_ATTRIBUTE objects for the PBDOM_ELEMENT object, the GetAttributes method returns an empty array.						
Syntax	<i>pbdom_element_name</i> .GetAttributes(ref pbdom_attribute <i>pbdom_attribute_array</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_element_name</i></td> <td>The name of a PBDOM_ELEMENT object</td> </tr> <tr> <td><i>pbdom_attribute_array</i></td> <td>An empty and unbounded array to be filled with references to the PBDOM_ATTRIBUTE objects contained in the PBDOM_ELEMENT object</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object	<i>pbdom_attribute_array</i>	An empty and unbounded array to be filled with references to the PBDOM_ATTRIBUTE objects contained in the PBDOM_ELEMENT object
Argument	Description						
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object						
<i>pbdom_attribute_array</i>	An empty and unbounded array to be filled with references to the PBDOM_ATTRIBUTE objects contained in the PBDOM_ELEMENT object						
Return value	Boolean. Returns true if an array of PBDOM_ATTRIBUTE objects for the PBDOM_ELEMENT object has been retrieved, and false otherwise.						
Usage	GetAttributes returns the complete set of PBDOM_ATTRIBUTE objects for a PBDOM_ELEMENT object as an array of PBDOM_ATTRIBUTE objects, or as an empty list (empty array) if there are none. The returned array items are “live” and changes to any item affect the referenced PBDOM_ATTRIBUTE.						
See also	GetAttribute GetAttributeValue HasAttributes SetAttribute SetAttributes						

## GetAttributeValue

Description	The GetAttributeValue method is overloaded: <ul style="list-style-type: none"> <li>• Syntax 1 returns the string value of a PBDOM_ATTRIBUTE object with the specified name.</li> <li>• Syntax 2 returns the string value of a PBDOM_ATTRIBUTE object with the specified name, using the prefix and URI of the namespace of the PBDOM_ATTRIBUTE.</li> <li>• Syntax 3 returns the string value of a PBDOM_ATTRIBUTE object with the specified name, using the prefix and URI of the namespace of the PBDOM_ATTRIBUTE. Syntax 3 also provides a default string value to return if the attribute does not exist.</li> </ul>
-------------	---

- Syntax 4 returns the string value of a PBDOM\_ATTRIBUTE object with the specified name. Syntax 4 also provides a default string value to return if the attribute does not exist.

Syntax

For this syntax	See
GetAttributeValue(string <i>strAttributeName</i> )	GetAttributeValue Syntax 1
GetAttributeValue(string <i>strAttributeName</i> , string <i>strNamespacePrefix</i> , string <i>strNamespaceUri</i> )	GetAttributeValue Syntax 2
GetAttributeValue(string <i>strAttributeName</i> , string <i>strNamespacePrefix</i> , string <i>strNamespaceUri</i> , string <i>strDefaultValue</i> )	GetAttributeValue Syntax 3
GetAttributeValue(string <i>strAttributeName</i> , string <i>strDefaultValue</i> )	GetAttributeValue Syntax 4

## GetAttributeValue Syntax 1

Description

Returns the string value of the PBDOM\_ATTRIBUTE object (within a PBDOM\_ELEMENT object) with the specified name and within no namespace.

Syntax

*pbdom\_element\_name*.GetAttributeValue(string *strAttributeName*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strAttributeName</i>	The name of the attribute whose value is to be returned

Return value

String. The string value of the PBDOM\_ATTRIBUTE object specified in *strAttributeName*. If no such object exists, the GetAttributeValue method returns null.

Usage

If the text value of the PBDOM\_ATTRIBUTE object is empty, the GetAttributeValue method returns an empty string.

See also

GetAttribute  
 GetAttributeValue Syntax 2  
 GetAttributeValue Syntax 3  
 GetAttributeValue Syntax 4  
 HasAttributes  
 SetAttribute  
 SetAttributes

## GetAttributeValue Syntax 2

**Description** Returns the string value of the PBDOM\_ATTRIBUTE object (within a PBDOM\_ELEMENT object) with the specified name and within the specified namespace.

**Syntax** *pbdom\_element\_name*.GetAttributeValue( string *strAttributeName*, string *strNamespacePrefix*, string *strNamespaceUri*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strAttributeName</i>	The name of the attribute whose value is to be returned
<i>strNamespacePrefix</i>	The prefix of the namespace of the PBDOM_ATTRIBUTE whose value is to be returned
<i>strNamespaceUri</i>	The URI of the namespace of the PBDOM_ATTRIBUTE whose value is to be returned

**Return value** String. The string value of the PBDOM\_ATTRIBUTE object specified in *strAttributeName*. If no such object exists, the GetAttributeValue method returns an empty string.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – If any of the input arguments is invalid, for example, null.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – If there was any memory allocation failure during the execution of this method.

EXCEPTION\_INVALID\_NAME – If the input attribute name or namespace prefix or namespace URI is invalid.

**See also** GetAttribute  
GetAttributeValue Syntax 1  
GetAttributeValue Syntax 3  
GetAttributeValue Syntax 4  
HasAttributes  
SetAttribute  
SetAttributes

## GetAttributeValue Syntax 3

**Description** Returns the string value of the PBDOM\_ATTRIBUTE object (within a PBDOM\_ELEMENT object) with the specified name and within the specified namespace. If no such PBDOM\_ATTRIBUTE exists, the default value is returned.

**Syntax** *pbdom\_element\_name*.GetAttributeValue( string *strAttributeName*, string *strNamespacePrefix*, string *strNamespaceUri*, string *strDefaultValue*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strAttributeName</i>	The name of the attribute whose value is to be returned
<i>strNamespacePrefix</i>	The prefix of the namespace of the PBDOM_ATTRIBUTE whose value is to be returned
<i>strNamespaceUri</i>	The URI of the namespace of the PBDOM_ATTRIBUTE whose value is to be returned
<i>strDefaultValue</i>	Default string value to return if the attribute does not exist

**Return value** String. The string value of the PBDOM\_ATTRIBUTE object specified in *strAttributeName*. If no such object exists, the GetAttributeValue method returns the string provided in *strDefaultValue*.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – If any of the input arguments is invalid, for example, null.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – If there was any memory allocation failure during the execution of this method.

EXCEPTION\_INVALID\_NAME – If the input attribute name or namespace prefix or namespace URI is invalid.

**See also** GetAttribute  
GetAttributeValue Syntax 1  
GetAttributeValue Syntax 2  
GetAttributeValue Syntax 4  
HasAttributes  
SetAttribute  
SetAttributes

## GetAttributeValue Syntax 4

Description	Returns the string value of the PBDOM_ATTRIBUTE object (within a PBDOM_ELEMENT object) with the specified name. If no such PBDOM_ATTRIBUTE exists, the default value is returned.								
Syntax	<i>pbdom_element_name</i> .GetAttributeValue(string <i>strAttributeName</i> , string <i>strDefaultValue</i> )								
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_element_name</i></td> <td>The name of a PBDOM_ELEMENT object</td> </tr> <tr> <td><i>strAttributeName</i></td> <td>The name of the attribute whose value is to be returned</td> </tr> <tr> <td><i>strDefaultValue</i></td> <td>Default string value to return if the attribute does not exist</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object	<i>strAttributeName</i>	The name of the attribute whose value is to be returned	<i>strDefaultValue</i>	Default string value to return if the attribute does not exist
Argument	Description								
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object								
<i>strAttributeName</i>	The name of the attribute whose value is to be returned								
<i>strDefaultValue</i>	Default string value to return if the attribute does not exist								
Return value	String. The string value of the PBDOM_ATTRIBUTE object specified in <i>strAttributeName</i> . If no such object exists, the GetAttributeValue method returns the string provided in <i>strDefaultValue</i> .								
See also	GetAttribute GetAttributeValue Syntax 1 GetAttributeValue Syntax 2 GetAttributeValue Syntax 3 HasAttributes SetAttribute SetAttributes								

## GetChildElement

Description	The GetChildElement method is overloaded: <ul style="list-style-type: none"> <li>Syntax 1 returns the first child PBDOM_ELEMENT object that matches the name indicated by the method parameter.</li> <li>Syntax 2 returns the first child PBDOM_ELEMENT object that matches the name and namespace indicated by the method parameter.</li> </ul>						
Syntax	<table border="1"> <thead> <tr> <th>For this syntax</th> <th>See</th> </tr> </thead> <tbody> <tr> <td>GetChildElement(string <i>strElementName</i>)</td> <td>GetChildElement Syntax 1</td> </tr> <tr> <td>GetChildElement(string <i>strElementName</i>, string <i>strNamespacePrefix</i>, string <i>strNamespaceUri</i>)</td> <td>GetChildElement Syntax 2</td> </tr> </tbody> </table>	For this syntax	See	GetChildElement(string <i>strElementName</i> )	GetChildElement Syntax 1	GetChildElement(string <i>strElementName</i> , string <i>strNamespacePrefix</i> , string <i>strNamespaceUri</i> )	GetChildElement Syntax 2
For this syntax	See						
GetChildElement(string <i>strElementName</i> )	GetChildElement Syntax 1						
GetChildElement(string <i>strElementName</i> , string <i>strNamespacePrefix</i> , string <i>strNamespaceUri</i> )	GetChildElement Syntax 2						

## GetChildElement Syntax 1

**Description** Returns the first child PBDOM\_ELEMENT object, matching the name indicated by the method parameter that is contained in the PBDOM\_ELEMENT object from which the method is invoked.

**Syntax** *pbdom\_element\_name*.GetChildElement(string *strElementName*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strElementName</i>	The local name of the child PBDOM_ELEMENT object to be returned

**Return value** PBDOM\_ELEMENT. The first child PBDOM\_ELEMENT object whose name matches the value of the method parameter. If no PBDOM\_ELEMENT object exists for the specified name, the GetChildElement method returns a value of null.

**See also** GetChildElement Syntax 2  
GetChildElements  
HasChildElements  
HasChildren  
IsRootElement  
RemoveChildElement  
RemoveChildElements

## GetChildElement Syntax 2

**Description** Returns the first child PBDOM\_ELEMENT object, matching the name and namespace indicated by the method parameter contained in the PBDOM\_ELEMENT object from which the method is invoked.

**Syntax** *pbdom\_element\_name*.GetChildElement(string *strElementName*, string *strNamespacePrefix*, string *strNamespaceUri*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strElementName</i>	The local name of the child PBDOM_ELEMENT object to be returned
<i>strNamespacePrefix</i>	The prefix of the namespace of the child PBDOM_ELEMENT object to be returned
<i>strNamespaceUri</i>	The URI of the namespace of the child PBDOM_ELEMENT object to be returned

Return value	PBDOM_ELEMENT. The first child PBDOM_ELEMENT object whose name and namespace information match the values of the method parameters. If no PBDOM_ELEMENT object exists for the specified name and namespace information, the GetChildElement method returns a value of null.
Throws	EXCEPTION_INVALID_ARGUMENT – If any of the input arguments is invalid, for example, null.  EXCEPTION_INVALID_NAME – If the input Element Name or input namespace prefix or namespace URI is invalid.
See also	GetChildElement Syntax 1 GetChildElements HasChildElements HasChildren IsRootElement RemoveChildElement RemoveChildElements

## GetChildElements

Description	The GetChildElements method is overloaded: <ul style="list-style-type: none"> <li>• Syntax 1 retrieves a list of all child PBDOM_ELEMENT objects nested one level deep within a PBDOM_ELEMENT object. The list is stored in the array specified when the method is invoked.</li> <li>• Syntax 2 retrieves a list of all child PBDOM_ELEMENT objects nested one level deep within a PBDOM_ELEMENT object specified by the name provided and belonging to no namespace. The list is stored in the array specified when the method is invoked.</li> <li>• Syntax 3 retrieves a list of all child PBDOM_ELEMENT objects nested one level deep within a PBDOM_ELEMENT object specified by the local name and namespace provided.</li> </ul>
-------------	--

### Syntax

For this syntax	See
GetChildElements(ref pbdom_element pbdom_element_array[])	GetChildElements Syntax 1
GetChildElements(string strElementName, ref pbdom_element pbdom_element_array[])	GetChildElements Syntax 2
GetChildElements(string strElementName, string strNamespacePrefix, string strNamespaceUri, ref pbdom_element pbdom_element_array[])	GetChildElements Syntax 3

## GetChildElements Syntax 1

**Description** Retrieves a list of all child PBDOM\_ELEMENT objects nested one level deep within a PBDOM\_ELEMENT object. The list is stored in the array specified when the method is invoked.

**Syntax** `pbdom_element_name.GetChildElements(ref pbdom_element  
pbdom_element_array)`

<b>Argument</b>	<b>Description</b>
<code><i>pbdom_element_name</i></code>	The name of a PBDOM_ELEMENT object
<code>pbdom_element_array</code>	The array that stores the child PBDOM_ELEMENT objects

**Return value** Boolean. Returns true if child PBDOM\_ELEMENT objects have been collected, and false otherwise.

**Usage** If the PBDOM\_ELEMENT object has no nested elements, GetChildElements returns an empty array.

**See also** GetChildElement  
GetChildElements Syntax 2  
GetChildElements Syntax 3  
HasChildElements  
HasChildren  
IsRootElement  
RemoveChildElement  
RemoveChildElements



## GetChildElements Syntax 2

**Description** Retrieves a list of all child PBDOM\_ELEMENT objects nested one level deep within a PBDOM\_ELEMENT object specified by the name provided and belonging to no namespace. The list is stored in the array specified when the method is invoked.

**Syntax** `pbdom_element_name.GetChildElements(string strElementName, ref pbdom_element pbdom_element_array[])`

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strElementName</i>	The name of the PBDOM_ELEMENT object for which to find children
<i>pbdom_element_array</i>	The array that stores the child PBDOM_ELEMENT objects

**Return value** Boolean. Returns true if child PBDOM\_ELEMENT objects have been collected, and false otherwise.

**Usage** If the PBDOM\_ELEMENT object has no nested elements, GetChildElements returns an empty array.

**See also** GetChildElement  
GetChildElements Syntax 1  
GetChildElements Syntax 3  
HasChildElements  
HasChildren  
IsRootElement  
RemoveChildElement  
RemoveChildElements

## GetChildElements Syntax 3

**Description** Retrieves a list of all child PBDOM\_ELEMENT objects nested one level deep within a PBDOM\_ELEMENT object specified by the local name and namespace provided.

**Syntax** `pbdom_element_name.GetChildElements(string strElementName, string strNamespacePrefix, string strNamespaceUri, ref pbdom_element pbdom_element_array[])`

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strElementName</i>	The name of a PBDOM_ELEMENT object for which to find children
<i>strNamespacePrefix</i>	The prefix of the namespace of the child PBDOM_ELEMENT objects to match
<i>strNamespaceUri</i>	The URI of the namespace of the child PBDOM_ELEMENT objects to match
<i>pbdom_element_array</i> []	The array that stores the child PBDOM_ELEMENT objects

**Return value** Boolean. Returns true if child PBDOM\_ELEMENT objects have been collected, and false otherwise.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – If any of the parameters is invalid.  
EXCEPTION\_INVALID\_NAME – If the input element name or namespace prefix or namespace URI is invalid. The only exception is if the input element name is an empty string.

**Usage** If the PBDOM\_ELEMENT object has no nested elements, GetChildElements returns an empty array.

If the value of *strElementName* is an empty string, then all child elements match.

**See also** GetChildElement  
GetChildElements Syntax 1  
GetChildElements Syntax 2  
HasChildElements  
HasChildren  
IsRootElement  
RemoveChildElement  
RemoveChildElements

## GetContent

**Description** Obtains an array of PBDOM\_OBJECT objects, each of which is a child node of the PBDOM\_ELEMENT object from which the method is invoked. The returned array is “live” in that changes to any item of the array affect the actual item to which the array refers.

**Syntax** `pbdom_element_name.GetContent(ref pbdom_object pbdom_object_array [ ])`

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>pbdom_object_array</i>	The name of an array of PBDOM_OBJECT objects that receive references to the PBDOM_OBJECT objects contained within the PBDOM_ELEMENT object

**Return value** Boolean. Returns true for success and false otherwise.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – If the input array is null.

**Examples** The GetContent method is invoked for the Root> PBDOM\_ELEMENT object in the following XML DOM document:

```
<Root>
  <Element_1>
    <Element_1_1/>
    <Element_1_2/>
    <Element_1_3/>
  </Element_1>
  <Element_2/>
  <Element_3/>
</Root>
```

The GetContent method is invoked from the following PowerScript code:

```
PBDOM_DOCUMENT pbdom_doc
PBDOM_ELEMENT pbdom_elem_root
PBDOM_OBJECT pbdom_obj_array[]

pbdom_elem_root = pbdom_doc.GetRootElement()
pbdom_obj_array = pbdom_elem_root.GetContent(pbdom_obj_array)
```

If the GetContent method returns the value true, the PBDOM\_OBJECT object pbdom\_obj\_array then contains the following content:

Array element	Value
1	<Element_1>
2	<Element_2>
3	<Element_3>

See also                   AddContent Syntax 1  
                               AddContent Syntax 2  
                               InsertContent  
                               RemoveContent  
                               SetContent

## GetName

Description               Retrieves the local name of a PBDOM\_ELEMENT object.

Syntax                    *pbdom\_element\_name*.GetName()

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object

Return value             String. The name of the element as it appears in the XML document but without any namespace prefix.

Examples                 The GetName method returns the string *abc* when it is invoked for the name of the following element:

```
<ns:abc>My Element</ns:abc>
```

Usage                    For an XML element that appears in the form *[namespace\_prefix]:[element\_name]*, the local element name is *element\_name*. When the XML element has no namespace prefix, the local name is simply the element name.

Use the GetQualifiedName method to obtain the fully qualified name of an element (with the namespace prefix).

See also                 GetNamespacePrefix  
                               GetNamespaceUri  
                               RemoveNamespaceDeclaration  
                               SetName

## GetNamespacePrefix

**Description** Returns the namespace prefix for a PBDOM\_ELEMENT object. If no namespace prefix exists for the PBDOM\_ELEMENT object, GetNamespacePrefix returns an empty string.

**Syntax** *pbdom\_element\_name*.GetNamespacePrefix()

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object

**Return value** String. The namespace prefix for the PBDOM\_ELEMENT object.

**See also** AddNamespaceDeclaration  
GetNamespaceUri  
GetQualifiedName  
RemoveNamespaceDeclaration  
SetNamespace

## GetNamespaceUri

**Description** Returns the URI that is mapped to a PBDOM\_ELEMENT object prefix or, if there is no prefix, to the PBDOM\_ELEMENT object default namespace. If no URI is mapped to the PBDOM\_ELEMENT object, GetNameSpaceUri returns an empty string.

**Syntax** *pbdom\_element\_name*.GetNamespaceUri()

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object

**Return value** String. The namespace URI for the PBDOM\_ELEMENT object.

**See also** AddNamespaceDeclaration  
GetNamespacePrefix  
GetQualifiedName  
RemoveNamespaceDeclaration  
SetNamespace

## GetObjectClass

**Description** Returns a long integer code that indicates the class of the current PBDOM\_OBJECT.

**Syntax** `pbdom_object_name.GetObjectClass()`

<b>Argument</b>	<b>Description</b>
<code>pbdom_object_name</code>	The name of a PBDOM_OBJECT object

**Return value** Long. A code that indicates the class of the current PBDOM\_OBJECT. If `pbdom_object_name` is a PBDOM\_ELEMENT object, the returned value is 3.

**Examples** The `GetObjectClass` method returns a value specific to the class of the object from which the method is invoked.

```
PBDOM_OBJECT pbdom_obj

pbdom_obj = Create PBDOM_ELEMENT
MessageBox ("Class", &
    string(pbdom_obj.GetObjectClass()))
```

This example illustrates polymorphism: `pbdom_obj` is declared as `PBDOM_OBJECT` but instantiated as `PBDOM_ELEMENT`. A message box returns the result of the `GetObjectClass` method invoked for `PBDOM_ELEMENT` object. Here the result is 3, indicating that `pbdom_obj` is a `PBDOM_ELEMENT` object.

**Usage** This method can be used for diagnostic purposes to dynamically determine the type of a `PBDOM_OBJECT` at runtime.

## GetObjectClassString

**Description** Returns a string form of the class of the PBDOM\_OBJECT.

**Syntax** `pbdom_object_name.GetObjectClassString()`

Argument	Description
<i>pbdom_object_name</i>	The name of your PBDOM_OBJECT object

**Return value** String. A string that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_ELEMENT object, the returned string is "pbdom\_element".

**Examples** The GetObjectClass method returns a string specific to the class of the object from which the method is invoked.

```
PBDOM_OBJECT pbdom_obj

pbdom_obj = Create PBDOM_ELEMENT
MessageBox ("Class", pbdom_obj.GetObjectClassString())
```

This example illustrates polymorphism: *pbdom\_obj* is declared as PBDOM\_OBJECT but instantiated as PBDOM\_ELEMENT object. A message box returns the result of the GetObjectClassString method invoked for PBDOM\_ELEMENT object. Here the result is *pbdom\_element*, indicating that *pbdom\_obj* is a PBDOM\_ELEMENT object.

**Usage** This method can be used for diagnostic purposes to dynamically determine the actual type of a PBDOM\_OBJECT at runtime.

## GetOwnerDocumentObject

**Description** Returns the PBDOM\_DOCUMENT object that owns the PBDOM\_ELEMENT object.

**Syntax** `pbdom_element_name.GetOwnerDocumentObject()`

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object

**Return value** PBDOM\_DOCUMENT. The PBDOM\_DOCUMENT that owns the PBDOM\_ELEMENT object from which the GetOwnerDocumentObject method is invoked. A return value of null indicates that the PBDOM\_ELEMENT object is not owned by any PBDOM\_DOCUMENT.

Examples

The GetOwnerDocumentObject method is invoked from the following PowerScript code, where pbdom\_root\_elem refers to the root element of the PBDOM\_DOCUMENT object pbdom\_doc:

```
PBDOM_DOCUMENT pbdom_doc
PBDOM_ELEMENT pbdom_root_elem

pbdom_root_elem = pbdom_doc.GetRootElement()

IF
    pbdom_doc.Equals &
        (pbdom_root_elem.GetOwnerDocumentObject())
THEN
    MessageBox ("Equals", "The objects are equal")
END IF
```

The Equals method tests for equality between pbdom\_doc and the PBDOM\_DOCUMENT object returned from the GetOwnerDocumentObject method. A message box reports that the objects are equal.

See also

GetParentObject  
SetParentObject

## GetParentObject

Description

Returns the parent object for the PBDOM\_ELEMENT object.

Syntax

*pbdom\_element\_name*.GetParentObject()

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object

Return value

PBDOM\_OBJECT. The parent object of the PBDOM\_ELEMENT object from which the GetParentObject method is invoked. A return value of null indicates the PBDOM\_ELEMENT object has no parent.

See also

GetOwnerDocumentObject  
SetParentObject



## GetQualifiedName

Description	Returns the full name of a PBDOM_ELEMENT object in the form <code>[namespace_prefix]:[local_name]</code> . If there is no namespace prefix for the PBDOM_ELEMENT object, the GetQualifiedName method returns the local name.				
Syntax	<code>pbdom_element_name.GetQualifiedName()</code>				
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>pbdom_element_name</code></td> <td>The name of a PBDOM_ELEMENT object</td> </tr> </tbody> </table>	Argument	Description	<code>pbdom_element_name</code>	The name of a PBDOM_ELEMENT object
Argument	Description				
<code>pbdom_element_name</code>	The name of a PBDOM_ELEMENT object				
Return value	String. The full name of the PBDOM_ELEMENT object. The full name consists of both a namespace prefix and a local name.				
See also	AddNamespaceDeclaration GetNamespacePrefix GetNamespaceUri RemoveNamespaceDeclaration SetNamespace				

## GetText

Description	Obtains a concatenation of the text values of all the PBDOM_TEXT and PBDOM_CDATA nodes contained within the PBDOM_ELEMENT object from which the method is invoked.				
Syntax	<code>pbdom_element_name.GetText()</code>				
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>pbdom_element_name</code></td> <td>The name of a PBDOM_ELEMENT object</td> </tr> </tbody> </table>	Argument	Description	<code>pbdom_element_name</code>	The name of a PBDOM_ELEMENT object
Argument	Description				
<code>pbdom_element_name</code>	The name of a PBDOM_ELEMENT object				
Return value	String				
Examples	<p>The GetText method is invoked for the abc PBDOM_ELEMENT object:</p> <pre>&lt;abc&gt;Root Element Data&lt;data&gt;ABC Data &lt;/data&gt; now with extra info&lt;/abc&gt;</pre> <p>The GetText method returns the following string:</p> <pre>Root Element Data now with extra info</pre> <p>The text “ABC Data” is excluded because it is not contained within the PBDOM_ELEMENT abc.</p>				
See also	GetTextNormalize GetTextTrim, SetText				

## GetTextNormalize

**Description** Returns the normalized text data contained in a PBDOM\_ELEMENT object.

**Syntax** `pbdom_element_name.GetTextNormalize()`

Argument	Description
<code>pbdom_element_name</code>	The name of a PBDOM_ELEMENT object

**Return value** String

**Examples** The GetTextNormalize method is invoked for the abc element of the following XML:

```
<abc>      Root      Element      Data      <data>ABC
Data </data> now with extra info      </abc>
```

The GetTextNormalize method returns the following string:

```
Root Element Data now with extra info
```

**Usage** The text data returned includes any text data contained in PBDOM\_CDATA objects. All surrounding whitespace characters are removed. Internal whitespace characters are normalized to a single space. The GetTextNormalize method returns an empty string if no text values exist for the PBDOM\_ELEMENT object or if there are only whitespace characters.

**See also** GetText  
GetTextTrim  
SetText

## GetTextTrim

**Description** Returns the text data contained within a PBDOM\_ELEMENT object with any leading and trailing whitespace characters removed.

**Syntax** `pbdom_element_name.GetTextTrim()`

Argument	Description
<code>pbdom_element_name</code>	The name of a PBDOM_ELEMENT object

**Return value** String

**Examples** The GetTextTrim method is invoked for the abc element of the following XML:

```
<abc>      Root      Element Data <![CDATA[
with      some cdata text      ]]></abc>
```

The `GetTextTrim` method returns the following string:

```
Root Element Data with some cdata text
```

Usage	Surrounding whitespace characters are removed from the returned text data. The <code>GetTextTrim</code> method returns an empty string if no text value exists for the <code>PBDOM_ELEMENT</code> object or if the text value contains only whitespace characters.
See also	<code>GetText</code> <code>GetTextNormalize</code> <code>SetText</code>

## HasAttributes

**Description** Indicates whether a `PBDOM_ELEMENT` object has one or more attributes.

**Syntax** `pbdom_element_name.HasAttributes()`

Argument	Description
<code>pbdom_element_name</code>	The name of a <code>PBDOM_ELEMENT</code> object

**Return value** Boolean. Returns true if this `PBDOM_ELEMENT` object has at least one attribute and false if this `PBDOM_ELEMENT` object has no attributes.

**Examples** In the following document fragment, only the element `site` has an attribute (`href`):

```
<books>
  <title>Inside Wizardry</title>
  <author>Ron Potter</author>
  <site href="http://www.mybooks.com/press"/>
</books>
```

If the `PBDOM_ELEMENT` object `pbdom_elem_site` represents the element `site`, the following call returns true:

```
pbdom_elem_site.HasAttributes()
```

See also	<code>GetAttribute</code> <code>GetAttributes</code> <code>GetAttributeValue</code> <code>SetAttribute</code> <code>SetAttributes</code>
----------	--

## HasChildElements

**Description** Indicates whether a PBDOM\_ELEMENT object has one or more child PBDOM\_ELEMENT objects.

**Syntax** *pbdom\_element\_name*.HasChildElements()

<b>Argument</b>	<b>Description</b>
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object

**Return value** Boolean. Returns true if this PBDOM\_ELEMENT object has at least one child PBDOM\_ELEMENT object and false if this PBDOM\_ELEMENT object has no child PBDOM\_ELEMENT objects.

**Examples** The HasChildElements method is invoked for the books PBDOM\_ELEMENT object in the following XML fragment:

```
<books>
  <title>Inside OLE</title>
  <author>Kraig Brockschmidt</author>
  <site href="http://www.microsoft.com/press"/>
</books>
```

The books object has three child PBDOM\_ELEMENT objects: title, author, and site. The HasChildElements method returns true.

**See also** GetChildElement  
GetChildElements  
HasChildren  
IsRootElement  
RemoveChildElement  
RemoveChildElements

## HasChildren

**Description** Indicates whether a PBDOM\_ELEMENT object has one or more child objects.

**Syntax** `pbdom_element_name.HasChildren()`

Argument	Description
<code>pbdom_element_name</code>	The name of a PBDOM_ELEMENT object

**Return value** Boolean. Returns true if this PBDOM\_ELEMENT object has at least one child object and false if this PBDOM\_ELEMENT object has no child objects.

**Examples** The HasChildren method is invoked for elements in the following XML fragment:

```
<books>
  <title>Inside OLE</title>
  <author>Kraig Brockschmidt</author>
  <site href="http://www.microsoft.com/press"/>
</books>
```

The books element has three child elements: title, author, and site. The title and author elements each have a child PBDOM\_TEXT object. The HasChildren method returns a value of true when invoked for these elements.

In contrast, the site element has a PBDOM\_ATTRIBUTE href, which is not considered a child PBDOM\_OBJECT. The HasChildren method returns a value of False when invoked for the site element.

**Usage** PBDOM's implementation of the HasChildren method differs from JDOM's implementation in that the JDOM HasChildren method returns true only if an Element contains child Elements. Text and other types of objects do not count.

PBDOM provides an alternative method, HasChildElements, to specifically detect whether a PBDOM\_ELEMENT object has at least one child PBDOM\_ELEMENT object.

**See also** HasChildElements  
IsRootElement

## InsertContent

Description Inserts a new PBDOM\_OBJECT into a PBDOM\_ELEMENT object.

Syntax *pbdom\_element\_name*.InsertContent(*pbdom\_object pbdom\_object\_new*,  
*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>pbdom_object_new</i>	The PBDOM_OBJECT to insert
<i>pbdom_object_ref</i>	A positional reference PBDOM_OBJECT in front of which the new PBDOM_OBJECT is to be inserted

Return value PBDOM\_OBJECT. The PBDOM\_ELEMENT object modified and returned as a PBDOM\_OBJECT.

Throws EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If an invalid PBDOM\_OBJECT is added. See AddContent on page 224 for the valid PBDOM\_OBJECT objects that can be added to a PBDOM\_ELEMENT object. This exception is also thrown if the input PBDOM\_OBJECT or the reference PBDOM\_OBJECT is this PBDOM\_ELEMENT object itself.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT to insert has not been given a user-defined name. The same exception is also thrown if the reference PBDOM\_OBJECT is also not given a user-defined name, unless the reference PBDOM\_OBJECT is specifically set to null.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT to insert is not associated with a derived PBDOM\_OBJECT. The same exception is also thrown if the reference PBDOM\_OBJECT is also not associated with a derived PBDOM\_OBJECT unless the reference PBDOM\_OBJECT is specifically set to null.

EXCEPTION\_INVALID\_ARGUMENT – If the reference PBDOM\_OBJECT (second parameter) is intended to be null but is not specifically set to null using the SetNull method.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – If the input PBDOM\_OBJECT to insert already has a parent.

EXCEPTION\_WRONG\_PARENT\_ERROR – If the reference PBDOM\_OBJECT is not a child of this PBDOM\_ELEMENT object.

EXCEPTION\_HIERARCHY\_ERROR – If inserting the input PBDOM\_OBJECT will cause the current PBDOM\_ELEMENT object to be no longer well formed.

## Examples

The following PowerScript code is used to create an XML document:

```

pbdom_doc1 = Create PBDOM_DOCUMENT
pbdom_elem_1 = Create PBDOM_ELEMENT
pbdom_elem_2 = Create PBDOM_ELEMENT
pbdom_elem_3 = Create PBDOM_ELEMENT

pbdom_elem_1.SetName ("pbdom_elem_1")
pbdom_elem_2.SetName ("pbdom_elem_2")
pbdom_elem_3.SetName ("pbdom_elem_3")

pbdom_doc1.NewDocument ("", "", "Root_Element", "", "")
pbdom_elem_root = pbdom_doc1.GetRootElement()
pbdom_elem_root.AddContent (pbdom_elem_1)
pbdom_elem_root.AddContent (pbdom_elem_3)

```

The following XML results:

```

!DOCTYPE Root_Element>
<Root_Element>
  <pbdom_elem_1 />
  <pbdom_elem_3 />
</Root_Element>

```

The InsertContent method is used to add an element between pbdom\_elem\_1 and pbdom\_elem\_3:

```

pbdom_elem_root.InsertContent (pbdom_elem_2, &
  pbdom_elem_3)

```

The following XML results:

```

<!DOCTYPE Root_Element>
<Root_Element>
  <pbdom_elem_1 />
  <pbdom_elem_2 />
  <pbdom_elem_3 />
</Root_Element>

```

## Usage

The inserted object becomes a child of the PBDOM\_ELEMENT object. The new PBDOM\_OBJECT is positioned before another PBDOM\_OBJECT, which is specified in the second of two parameters.

## See also

AddContent Syntax 1  
 AddContent Syntax 2  
 GetContent  
 RemoveContent  
 SetContent

## IsAncestorObjectOf

**Description** Determines whether a PBDOM\_ELEMENT object is the ancestor of the PBDOM\_OBJECT indicated by the method parameter.

**Syntax** *pbdom\_element\_name*.IsAncestorObjectOf(*pbdom\_object*  
*pbdom\_object\_ref*)

<b>Argument</b>	<b>Description</b>
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to be tested for equality with this PBDOM_ELEMENT object

**Return value** Boolean. Returns true if this PBDOM\_ELEMENT object is the ancestor of the specified PBDOM\_OBJECT, and false otherwise.

## IsRootElement

**Description** Indicates whether a PBDOM\_ELEMENT object is the root element of a PBDOM\_DOCUMENT object.

**Syntax** *pbdom\_element\_name*.IsRootElement()

<b>Argument</b>	<b>Description</b>
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object

**Return value** Boolean. Returns true if this PBDOM\_ELEMENT object is the root element of a PBDOM\_DOCUMENT, and false otherwise.

**See also** GetChildElement  
GetChildElements  
HasChildElements  
HasChildren  
RemoveChildElement  
RemoveChildElements



## RemoveAttribute

- Description
- The RemoveAttribute method is overloaded:
- Syntax 1 removes a PBDOM\_ATTRIBUTE from its owner PBDOM\_ELEMENT object using a reference to the PBDOM\_ATTRIBUTE.
  - Syntax 2 removes a PBDOM\_ATTRIBUTE from its owner PBDOM\_ELEMENT object using the name of the PBDOM\_ATTRIBUTE.
  - Syntax 3 removes a PBDOM\_ATTRIBUTE from its owner PBDOM\_ELEMENT object using the name and namespace of the PBDOM\_ATTRIBUTE.

### Syntax

For this syntax	See
RemoveAttribute( <i>pbdom_attribute</i> <i>pbdom_attribute_ref</i> )	RemoveAttribute Syntax 1
RemoveAttribute(string <i>strAttributeName</i> )	RemoveAttribute Syntax 2
RemoveAttribute(string <i>strAttributeName</i> , string <i>strNamespacePrefix</i> , string <i>strNamespaceUri</i> )	RemoveAttribute Syntax 3

## RemoveAttribute Syntax 1

Description

Removes a PBDOM\_ATTRIBUTE from its owner PBDOM\_ELEMENT object.

Syntax

*pbdom\_element\_name*.RemoveAttribute(*pbdom\_attribute*  
*pbdom\_attribute\_ref*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>pbdom_attribute_ref</i>	The PBDOM_ATTRIBUTE object to remove from this PBDOM_ELEMENT object

Return value

Boolean. Returns true if the specified PBDOM\_ATTRIBUTE was removed, and false otherwise.

## RemoveAttribute Syntax 2

**Description** Removes a PBDOM\_ATTRIBUTE specified by the name provided that is not contained in a namespace. If no such PBDOM\_ATTRIBUTE exists, RemoveAttribute does nothing.

**Syntax** `pbdom_element_name.RemoveAttribute(string strAttributeName)`

Argument	Description
<code>pbdom_element_name</code>	The name of a PBDOM_ELEMENT object
<code>strAttributeName</code>	The name of the PBDOM_ATTRIBUTE object to remove

**Return value** Boolean. Returns true if the specified PBDOM\_ATTRIBUTE was removed, and false otherwise.

## RemoveAttribute Syntax 3

**Description** Removes a PBDOM\_ATTRIBUTE specified by the name and namespace provided. If no such PBDOM\_ATTRIBUTE exists, RemoveAttribute does nothing.

**Syntax** `pbdom_element_name.RemoveAttribute(string strAttributeName, string strNamespacePrefix, string strNamespaceUri)`

Argument	Description
<code>pbdom_element_name</code>	The name of a PBDOM_ELEMENT object
<code>strAttributeName</code>	The name of the PBDOM_ATTRIBUTE object to remove
<code>strNamespacePrefix</code>	Prefix of the namespace of the PBDOM_ATTRIBUTE to remove
<code>strNamespaceUri</code>	URI of the namespace of the PBDOM_ATTRIBUTE to remove

**Return value** Boolean. Returns true if the specified PBDOM\_ATTRIBUTE was removed, and false otherwise.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – If any of the input parameters is invalid, for example, null.

EXCEPTION\_INVALID\_STRING – If the input Attribute Name is invalid (for example, contains a colon), or if the namespace prefix or URI is invalid.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – If a memory allocation failure occurred during the execution of this method.

## RemoveChildElement

- Description                      The RemoveChildElement method is overloaded:
- Syntax 1 removes the first child PBDOM\_ELEMENT object (one level deep) that has the local name provided and belongs to no namespace.
  - Syntax 2 removes the first child PBDOM\_ELEMENT object (one level deep) that has the local name provided and belongs to the specified namespace.

### Syntax

For this syntax	See
RemoveChildElement(string <i>strElementName</i> )	RemoveChildElement Syntax 1
RemoveChildElement(string <i>strElementName</i> , string <i>strNamespacePrefix</i> , string <i>strNamespaceUri</i> )	RemoveChildElement Syntax 2

## RemoveChildElement Syntax 1

- Description                      Removes the first child PBDOM\_ELEMENT object (one level deep) that has the local name provided and belongs to no namespace.

Syntax                              *pbdom\_element\_name*.RemoveChildElement(string *strElementName*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strElementName</i>	The name of the child PBDOM_ELEMENT object to remove

Return value                      Boolean. Returns true if the specified PBDOM\_ELEMENT object was removed, and false otherwise.

Throws                              EXCEPTION\_INVALID\_ARGUMENT – If the input parameter is invalid, for example, null.

EXCEPTION\_INVALID\_STRING – If the input element name is invalid.

See also                            GetChildElement  
 GetChildElements  
 HasChildElements  
 HasChildren  
 IsRootElement  
 RemoveChildElement Syntax 2  
 RemoveChildElements

## RemoveChildElement Syntax 2

**Description** Removes the first child PBDOM\_ELEMENT object (one level deep) that has the local name provided and belongs to the specified namespace.

**Syntax** *pbdom\_element\_name*.RemoveChildElement(string *strElementName*, string *strNamespacePrefix*, string *strNamespaceUri*)

<b>Argument</b>	<b>Description</b>
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strElementName</i>	The name of the PBDOM_ELEMENT object to remove
<i>strNamespacePrefix</i>	Prefix of the namespace of the PBDOM_ELEMENT object to remove
<i>strNamespaceUri</i>	URI of the namespace of the PBDOM_ATTRIBUTE to remove

**Return value** Boolean. Returns true if the specified PBDOM\_ELEMENT object was removed and false otherwise.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – If the input parameter is invalid, for example, null.

EXCEPTION\_INVALID\_STRING – If the input element name is invalid or the input namespace prefix or URI is invalid.

**See also** GetChildElement  
GetChildElements  
HasChildElements  
HasChildren  
IsRootElement  
RemoveChildElement Syntax 1  
RemoveChildElements

## RemoveChildElements

- Description
- The RemoveChildElements method is overloaded:
- Syntax 1 method removes from the current PBDOM\_ELEMENT object all child PBDOM\_ELEMENT objects. It uses no parameters.
  - Syntax 2 method removes from the current PBDOM\_ELEMENT object all child PBDOM\_ELEMENT objects that have the specified local name and belong to no namespace.
  - Syntax 3 removes from the current PBDOM\_ELEMENT object all child PBDOM\_ELEMENT objects (one level deep) that have the specified local name and belong to the specified namespace.

### Syntax

For this syntax	See
RemoveChildElements()	RemoveChildElements Syntax 1
RemoveChildElements(string <i>strElementName</i> )	RemoveChildElements Syntax 2
RemoveChildElements(string <i>strElementName</i> , string <i>strNamespacePrefix</i> , string <i>strNamespaceUri</i> )	RemoveChildElements Syntax 3

## RemoveChildElements Syntax 1

- Description
- Removes from the current PBDOM\_ELEMENT object all child PBDOM\_ELEMENT objects. It uses no parameters.
- Syntax
- ```
pbdom_element_name.RemoveChildElements()
```
- | Argument                  | Description                        |
|---------------------------|------------------------------------|
| <i>pbdom_element_name</i> | The name of a PBDOM_ELEMENT object |
- Return value
- Boolean. Returns true if any child PBDOM\_ELEMENT object was removed and false otherwise.
- See also
- GetChildElement
  - GetChildElements
  - HasChildElements
  - HasChildren
  - IsRootElement
  - RemoveChildElement
  - RemoveChildElements Syntax 2
  - RemoveChildElements Syntax 3

## RemoveChildElements Syntax 2

**Description** Removes from the current PBDOM\_ELEMENT object all child PBDOM\_ELEMENT objects that have the specified local name and belong to no namespace.

**Syntax** `pbdom_element_name.RemoveChildElements(string strElementName)`

| Argument                  | Description                                           |
|---------------------------|-------------------------------------------------------|
| <i>pbdom_element_name</i> | The name of a PBDOM_ELEMENT object                    |
| <i>strElementName</i>     | The name of the child PBDOM_ELEMENT objects to remove |

**Return value** Boolean. Returns true if any child PBDOM\_ELEMENT object was removed, and false otherwise.

**See also** GetChildElement  
GetChildElements  
HasChildElements  
HasChildren  
IsRootElement  
RemoveChildElement  
RemoveChildElements Syntax 1  
RemoveChildElements Syntax 3

## RemoveChildElements Syntax 3

**Description** Removes from the current PBDOM\_ELEMENT object all child PBDOM\_ELEMENT objects (one level deep) that have the specified local name and belong to the specified namespace.

**Syntax** `pbdom_element_name.RemoveChildElements(string strElementName, string strNamespacePrefix, string strNamespaceUri)`

| Argument                  | Description                                                          |
|---------------------------|----------------------------------------------------------------------|
| <i>pbdom_element_name</i> | The name of a PBDOM_ELEMENT object                                   |
| <i>strElementName</i>     | The name of the child PBDOM_ELEMENT objects to remove                |
| <i>strNamespacePrefix</i> | Prefix of the namespace of the child PBDOM_ELEMENT objects to remove |
| <i>strNamespaceUri</i>    | URI of the namespace of the child PBDOM_ATTRIBUTE objects to remove  |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Return value | Boolean. Returns true if any child PBDOM_ELEMENT object was removed and false otherwise.                                                                                                                                                                                                                                                                                                                                                   |
| Throws       | <p>EXCEPTION_INVALID_ARGUMENT – If any of the input parameters is invalid, for example, null.</p> <p>EXCEPTION_INVALID_NAME – If the input element name or namespace prefix or URI is invalid. The only exception is if the input element name is an empty string, in which case all element names match.</p> <p>EXCEPTION_MEMORY_ALLOCATION_FAILURE – If there was any memory allocation failure during the execution of this method.</p> |
| See also     | <p>GetChildElement</p> <p>GetChildElements</p> <p>HasChildElements</p> <p>HasChildren</p> <p>IsRootElement</p> <p>RemoveChildElement</p> <p>RemoveChildElements Syntax 1</p> <p>RemoveChildElements Syntax 2</p>                                                                                                                                                                                                                           |

## RemoveContent

| Description               | Removes a child PBDOM_OBJECT from a PBDOM_ELEMENT object. All children of the removed PBDOM_OBJECT are also removed.                                                                                                                                                                                                                                                                                                                                                                        |          |             |                           |                                    |                         |                            |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|-------------|---------------------------|------------------------------------|-------------------------|----------------------------|
| Syntax                    | <i>pbdom_element_name</i> .RemoveContent( <i>pbdom_object pbdom_object_ref</i> )                                                                                                                                                                                                                                                                                                                                                                                                            |          |             |                           |                                    |                         |                            |
|                           | <table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_element_name</i></td> <td>The name of a PBDOM_ELEMENT object</td> </tr> <tr> <td><i>pbdom_object_ref</i></td> <td>The PBDOM_OBJECT to remove</td> </tr> </tbody> </table>                                                                                                                                                                                                   | Argument | Description | <i>pbdom_element_name</i> | The name of a PBDOM_ELEMENT object | <i>pbdom_object_ref</i> | The PBDOM_OBJECT to remove |
| Argument                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |          |             |                           |                                    |                         |                            |
| <i>pbdom_element_name</i> | The name of a PBDOM_ELEMENT object                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |             |                           |                                    |                         |                            |
| <i>pbdom_object_ref</i>   | The PBDOM_OBJECT to remove                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |          |             |                           |                                    |                         |                            |
| Return value              | Boolean. Returns true if the specified content was removed and false otherwise.                                                                                                                                                                                                                                                                                                                                                                                                             |          |             |                           |                                    |                         |                            |
| Throws                    | <p>EXCEPTION_USE_OF_UNNAMED_PBDOM_OBJECT – If the input PBDOM_OBJECT has not been given a user-defined name.</p> <p>EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – If the input PBDOM_OBJECT is not associated with a derived PBDOM_OBJECT.</p> <p>EXCEPTION_WRONG_DOCUMENT_ERROR – If the input PBDOM_OBJECT is not from the same document as this PBDOM_ELEMENT object.</p> <p>EXCEPTION_WRONG_PARENT_ERROR – If the input PBDOM_OBJECT is not a child of the current PBDOM_ELEMENT object.</p> |          |             |                           |                                    |                         |                            |

Examples

The RemoveContent method is used to modify the following XML fragment:

```
<Telephone_Book>
  <Entry>
    <Particulars>
      <Name>John Doe</Name>
      <Age>21</Age>
      <Phone_Number>1234567</Phone_Number>
    </Particulars>
  </Entry>
</Telephone_Book>
```

The RemoveContent method is invoked from the following PowerScript code:

```
PBDOM_DOCUMENT pbdom_doc
PBDOM_ELEMENT pbdom_entry

pbdom_doc.GetRootElement().RemoveContent(pbdom_entry)
```

The following XML results:

```
<Telephone_Book></Telephone_Book>
```

See also

- AddContent Syntax 1
- AddContent Syntax 2
- GetContent
- InsertContent
- SetContent

## RemoveNamespaceDeclaration

Description

Removes the specified PBDOM\_NAMESPACE declaration for a PBDOM\_ELEMENT object. If the namespace prefix is an empty string, RemoveNamespaceDeclaration removes a default namespace declaration.

Syntax

```
pbdom_element_name.RemoveNamespaceDeclaration(string strNamespacePrefix, string strNamespaceUri)
```

| Argument                  | Description                                   |
|---------------------------|-----------------------------------------------|
| <i>pbdom_element_name</i> | The name of a PBDOM_ELEMENT object            |
| <i>strNamespacePrefix</i> | Prefix of the namespace declaration to remove |
| <i>strNamespaceUri</i>    | URI of the namespace declaration to remove    |

Return value

Boolean. Returns true if the namespace has been removed from the PBDOM\_ELEMENT object, and false otherwise.



|          |                                                                                                                                                                                                                                                                                                                                                                      |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Throws   | <p>EXCEPTION_INVALID_ARGUMENT – If any of the input parameters is invalid, for example, null.</p> <p>EXCEPTION_INVALID_NAME – If the namespace prefix or URI is invalid, or both the namespace prefix and URI are invalid as a pair.</p> <p>EXCEPTION_MEMORY_ALLOCATION_FAILURE – If any memory allocation failure occurred during the execution of this method.</p> |
| See also | <p>AddNamespaceDeclaration</p> <p>GetNamespacePrefix</p> <p>GetNamespaceUri</p> <p>GetQualifiedName</p> <p>SetNamespace</p>                                                                                                                                                                                                                                          |

## SetAttribute

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | <p>The SetAttribute method is overloaded:</p> <ul style="list-style-type: none"> <li>• Syntax 1 adds a predefined PBDOM_ATTRIBUTE object to a PBDOM_ELEMENT object.</li> <li>• Syntax 2 adds a PBDOM_ATTRIBUTE object and its value to a PBDOM_ELEMENT object using strings for the name and value of the PBDOM_ATTRIBUTE.</li> <li>• Syntax 3 adds an attribute/value pair to a PBDOM_ELEMENT object using strings for the name and value of the PBDOM_ATTRIBUTE, and the prefix and URI of the namespace to which the PBDOM_ATTRIBUTE belongs.</li> </ul> |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Syntax

| For this syntax                                                                                                                                                    | See                   |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| SetAttribute( <i>pbdom_attribute pbdom_attribute_ref</i> )                                                                                                         | SetAttribute Syntax 1 |
| SetAttribute(string <i>strName</i> , string <i>strValue</i> )                                                                                                      | SetAttribute Syntax 2 |
| SetAttribute(string <i>strName</i> , string <i>strValue</i> , string <i>strNamespacePrefix</i> , string <i>strNamespaceUri</i> , boolean <i>bVerifyNamespace</i> ) | SetAttribute Syntax 3 |

## SetAttribute Syntax 1

**Description** Adds a predefined PBDOM\_ATTRIBUTE object to a PBDOM\_ELEMENT object. Any existing attribute with the same name and namespace URI is overwritten.

**Syntax** `pbdom_element_name.SetAttribute(pbdom_attribute pbdom_attribute_ref)`

| Argument                   | Description                                                        |
|----------------------------|--------------------------------------------------------------------|
| <i>pbdom_element_name</i>  | The name of a PBDOM_ELEMENT object                                 |
| <i>pbdom_attribute_ref</i> | The PBDOM_ATTRIBUTE object to be set for this PBDOM_ELEMENT object |

**Return value** PBDOM\_ELEMENT. The PBDOM\_ELEMENT object modified to contain the specified PBDOM\_ATTRIBUTE.

**Throws** EXCEPTION\_INVALID\_ARGUMENT – The input PBDOM\_ATTRIBUTE is invalid. This can happen if it has not been initialized properly or it is a null object reference.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – The input PBDOM\_ATTRIBUTE has not been given a user-defined name.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_OWNER – The input PBDOM\_ATTRIBUTE already has an owner element.

**Examples** **Example 1** The SetAttribute method is invoked for the following element:

```
<image></image>
```

The SetAttribute method is invoked from the following PowerScript code, where `elem_image` represents the image element from the preceding XML:

```
attr_src.SetName("src")
attr_src.SetValue("logo.gif")
elem_image.SetAttribute(attr_src)
```

The following XML results:

```
<image src="logo.gif"></image>
```

**Example 2** The following example demonstrates the impact of setting a PBDOM\_ATTRIBUTE for a PBDOM\_ELEMENT object where the PBDOM\_ELEMENT object already contains an attribute of the same name and namespace URI as the input PBDOM\_ATTRIBUTE.

The example creates a PBDOM\_DOCUMENT based on the following document:

```
<root xmlns:pre1="http://www.pre.com"
xmlns:pre2="http://www.pre.com">
  <child1 pre1:a="123"/>
</root>
```

Then it creates a PBDOM\_ATTRIBUTE object and sets its name to `a` and its prefix and URI to `pre2` and `http://www.pre.com`. The `bVerifyNamespace` argument is set to `false` because this PBDOM\_ATTRIBUTE has not been assigned an owner PBDOM\_ELEMENT object yet, so that the verification for a predeclared namespace would fail. The text value is set to `456`.

The `child1` element already contains an attribute named `a` that belongs to the namespace `http://www.pre.com`, as indicated by the prefix `pre1`. The new PBDOM\_ATTRIBUTE uses the prefix `pre2`, but it represents the same namespace URI, so setting the new PBDOM\_ATTRIBUTE to `child1` successfully replaces the existing `pre1:a` with the new PBDOM\_ATTRIBUTE `pre2:a`.

```
PBDOM_BUILDER pbdom_buildr
PBDOM_DOCUMENT pbdom_doc
PBDOM_ATTRIBUTE pbdom_attr
string strXML = "<root
xmlns:pre1=~\"http://www.pre.com~\"
xmlns:pre2=~\"http://www.pre.com~\"><child1
pre1:a=~\"123~\"/></root>"

try
  pbdom_buildr = Create PBDOM_BUILDER
  pbdom_doc = pbdom_buildr.BuildFromString (strXML)

  // Create a PBDOM_ATTRIBUTE and set its properties
  pbdom_attr = Create PBDOM_ATTRIBUTE
  pbdom_attr.SetName ("a")
  pbdom_attr.SetNamespace ("pre2", &
    "http://www.pre.com", false)
  pbdom_attr.SetText ("456")

  // Attempt to obtain the child1 element and
  // set the new attribute to it
  pbdom_doc.GetRootElement(). &
    GetChildElement("child1").SetAttribute(pbdom_attr)

  pbdom_doc.SaveDocument &
    ("pbdom_elem_set_attribute_1.xml")
```

```
catch (PBDOM_EXCEPTION except)
    MessageBox ("PBDOM_EXCEPTION", except.GetMessage())
end try
```

When saved and converted to an XML document, the document looks like the following :

```
<root xmlns:pre1="http://www.pre.com"
xmlns:pre2="http://www.pre.com"
    <child1 pre2:a="456"/
</root
```

### Usage

This method allows the caller to add a predefined PBDOM\_ATTRIBUTE object to a PBDOM\_ELEMENT object. If this PBDOM\_ELEMENT object already contains an existing attribute with the same name and namespace URI as the input PBDOM\_ATTRIBUTE, the existing attribute is replaced by the input PBDOM\_ATTRIBUTE.

If a PBDOM\_ATTRIBUTE has been created to represent the original attribute, it is still valid after the call, but the attribute that it represents has been detached from the original owner element. Calling GetOwnerElementObject on this PBDOM\_ATTRIBUTE returns a null value.

### See also

- GetAttribute
- GetAttributes
- GetAttributeValue
- HasAttributes
- SetAttribute Syntax 2
- SetAttribute Syntax 3
- SetAttributes

## SetAttribute Syntax 2

**Description** Adds a PBDOM\_ATTRIBUTE object and its value to a PBDOM\_ELEMENT object. Any existing attribute with the same name and namespace URI is overwritten.

**Syntax** `pbdom_element_name.SetAttribute(string strName, string strValue)`

| Argument                        | Description                                  |
|---------------------------------|----------------------------------------------|
| <code>pbdom_element_name</code> | The name of a PBDOM_ELEMENT object           |
| <code>strName</code>            | The name of the PBDOM_ATTRIBUTE to be added  |
| <code>strValue</code>           | The value of the PBDOM_ATTRIBUTE to be added |

**Return value** PBDOM\_ELEMENT. The PBDOM\_ELEMENT object modified to contain the specified PBDOM\_ATTRIBUTE with the specified value.

**Throws**

EXCEPTION\_INVALID\_ARGUMENT – One or both of the input strings are invalid. This can happen if either or both strings have not been initialized properly or are null.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_ELEMENT object's internal implementation is null. The occurrence of this exception is rare but can take place if severe memory corruption occurs.

EXCEPTION\_INVALID\_NAME – An invalid name for the attribute is supplied.

EXCEPTION\_INVALID\_STRING – An invalid string for the attribute value is supplied.

**Examples** **Example 1** The SetAttribute method is invoked for the following XML element:

```
<code>0789725045</code>
```

The SetAttribute method is invoked from the following PowerScript statement, where `elem_code` represents the code element:

```
elem_code.SetAttribute("type", "ISBN")
```

The following XML element results:

```
<code type="ISBN">0789725045</code>
```

**Example 2** The following example demonstrates the effect of setting an attribute for a PBDOM\_ELEMENT object when the PBDOM\_ELEMENT object already contains an attribute of the same name. The example creates a PBDOM\_DOCUMENT based on the following document:

```
<root xmlns:pre1="http://www.pre.com">
  <child1 pre1:a="123" b="456"/>
</root>
```

The child1 element already contains an attribute named b with value 456. Calling the SetAttribute method with name b and value 789 creates a new attribute for child1 that replaces the original b attribute.

```
PBDOM_BUILDER      pbdom_buildr
PBDOM_DOCUMENT     pbdom_doc
string strXML = "<root
xmlns:pre1=~"http://www.pre.com~" ><child1
pre1:a=~"123~" b=~"456~"/></root>"

try
  pbdom_buildr = Create PBDOM_BUILDER
  pbdom_doc = pbdom_buildr.BuildFromString (strXML)
  pbdom_doc.GetRootElement(). &
    GetChildElement("child1").SetAttribute("b", "789")
catch (PBDOM_EXCEPTION except)
  MessageBox ("PBDOM_EXCEPTION", except.GetMessage())
end try
```

After the PBDOM\_DOCUMENT object is saved and converted to XML, the XML document looks like the following:

```
<root xmlns:pre1="http://www.pre.com">
  <child1 pre1:a="123" b="789"/>
</root>
```

#### Usage

This method allows the caller to add an attribute/value pair to a PBDOM\_ELEMENT object. If the PBDOM\_ELEMENT object already contains an existing attribute that has the same name as the input name and that belongs to no namespace, the original attribute is removed from this PBDOM\_ELEMENT object and a new one (corresponding to the specified attribute name and value) is created and set in its place.

If a PBDOM\_ATTRIBUTE has been created to represent the original attribute, it is still valid, but the attribute that it represents has been detached from the original owner element. Calling GetOwnerElementObject on this PBDOM\_ATTRIBUTE returns a null value.

#### See also

GetAttribute  
GetAttributes  
GetAttributeValue  
HasAttributes  
SetAttribute Syntax 1  
SetAttribute Syntax 3, SetAttributes

## SetAttribute Syntax 3

Description	Adds an attribute/value pair to a PBDOM_ELEMENT object. The attribute namespace is specified, and any existing attribute of the same name and namespace URI is removed.														
Syntax	<code><i>pbdom_element_name</i>.SetAttribute(string <i>strName</i>, string <i>strValue</i>, string <i>strNamespacePrefix</i>, string <i>strNamespaceUri</i>, boolean <i>bVerifyNamespace</i>)</code>														
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_element_name</i></td> <td>The name of a PBDOM_ELEMENT object</td> </tr> <tr> <td><i>strName</i></td> <td>The name of the PBDOM_ATTRIBUTE to be added</td> </tr> <tr> <td><i>strValue</i></td> <td>The value of the PBDOM_ATTRIBUTE to be added</td> </tr> <tr> <td><i>strNamespacePrefix</i></td> <td>The prefix of the namespace to which the PBDOM_ATTRIBUTE belongs</td> </tr> <tr> <td><i>strNamespaceUri</i></td> <td>The URI of the namespace to which the PBDOM_ATTRIBUTE belongs</td> </tr> <tr> <td><i>bVerifyNamespace</i></td> <td>Specifies whether or not the method should verify the existence of an in-scope namespace declaration for the given prefix and URI</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object	<i>strName</i>	The name of the PBDOM_ATTRIBUTE to be added	<i>strValue</i>	The value of the PBDOM_ATTRIBUTE to be added	<i>strNamespacePrefix</i>	The prefix of the namespace to which the PBDOM_ATTRIBUTE belongs	<i>strNamespaceUri</i>	The URI of the namespace to which the PBDOM_ATTRIBUTE belongs	<i>bVerifyNamespace</i>	Specifies whether or not the method should verify the existence of an in-scope namespace declaration for the given prefix and URI
Argument	Description														
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object														
<i>strName</i>	The name of the PBDOM_ATTRIBUTE to be added														
<i>strValue</i>	The value of the PBDOM_ATTRIBUTE to be added														
<i>strNamespacePrefix</i>	The prefix of the namespace to which the PBDOM_ATTRIBUTE belongs														
<i>strNamespaceUri</i>	The URI of the namespace to which the PBDOM_ATTRIBUTE belongs														
<i>bVerifyNamespace</i>	Specifies whether or not the method should verify the existence of an in-scope namespace declaration for the given prefix and URI														
Return value	Long. Returns 0 if no namespace verification error occurs and -1 if no in-scope namespace declaration exists for the given prefix and URI settings.														
Throws	<p>EXCEPTION_INVALID_ARGUMENT – If any of the arguments is invalid. This can happen if any of the input strings has been set to null using the PowerScript SetNull function.</p> <p>EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – This PBDOM_ELEMENT object's internal implementation is null. The occurrence of this exception is rare but can take place if severe memory corruption occurs.</p> <p>EXCEPTION_INVALID_NAME – The input namespace prefix or the URI, or their combination, is not valid. This will happen if:</p> <ul style="list-style-type: none"> <li>• The namespace prefix is an empty string and the URI is not an empty string. If both are empty strings, the NONNAMESPACE namespace is being specified and this prefix/URI combination is correct.</li> <li>• The namespace prefix is xmlns and the URI is not <a href="http://www.w3.org/2000/xmlns/">http://www.w3.org/2000/xmlns/</a>. This namespace prefix/URI pair is unique and exclusive and cannot be used separately. The use of this pair signifies a namespace declaration.</li> <li>• The namespace prefix string is invalid. That is, it does not conform to the W3C “Namespaces in XML” specifications for the name of a prefix.</li> </ul>														

- The namespace URI string is invalid. That is, it does not conform to the W3C specifications for a URI string.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – If there has been any memory allocation failure during this method call.

## Examples

**Example 1** The SetAttribute method is invoked for the following XML element:

```
<code>0789725045</code>
```

The SetAttribute method is invoked from the following PowerScript statement, where elem\_code represents the code element:

```
elem_code.SetAttribute("type", "ISBN", "ns", &  
"http://www.books.com/codes", false)
```

The following XML element results:

```
<code ns:type="ISBN">0789725045</code>
```

**Example 2** The following example demonstrates the effect of setting an attribute with a particular name and namespace URI for an element that already contains an existing attribute with the same name and namespace URI. It creates a PBDOM\_DOCUMENT based on the following XML:

```
<root xmlns:pre1="http://www.pre.com"  
xmlns:pre2="http://www.pre.com">  
  <child1 pre1:a="123"/>  
</root>
```

The child1 element already contains an attribute named a that belongs to the namespace http://www.pre.com, as indicated by the pre1 prefix. The call to SetAttribute attempts to set an attribute for child1 with the same name, a, but with the namespace prefix pre2.

The last parameter, *bVerifyNamespace*, is set to true. This tells the SetAttribute method to check first to see if an in-scope namespace declaration for pre2 and http://www.pre.com exists. An in-scope declaration for this namespace prefix/URI pair does exist, and so the verification succeeds.

The original pre1:a attribute is removed from the child1 element and a new attribute pre2:a, belonging to the same namespace and with the value 456, is created and set in its place. The new attribute replaces the original attribute, instead of being set as an additional attribute, because both attributes have the same URI.

```
PBDOM_BUILDER      pbdom_buildr  
PBDOM_DOCUMENT    pbdom_doc  
string strXML = "<root
```



```

xmlns:pre1=~"http://www.pre.com~"
xmlns:pre2=~"http://www.pre.com~"><child1
pre1:a=~"123~/></root>"

try
    pbdom_buildr = Create PBDOM_BUILDER
    pbdom_doc = pbdom_buildr.BuildFromString (strXML)

    pbdom_doc.GetRootElement().GetChildElement("child1").SetAttribute("a", "456", "pre2", "http://www.pre.com", true)

catch (PBDOM_EXCEPTION pbdom_except)
    MessageBox ("PBDOM_EXCEPTION",
pbdom_except.GetMessage())
end try

```

**Usage**

This method allows the caller to add an attribute/value pair to a PBDOM\_ELEMENT object.

The parameter *bVerifyNamespace*, when set to true, instructs the method to perform a thorough search up the DOM node tree, starting at the current PBDOM\_ELEMENT object, to check for an in-scope namespace declaration for the given prefix and URI. If a namespace declaration is not found, no attribute is created. If a namespace declaration is found, an attribute is created.

If the *bVerifyNamespace* parameter is set to false, no verification search is performed, and the method always returns 0.

If the PBDOM\_ELEMENT object already contains an existing attribute that has the same name as the input name and the same namespace URI as the input namespace URI, the original attribute is replaced with a new one with the same name and URI.

If a PBDOM\_ATTRIBUTE has been created to represent the original attribute, it is still valid, but the attribute that it represents has been detached from the original owner element. Calling `GetOwnerElementObject` on this PBDOM\_ATTRIBUTE returns a null value.

**See also**

[GetAttribute](#)  
[GetAttributes](#)  
[GetAttributeValue](#)  
[HasAttributes](#)  
[SetAttribute Syntax 1](#)  
[SetAttribute Syntax 2, SetAttributes](#)

## SetAttributes

**Description** Sets the attributes for the DOM element represented by the current PBDOM\_ELEMENT object.

**Syntax** `pbdom_element_name.SetAttributes(pbdom_attribute pbdom_attribute_array[])`

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>pbdom_attribute_array</i>	An array of PBDOM_ATTRIBUTE objects

**Return value** PBDOM\_ELEMENT. The PBDOM\_ELEMENT object modified.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – The internal implementation of this PBDOM\_ELEMENT object or one of the PBDOM\_ATTRIBUTE array items is null. This exception is rare but can take place if severe memory corruption occurs.

EXCEPTION\_INVALID\_ARGUMENT – One of the PBDOM\_ATTRIBUTE array items is null.

EXCEPTION\_INVALID\_NAME – If two or more PBDOM\_ATTRIBUTES in the array contain the same name and namespace URI.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – One of the PBDOM\_ATTRIBUTE array items has not been named.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_OWNER – One of the PBDOM\_ATTRIBUTE array items already has an owner PBDOM\_ELEMENT object.

**Examples** This example demonstrates setting the attributes of a PBDOM\_ELEMENT object using an array of PBDOM\_ATTRIBUTE objects. It builds a PBDOM\_DOCUMENT based on the following XML:

```
<root xmlns:pre1="http://www.pre.com">
  <child1 pre1:a="123"/>
</root>
```

The code creates an array of three PBDOM\_ATTRIBUTE objects with names a, b, and c, and sets their namespace prefixes and URIs to pre1 and http://www.pre.com. The call to SetAttributes attempts to set the attributes of child1 using the PBDOM\_ATTRIBUTES of this array. When you save PBDOM\_DOCUMENT and convert it to an XML document, the result is:

```
<root xmlns:pre1="http://www.pre.com">
  <child1 pre1:a="456" pre1:b="456" pre1:c="456" />
</root>
```

Although child1 originally contained the pre1:a attribute, and the PBDOM\_ATTRIBUTE array also contained an item with name a within the namespace URI http://www.pre.com, no exception is thrown. The original pre1:a attribute is replaced by the PBDOM\_ATTRIBUTE array item with name a within the namespace URI http://www.pre.com.

```

PBDOM_BUILDER      pbdom_buildr
PBDOM_DOCUMENT     pbdom_doc
PBDOM_ATTRIBUTE    pbdom_attr_array[]
string              Name[]
long                l = 0
string strXML = "<root
xmlns:pre1=~"http://www.pre.com~"><child1
pre1:a=~"123~/></root>"

try
    pbdom_buildr = Create PBDOM_BUILDER
    pbdom_doc = pbdom_buildr.BuildFromString (strXML)

    Name[1] = "a"
    Name[2] = "b"
    Name[3] = "c"

    for l = 1 to 3
        pbdom_attr_array[l] = Create PBDOM_ATTRIBUTE
        pbdom_attr_array[l].SetName (Name[l])
        pbdom_attr_array[l].SetNamespace ("pre1", &
            "http://www.pre.com", false)
        pbdom_attr_array[l].SetText ("456")
    next

    pbdom_doc.GetRootElement().GetChildElement &
        ("child1").SetAttributes(pbdom_attr_array)
    pbdom_doc.SaveDocument ("set_attributes.xml")

catch (PBDOM_EXCEPTION except)
    MessageBox ("PBDOM_EXCEPTION", except.GetMessage())
end try

```

#### Usage

This method sets the attributes of the DOM element represented by this PBDOM\_ELEMENT object. The supplied array should contain only objects of type PBDOM\_ATTRIBUTE.

When all objects in the supplied array are legal and before the new attributes are added, all old attributes have their parentage set to null (no parent) and the old attribute list is cleared from this PBDOM\_ELEMENT object. This has the effect that any active attribute list (previously obtained with a call to GetAttributes) also changes to reflect the new situation with the old attributes. In addition, all PBDOM\_ATTRIBUTES in the supplied array have their parentage set to this current PBDOM\_ELEMENT object.

Passing an empty array clears the existing attributes of this PBDOM\_ELEMENT object.

This method fails and an exception is thrown if the PBDOM\_ATTRIBUTE array contains two or more PBDOM\_ATTRIBUTES with the same name and namespace URI.

No exception is thrown if this PBDOM\_ELEMENT object contains an existing attribute whose name and namespace URI matches one of the PBDOM\_ATTRIBUTE array items. All the existing attributes of this PBDOM\_ELEMENT object are removed, so it does not matter whether any existing attribute matches any of the PBDOM\_ATTRIBUTE items in the array in terms of name and namespace URI.

In the event of an exception, the original attributes of the PBDOM\_ELEMENT object remain unchanged, and the PBDOM\_ATTRIBUTES in the supplied array are not altered.

If any PBDOM\_ATTRIBUTE has been created to represent any original attribute, it is still valid, but the attribute it represents has been detached from the original owner element. Calling GetOwnerElementObject on this PBDOM\_ATTRIBUTE returns a null value.

See also

GetAttribute  
GetAttributes  
GetAttributeValue  
HasAttributes  
SetAttribute

## SetContent

**Description** Sets the content of the PBDOM\_ELEMENT object using an array containing PBDOM\_OBJECT objects legal for a PBDOM\_ELEMENT object. Any existing children of the PBDOM\_ELEMENT object are removed when the SetContent method is invoked.

If the input array reference is null, all contents of the PBDOM\_ELEMENT object are removed. If the array contains illegal objects, an exception is thrown, and nothing is altered.

**Syntax** `pbdom_element_name.SetContent(pbdom_object pbdom_object_array[])`

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>pbdom_object_array</i>	An array of PBDOM_OBJECTS to form the contents the PBDOM_ELEMENT object

**Return value** PBDOM\_OBJECT. The PBDOM\_ELEMENT object modified and returned as a PBDOM\_OBJECT.

**Throws** EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If an input PBDOM\_OBJECT array item has not been given a user-defined name.

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If an input PBDOM\_OBJECT array item is not associated with a derived PBDOM\_OBJECT.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – If an input PBDOM\_OBJECT array item already has a parent PBDOM\_OBJECT.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If an inappropriate PBDOM\_OBJECT array item is found. This happens if the PBDOM\_OBJECT array item is not allowed to be added as a child of a PBDOM\_ELEMENT object (for example, a PBDOM\_DOCUMENT).

EXCEPTION\_HIERARCHY\_ERROR – If one of the PBDOM\_OBJECT array items, if set as part of the contents of this PBDOM\_ELEMENT object, will cause the current PBDOM\_ELEMENT object to be no longer well formed.

Examples

The SetContent method is invoked on the following XML fragment:

```
<Telephone_Book>
  <Entry>
    <Particulars>
      <Name>John Doe</Name>
      <Age>21</Age>
      <Phone_Number>1234567</Phone_Number>
    </Particulars>
  </Entry>
</Telephone_Book>
```

The SetContent method is invoked from the following PowerScript code:

```
PBDOM_OBJECT pbdom_obj_array[]

pbdom_obj_array[1] = entry_1
pbdom_obj_array[2] = entry_2

pbdom_doc.GetRootElement().SetContent(pbdom_obj_array)
```

The entry\_1 PBDOM\_ELEMENT object contains the following:

```
<Entry>
  <Particulars>
    <Name>James Gomez</Name>
    <Age>25</Age>
    <Phone_Number>1111111</Phone_Number>
  </Particulars>
</Entry>
```

The entry\_2 PBDOM\_ELEMENT object contains the following:

```
<Entry>
  <Particulars>
    <Name>Mary Jones</Name>
    <Age>22</Age>
    <Phone_Number>2222222</Phone_Number>
  </Particulars>
</Entry>
```

The SetContent method returns the following:

```
<Telephone_Book>
  <Entry>
    <Particulars>
      <Name>James Gomez</Name>
      <Age>25</Age>
      <Phone_Number>1111111</Phone_Number>
    </Particulars>
  </Entry>
  <Entry>
    <Particulars>
      <Name>Mary Jones</Name>
      <Age>22</Age>
      <Phone_Number>2222222</Phone_Number>
    </Particulars>
  </Entry>
</Telephone_Book>
```

#### Usage

Only the following PBDOM\_OBJECT types can be validly added to a PBDOM\_ELEMENT object:

- PBDOM\_ELEMENT
- PBDOM\_CDATA
- PBDOM\_COMMENT
- PBDOM\_ENTITYREFERENCE
- PBDOM\_PROCESSINGINSTRUCTION
- PBDOM\_TEXT

#### See also

AddContent Syntax 1  
 AddContent Syntax 2  
 GetContent  
 InsertContent  
 RemoveContent

## SetDocument

**Description** Sets a PBDOM\_DOCUMENT as parent of a PBDOM\_ELEMENT object, making the PBDOM\_ELEMENT object the root element.

**Syntax** `pbdom_element_name.SetDocument(pbdom_document pbdom_document_ref)`

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>pbdom_document_ref</i>	The PBDOM_DOCUMENT to be set as the owner document and parent of this PBDOM_ELEMENT object

**Return value** PBDOM\_ELEMENT. The modified PBDOM\_ELEMENT object.

**Usage** The PBDOM\_OBJECT referenced must be a PBDOM\_DOCUMENT object. The PBDOM\_ELEMENT object must not already have a parent object. If the target PBDOM\_DOCUMENT already has a root element, the existing root element is replaced by the new PBDOM\_ELEMENT object.

## SetName

**Description** Sets the local name of a PBDOM\_ELEMENT object. This name refers to the local portion of the element tag name.

**Syntax** `pbdom_element_name.SetName(string strName)`

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>strName</i>	The new local name for the PBDOM_ELEMENT object

**Return value** Boolean. Returns true if the local name of the PBDOM\_ELEMENT object has been changed, and false otherwise.

**Examples** The SetName method is invoked for the abc element of the following XML fragment:

```
<abc>My Data</abc>
```

The SetName method is invoked in the following PowerShell code, in which the PBDOM\_ELEMENT object `elem` represents the `abc` element.

```
elem.SetName("def")
```

The following XML results:

```
<def>My Data</def>
```



Since the `elem` object still represents the same element, calling the `SetName` method changes the `def` element.

See also `GetName`

## SetNamespace

**Description** Sets the namespace for a `PBDOM_ELEMENT` object. If the namespace prefix and URI provided are empty strings, `SetNamespace` assigns no namespace to the `PBDOM_ELEMENT` object.

**Syntax** `pbdom_element_name.SetNamespace(string strNamespacePrefix, string strNamespaceUri, boolean bVerifyNamespace)`

Argument	Description
<code>pbdom_element_name</code>	The name of a <code>PBDOM_ELEMENT</code> object
<code>strNamespacePrefix</code>	Prefix of the namespace to be set for the <code>PBDOM_ELEMENT</code> object
<code>strNamespaceUri</code>	URI of the namespace to be set for the <code>PBDOM_ELEMENT</code> object
<code>bVerifyNamespace</code>	A boolean value indicating whether verification should be performed to ensure that the provided namespace prefix and URI have been declared either within this <code>PBDOM_ELEMENT</code> object or in an ancestor <code>PBDOM_ELEMENT</code> object

**Return value** Long. Returns 0 for success and -1 if no in-scope namespace declaration matching the input prefix and URI exists.

**Throws** `EXCEPTION_INVALID_ARGUMENT` – If any of the input arguments is invalid, for example, null.

`EXCEPTION_INVALID_NAME` – If the input namespace prefix or URI is invalid.

`EXCEPTION_MEMORY_ALLOCATION_FAILURE` – If a memory allocation failure occurred during the execution of this method.

`EXCEPTION_INTERNAL_XML_ENGINE_ERROR` – If an internal XML engine failure occurred during the execution of this method.

**Usage** If `bVerifyNamespace` is set to true and the namespace prefix and URI have not been declared, `SetNamespace` returns a value of -1 and fails.

If *bVerifyNamespace* is set to false, *SetNamespace* sets the namespace of the PBDOM\_ELEMENT object to the specified prefix and URI. It is the responsibility of the PBDOM user to ensure that such a namespace is declared and is in scope for this PBDOM\_ELEMENT object before the document is saved and converted to an XML document.

See also                   AddNamespaceDeclaration  
                               GetNamespacePrefix  
                               GetNamespaceUri  
                               GetQualifiedName  
                               RemoveNamespaceDeclaration

## SetParentObject

Description               Sets the referenced PBDOM\_OBJECT as the parent of the PBDOM\_ELEMENT object from which the method is invoked.

Syntax                    *pbdom\_element\_name*.SetParentObject(*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to be set as the parent of this PBDOM_ELEMENT object

Return value              PBDOM\_OBJECT. The PBDOM\_ELEMENT object modified and returned as a PBDOM\_OBJECT.

Throws

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not associated with a derived PBDOM\_OBJECT.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – The input PBDOM\_OBJECT already has a parent.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT is not allowed to be the parent of a PBDOM\_ELEMENT object.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT is nameable and has not been named.

Usage	If the class of the referenced PBDOM_OBJECT is PBDOM_DOCUMENT, then the behavior of SetParentObject is identical to that of the SetDocument method. If the class of the referenced PBDOM_OBJECT is PBDOM_ELEMENT, SetParentObject sets the referenced object as the parent of the PBDOM_ELEMENT object from which the method is invoked. If the referenced PBDOM_OBJECT is of any other class, an exception is thrown.
See also	GetOwnerDocumentObject GetParentObject

## SetText

Description	Sets the content of a PBDOM_ELEMENT object to the text provided.						
Syntax	<i>pbdom_element_name</i> .SetText(string <i>strText</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_element_name</i></td> <td>The name of a PBDOM_ELEMENT object</td> </tr> <tr> <td><i>strText</i></td> <td>String to be set as the content of the PBDOM_ELEMENT object</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object	<i>strText</i>	String to be set as the content of the PBDOM_ELEMENT object
Argument	Description						
<i>pbdom_element_name</i>	The name of a PBDOM_ELEMENT object						
<i>strText</i>	String to be set as the content of the PBDOM_ELEMENT object						
Return value	PBDOM_OBJECT. The PBDOM_ELEMENT object modified and returned as a PBDOM_OBJECT.						
Usage	Existing text content and non-text content are replaced by the text provided in <i>strText</i> . A value of null for <i>strText</i> is equivalent to an empty string value. If the PBDOM_ELEMENT is to have both text content and nested elements, use the SetContent method instead of SetText.						
See also	GetText GetTextNormalize GetTextTrim						



About this chapter

This chapter lists PBDOM exception codes and describes the PBDOM\_EXCEPTION class.

Contents

Topic	Page
PBDOM exceptions	287
PBDOM_EXCEPTION	293

## PBDOM exceptions

PBDOM defines an exception class derived from the standard PowerBuilder Exception class. This class extends the Exception class with a method, `GetExceptionCode`, that returns the unique code that identifies the exception being thrown.

The following table lists PBDOM exceptions and their code values. The circumstances in which each exception is thrown are described after the table.

**Table 14-1: PBDOM exceptions and code values**

Exception	Value
EXCEPTION_USE_OF_UNNAMED_PBDOM_OBJECT	1
EXCEPTION_WRONG_DOCUMENT_ERROR	2
EXCEPTION_MULTIPLE_ROOT_ELEMENT	3
EXCEPTION_INAPPROPRIATE_USE_OF_PBDOM_OBJECT	4
EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE	5
EXCEPTION_PBDOM_OBJECT_ALREADY_HAS_PARENT	6
EXCEPTION_MULTIPLE_DOCTYPE	7
EXCEPTION_ILLEGAL_PBOBJECT	8
EXCEPTION_WRONG_PARENT_ERROR	9
EXCEPTION_INVALID_ARGUMENT	10
EXCEPTION_INVALID_NAME	11
EXCEPTION_DATA_CONVERSION	12

Exception	Value
EXCEPTION_MEMORY_ALLOCATION_FAILURE	13
EXCEPTION_INTERNAL_XML_ENGINE_ERROR	14
EXCEPTION_MULTIPLE_XMLDECL	15
EXCEPTION_INVALID_STRING	16
EXCEPTION_INVALID_OPERATION	17
EXCEPTION_HIERARCHY_ERROR	18
EXCEPTION_PBDOM_OBJECT_ALREADY_HAS_OWNER	19
EXCEPTION_PBDOM_NOT_INITIALIZED	20

## PBDOM exception descriptions

### EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT

Code Value: 1

This exception is thrown when you use a nameable PBDOM\_OBJECT—for example, to invoke a method or serve as a parameter—without first being given a user-defined name.

### EXCEPTION\_WRONG\_DOCUMENT\_ERROR

Code Value: 2

This exception is thrown when you use incorrect PBDOM\_DOCUMENT objects when performing a PBDOM operation. For example, in a RemoveContent method call, if the PBDOM\_OBJECT you want to remove is not from the same document as the active PBDOM\_DOCUMENT whose RemoveContent method is being invoked, this exception is thrown.

### EXCEPTION\_MULTIPLE\_ROOT\_ELEMENT

Code Value: 3

This exception is thrown when a PBDOM method call causes a PBDOM\_DOCUMENT to contain more than one root element.

For example, in an AddContent method call, if the input PBDOM\_OBJECT to add is a PBDOM\_ELEMENT and the active PBDOM\_DOCUMENT already contains a root element, this exception is thrown.

**EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT**

Code Value: 4

This exception is thrown when a PBDOM\_OBJECT is used in an inappropriate manner. A typical scenario is one in which a PBDOM method call results in the violation of the well-formedness of a PBDOM\_DOCUMENT.

For example, in an AddContent method invoked on a PBDOM\_DOCUMENT object, only PBDOM\_OBJECTs of class PBDOM\_ELEMENT, PBDOM\_COMMENT, PBDOM\_PROCESSINGINSTRUCTION, and PBDOM\_DOCTYPE can be added. The inclusion of PBDOM\_OBJECTs of any other class results in this exception being thrown.

**EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE**

Code Value: 5

This exception is thrown when an invalid PBDOM\_OBJECT is used, either directly to invoke a method, or as a parameter.

Situations where a PBDOM\_OBJECT is deemed invalid include those where a PBDOM\_OBJECT is instantiated as a PBDOM\_OBJECT and not as a derived class object. They also include the situation where a PBDOM\_CHARACTERDATA object is instantiated directly as a PBDOM\_CHARACTERDATA object.

**EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT**

Code Value: 6

This exception occurs when a PBDOM\_OBJECT is set to be the child of another PBDOM\_OBJECT, but the prospective child already has a parent PBDOM\_OBJECT.

Examples of such method calls include the AddContent method and the SetParentObject, SetContent, and InsertContent methods of all classes derived from PBDOM\_OBJECT classes.

## **EXCEPTION\_MULTIPLE\_DOCTYPE**

Code Value: 7

This exception is thrown when a PBDOM method call causes a PBDOM\_DOCUMENT to contain more than one DOCTYPE.

For example, in an AddContent method call, if the input PBDOM\_OBJECT to add is a PBDOM\_DOCTYPE and the active PBDOM\_DOCUMENT already contains a DOCTYPE DOM Node, this exception is thrown.

## **EXCEPTION\_ILLEGAL\_PBOBJECT**

Code Value: 8

This exception is thrown in method calls that take an array of PBDOM\_OBJECTs in which one of the array items is invalid. A PBDOM\_OBJECT array item is deemed to be invalid when it has been specifically set to null or has not been initialized properly.

## **EXCEPTION\_WRONG\_PARENT\_ERROR**

Code Value: 9

This exception is thrown when an incorrect parent/child relationship error is encountered during a PBDOM operation.

Method calls in which this exception might be thrown include InsertContent and RemoveContent. These methods involve at least one PBDOM\_OBJECT parameter that is assumed to be a child of the PBDOM\_OBJECT to which the method is applied. If this parameter is not a child of the current PBDOM\_OBJECT, this exception is thrown.

## **EXCEPTION\_INVALID\_ARGUMENT**

Code Value: 10

This exception is thrown when an input PBDOM\_OBJECT parameter to a method is invalid. This can happen if it has not been initialized properly, or if it is a null object reference.

This exception might also be thrown when an input string parameter to a method is invalid. This can happen if the string has been set to null using the PowerScript SetNull function.



## **EXCEPTION\_INVALID\_NAME**

Code Value: 11

This exception is thrown when a name is supplied as a parameter and the name does not conform to the W3C specifications for an XML name or namespace prefix or namespace URI.

Methods in which this exception might be thrown include the `SetName`, `SetNamespace`, and `SetNamespace` methods.

## **EXCEPTION\_DATA\_CONVERSION**

Code Value: 12

This exception is thrown when you attempt to perform a data conversion operation and the conversion fails. This exception is thrown only in the `PBDOM_ATTRIBUTE` object's `Get` methods, for example, `GetDateValue` in `PBDOM_ATTRIBUTE`.

## **EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE**

Code Value: 13

This exception is thrown when insufficient memory is encountered while executing a method. PBDOM internally allocates, frees, and reallocates memory for storing strings, structures, and so on. Each memory allocation might fail, and if this occurs, this exception is thrown.

## **EXCEPTION\_INTERNAL\_XML\_ENGINE\_ERROR**

Code Value: 14

This exception is thrown when an internal error occurs that involves the XML engine used by PBDOM. PBDOM currently uses the Xerces XML parser as the underlying device for processing XML documents and for building up and sustaining the DOM tree.

There may be problems in the low-level XML parser engine, and if one is encountered, this exception, which is rare, might be thrown.

## **EXCEPTION\_MULTIPLE\_XMLDECL**

Code Value: 15

This exception is thrown when a PBDOM method call causes a PBDOM\_DOCUMENT to contain more than one XML declaration.

For example, in a SetContent method call invoked on a PBDOM\_DOCUMENT object, if the input PBDOM\_OBJECT array contains more than one PBDOM\_PROCESSINGINSTRUCTION that is constructed as an XML declaration, this exception is thrown.

## **EXCEPTION\_INVALID\_STRING**

Code Value: 16

This exception is thrown when a string is supplied as a parameter to a method that sets a text or attribute value, and the string contains characters that do not conform to the W3C specifications for acceptable XML characters.

Methods in which this exception might be thrown include SetText in PBDOM\_ATTRIBUTE and SetAttribute in PBDOM\_ELEMENT.

## **EXCEPTION\_INVALID\_OPERATION**

Code Value: 17

This exception is thrown when a method call could potentially cause severe and unexpected problems to the currently running PowerBuilder application.

## **EXCEPTION\_HIERARCHY\_ERROR**

Code Value: 18

This exception is thrown when a method call violates the well-formedness or validity of a PBDOM\_DOCUMENT.

## **EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_OWNER**

Code Value : 19

This exception is thrown when a PBDOM\_ELEMENT is set as the owner of a PBDOM\_ATTRIBUTE when the specified PBDOM\_ATTRIBUTE already has an owner PBDOM\_ELEMENT.

## EXCEPTION\_PBDOM\_NOT\_INITIALIZED

Code Value : 20

This exception is thrown in rare circumstances in which the PBDOM engine has failed to be initialized or has been uninitialized prematurely. In such situations, an exception is thrown to prevent a crash.

## PBDOM\_EXCEPTION

**Description** The PBDOM\_EXCEPTION class is derived from the PowerBuilder Exception class.

**Methods** This class extends the Exception class with one method that returns the unique code that identifies the exception being thrown:

GetExceptionCode

### GetExceptionCode

**Description** Returns the code of the exception being thrown.

**Syntax** *pbdom\_exception*.GetExceptionCode()

Argument	Description
<i>pbdom_exception</i>	The name of a PBDOM_EXCEPTION object

**Return value** Long. The code value associated with the exception being thrown.

## Examples

In this example, an attempt to call the PBDOM\_ELEMENT GetAttribute method on the root element of a PBDOM\_DOCUMENT with the parameter *xmlns:nuskin* causes an exception to be thrown, because the name is not a valid NCName (no-colon-name). The correct way to get an attribute that belongs to a namespace is to use the namespace version of the PBDOM\_ELEMENT GetAttribute method.

The EXCEPTION\_INVALID\_NAME (code value 11) exception is thrown and is displayed in a message box :

```
PBDOM_DOCUMENT    pbdom_doc1
PBDOM_DOCUMENT    pbdom_get_doc
PBDOM_ELEMENT     pbdom_elem_root
PBDOM_ATTRIBUTE   pbdom_attr
PBDOM_OBJECT      pbdom_obj

try
    pbdom_doc1 = Create PBDOM_DOCUMENT

    pbdom_doc1.NewDocument("nuskin", &
        "http://www.nuskin.com", "nuskin:root", "", "")
    pbdom_elem_root = pbdom_doc1.GetRootElement()
    pbdom_attr = &
        pbdom_elem_root.GetAttribute("xmlns:nuskin")

catch (PBDOM_EXCEPTION pbdom_except)
    MessageBox ("Exception", "Code : " &
        + string(pbdom_except.GetExceptionCode()) &
        + "~r~nText : " + pbdom_except.Text)
end try
```

## Usage

For a list of exception codes, see “PBDOM exceptions” on page 287. For a description of the conditions under which each exception can occur, see “PBDOM exception descriptions” on page 288.

## See also

GetAttribute Syntax 2 (PBDOM\_ELEMENT)  
GetMessage and SetMessage in the *PowerScript Reference*.

About this chapter

This chapter describes the PBDOM\_OBJECT class.

## **PBDOM\_OBJECT**

Description

A PBDOM\_OBJECT serves as the base class for all the PBDOM classes. It contains all the basic methods required by derived classes. The derived classes of a PBDOM\_OBJECT each inherit the base methods of a PBDOM\_OBJECT, and additionally contain their own specialized methods.

Methods

PBDOM\_OBJECT has the following methods:

- AddContent
- Clone
- Detach
- Equals
- GetContent
- GetOwnerDocumentObject
- GetName
- GetObjectClass
- GetObjectClassString
- GetParentObject
- GetText
- GetTextNormalize
- GetTextTrim
- HasChildren
- InsertContent
- IsAncestorObjectOf
- RemoveContent
- SetContent
- SetName
- SetParentObject

## AddContent

Description	Adds a new PBDOM_OBJECT into the current PBDOM_OBJECT.						
Syntax	<i>pbdom_object_name</i> .AddContent( <i>pbdom_object pbdom_object_ref</i> )						
	<table><thead><tr><th>Argument</th><th>Description</th></tr></thead><tbody><tr><td><i>pbdom_object_name</i></td><td>The name of the PBDOM_OBJECT</td></tr><tr><td><i>pbdom_object_ref</i></td><td>The PBDOM_OBJECT to add</td></tr></tbody></table>	Argument	Description	<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT	<i>pbdom_object_ref</i>	The PBDOM_OBJECT to add
Argument	Description						
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT						
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to add						
Return value	PBDOM_OBJECT.  The return value is the newly modified PBDOM_OBJECT.						
Throws	EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – This PBDOM_OBJECT object or the input PBDOM_OBJECT is not associated with a derived PBDOM_OBJECT class object.  EXCEPTION_INVALID_ARGUMENT – Input argument is invalid.						
Usage	When a new PBDOM_OBJECT is added to the current one, the new PBDOM_OBJECT becomes a child node of the current PBDOM_OBJECT.						
See also	GetContent InsertContent RemoveContent SetContent						

## Clone

Description	Creates a general duplicate of the current PBDOM_OBJECT.						
Syntax	<i>pbdom_object_name</i> .Clone( <i>boolean bDeep</i> )						
	<table><thead><tr><th>Argument</th><th>Description</th></tr></thead><tbody><tr><td><i>pbdom_object_name</i></td><td>The name of the PBDOM_OBJECT.</td></tr><tr><td><i>bDeep</i></td><td>A boolean specifying whether a deep or shallow clone is returned. Values are true for a deep clone and false for a shallow clone.</td></tr></tbody></table>	Argument	Description	<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT.	<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are true for a deep clone and false for a shallow clone.
Argument	Description						
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT.						
<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are true for a deep clone and false for a shallow clone.						
Return value	PBDOM_OBJECT. The return value is the clone of the PBDOM_OBJECT.						
Throws	EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – This PBDOM_OBJECT object is not associated with a derived PBDOM_OBJECT class object.						

**Usage**

The Clone method creates a general duplicate of the current PBDOM\_OBJECT. If the *bDeep* parameter is true, a deep clone is returned; otherwise, a shallow clone is returned.

A PBDOM\_OBJECT clone does not have a parent; however, it resides in the same PBDOM\_DOCUMENT as its original. If the original PBDOM\_OBJECT is standalone, the clone is also standalone.

If general, if *bDeep* is true, the Clone method recursively clones the subtree under the PBDOM\_OBJECT. If *bDeep* is false, the Clone method clones only the PBDOM\_OBJECT itself, together with as much information as possible.

---

### Cloning is class specific

Cloning is not uniform across all PBDOM\_OBJECT classes. See the documentation for each class for specific information.

---

## Detach

**Description** Detaches a PBDOM\_OBJECT from its parent.

**Syntax**

```
pbdom_object_name.Detach()
```

Argument	Description
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object is not associated with a derived PBDOM\_OBJECT class object.

**Examples**

This example detaches the root element of a PBDOM\_DOCUMENT called *pbdom\_doc* from its parent object—that is, from the PBDOM\_DOCUMENT itself. Then, it attempts to obtain the parent PBDOM\_OBJECT and tests whether it is null using the *IsValid* method:

```
pbdom_obj = pbdom_doc.GetRootElement()
pbdom_obj.Detach()
pbdom_parent_obj = pbdom_obj.GetParentObject()
if (not IsValid(pbdom_parent_obj)) then
    MessageBox ("Invalid", "Root Element has no
Parent")
end if
```

**Usage** If the PBDOM\_OBJECT has no parent, this method does nothing.

## Equals

Description

Tests for the equality of a referenced PBDOM\_OBJECT.

Syntax

*pbdom\_object\_name*.Equals(*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to test for equality with the current PBDOM_OBJECT

Return value

Boolean. Returns true if the current PBDOM\_OBJECT is equivalent to the input PBDOM\_OBJECT, and false otherwise.

Throws

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object or the input PBDOM\_OBJECT is not associated with a derived PBDOM\_OBJECT class object.

EXCEPTION\_INVALID\_ARGUMENT – The input PBDOM\_OBJECT is invalid. This can happen if the object has not been initialized properly or is a null object reference.

## GetContent

Description

Obtains an array of PBDOM\_OBJECT objects, each of which is a child node of the called PBDOM\_OBJECT.

Syntax

*pbdom\_object\_name*.GetContent(*ref pbdom\_object pbdom\_object\_array[]*)

Argument	Description
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT
<i>pbdom_object_array</i>	A reference to an array of PBDOM_OBJECT objects that will receive the PBDOM_OBJECT objects

Return value

Boolean. Returns true for success, and false otherwise.

Throws

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object is not associated with a derived PBDOM\_OBJECT class object.

Usage

The returned array is passed by reference, with items in the same order in which they appear in the PBDOM\_OBJECT. Any changes to any item of the array affect the actual item to which it refers.

See also

AddContent, InsertContent, RemoveContent, SetContent



## GetName

### Description

Obtains the name of the current PBDOM\_OBJECT. The returned string depends on the type of DOM Object that is contained within a PBDOM\_OBJECT.

DOM Object Type	Return Value
PBDOM_DOCTYPE	"#document"
PBDOM_ELEMENT	<p>The local tag name of the element, without any namespace prefixes.</p> <p>For example, if the element is:  <code>&lt;abc&gt;Value&lt;/abc&gt;</code>, then the string returned from <code>GetName</code> is "abc".</p> <p>Also, if the tag name of the element contains a namespace prefix, the prefix is not included in the returned string.</p> <p>For example, if the element is:  <code>&lt;MyMusic:CD xmlns:MyMusic="http://www.MyMusicDiscs.com"/&gt;</code>, then the string returned from <code>GetName</code> is "CD".</p>
PBDOM_ATTRIBUTE	<p>The local name of the attribute itself, without a namespace.</p> <p>For example, if the element with the attribute is:  <code>&lt;abc ATTRIBUTE_1="My Attribute"&gt;</code>, then <code>GetName</code> returns "ATTRIBUTE_1".</p> <p>If the name of the attribute contains a namespace prefix, then the prefix is not included in the returned string.</p> <p>For example, if the element with an attribute is:  <code>&lt;MyMusic:CD xmlns:MyMusic="http://www.MyMusicDiscs.com" MyMusic:Type="Jazz"/&gt;</code>, then <code>GetName</code> returns the string "Type".</p>
PBDOM_CDATA	"#cdata-section"
PBDOM_COMMENT	"#comment"
PBDOM_DOCTYPE	<p>The name that was given to the doctype object itself.</p> <p>For example, if the DOCTYPE declaration is:  <code>&lt;!DOCTYPE d_grid_object &gt;</code>, then <code>GetName</code> returns "d_grid_object".</p>

DOM Object Type	Return Value
PBDOM_PROCESSINGINSTRUCTION	The name that was given to the processing instruction itself.  For example, if the processing instruction definition is: <code>&lt;?works document="hello.doc" data="hello.wks" ?&gt;</code> , then <code>GetName</code> returns "works".
PBDOM_TEXT	"#text"

Syntax `pbdom_object_name.GetName()`

Argument	Description
<code>pbdom_object_name</code>	The name of the PBDOM_OBJECT

Return value The following table lists the return values, based on the type of DOM Object contained within the PBDOM\_OBJECT:

Throws EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If this PBDOM\_OBJECT is not a reference to an object derived from PBDOM\_OBJECT.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

Usage A PBDOM\_OBJECT cannot be instantiated directly.

See also `SetName`

## GetObjectClass

Description Returns a long integer code that indicates the class of this PBDOM\_OBJECT.

Syntax `pbdom_object_name.GetObjectClass()`

Argument	Description
<code>pbdom_object_name</code>	The name of the PBDOM_OBJECT

Return value Long. A code that indicates the class of the current PBDOM\_OBJECT.

Usage This method returns the following possible values:

Class	Long integer value
UNKNOWN (indicates an error)	0
PBDOM_OBJECT (the base class)	1
PBDOM_DOCUMENT	2

Class	Long integer value
PBDOM_ELEMENT	3
PBDOM_DOCTYPE	4
PBDOM_ATTRIBUTE	5
PBDOM_CHARACTERDATA	6
PBDOM_TEXT	7
PBDOM_CDATA	8
PBDOM_COMMENT	9
PBDOM_PROCESSINGINSTRUCTION	10
PBDOM_ENTITYREFERENCE	11

See also [GetObjectClassString](#)

## GetObjectClassString

**Description** Returns a string form of the class of the PBDOM\_OBJECT.

**Syntax** `pbdom_object_name.GetObjectClassString()`

Argument	Description
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT

**Return value** String. A string that indicates the class of the current PBDOM\_OBJECT.

**Usage** This method returns the following possible values:

Class	String returned
PBDOM_OBJECT	<code>pbdom_object</code>
PBDOM_DOCUMENT	<code>pbdom_document</code>
PBDOM_ELEMENT	<code>pbdom_element</code>
PBDOM_ENTITYREFERENCE	<code>pbdom_entityreference</code>
PBDOM_DOCTYPE	<code>pbdom_doctype</code>
PBDOM_ATTRIBUTE	<code>pbdom_attribute</code>
PBDOM_CHARACTERDATA	<code>pbdom_characterdata</code>
PBDOM_TEXT	<code>pbdom_text</code>
PBDOM_CDATA	<code>pbdom_cdata</code>
PBDOM_COMMENT	<code>pbdom_comment</code>
PBDOM_PROCESSINGINSTRUCTION	<code>pbdom_processinginstruction</code>

See also [GetObjectClass](#)

## GetOwnerDocumentObject

**Description** Returns the owning PBDOM\_DOCUMENT of the current PBDOM\_OBJECT.

**Syntax** *pbdom\_object\_name*.GetOwnerDocumentObject()

<b>Argument</b>	<b>Description</b>
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT

**Return value** PBDOM\_DOCUMENT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object is not associated with a derived PBDOM\_OBJECT class object.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

**Usage** The owning PBDOM\_DOCUMENT of the current PBDOM\_OBJECT is null if PBDOM\_OBJECT is not owned by any PBDOM\_DOCUMENT, or if the current PBDOM\_OBJECT is itself a PBDOM\_DOCUMENT object.

**See also** GetParentObject  
SetParentObject

## GetParentObject

Description	Returns the parent PBDOM_OBJECT of the current PBDOM_OBJECT.				
Syntax	<i>pbdom_object_name</i> .GetParentObject()				
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_object_name</i></td> <td>The name of the PBDOM_OBJECT</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT
Argument	Description				
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT				
Return value	PBDOM_OBJECT.				
Throws	<p>EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – This PBDOM_OBJECT object is not associated with a derived PBDOM_OBJECT class object.</p> <p>EXCEPTION_MEMORY_ALLOCATION_FAILURE – Insufficient memory was encountered while executing this method.</p>				
Examples	<p>Using the GetRootElement method, the root element of a PBDOM_DOCUMENT called <i>pbdom_doc</i> is returned into a PBDOM_OBJECT called <i>pbdom_obj</i>. The GetParentObject method returns the parent of the root element, which is the PBDOM_DOCUMENT itself, and stores it in <i>pbdom_parent_obj</i>.</p> <p>The GetObjectClassString method returns the class name of <i>pbdom_parent_obj</i> as a string that is displayed in a message box:</p> <pre> pbdom_document pbdom_doc pbdom_object pbdom_obj pbdom_object pbdom_parent_obj string strClassName // code omitted ... pbdom_doc = pbdombuilder_new.BuildFromString (strXML) pbdom_obj = pbdom_doc.GetRootElement() pbdom_parent_obj = pbdom_obj.GetParentObject() strClassName = pbdom_parent_obj.GetObjectClassString() MessageBox ("Parent Class Name", strClassName) </pre>				
Usage	If the PBDOM_OBJECT has no parent, null is returned.				
See also	GetOwnerDocumentObject SetParentObject				

## GetText

Description Obtains the text data that is contained within the current PBDOM\_OBJECT.

Syntax *pbdom\_object\_name*.GetText()

Argument	Description
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT

Return value String.

The following table lists the return values, based on the type of DOM Object contained within a PBDOM\_OBJECT:

DOM Object Type	Return Value
PBDOM_ELEMENT	<p>The concatenation of the text values of all the TEXT nodes contained within the PBDOM_ELEMENT.</p> <p>If the PBDOM_ELEMENT definition is <code>&lt;abc&gt;Root Element Data&lt;data&gt;ABC Data &lt;/data&gt; now with extra info &lt;/abc&gt;</code>, then GetText returns "Root Element Data now with extra info".</p> <hr/> <p><b>Extra Spaces</b> There are extra spaces between the word "Data" and "now" and again after the word "info". They are there because they originally exist in the text.</p> <hr/> <p>If the PBDOM_ELEMENT definition is: <code>&lt;abc&gt;Root Element Data&lt;/abc&gt;</code>, then GetText returns "Root Element Data".</p>
PBDOM_ATTRIBUTE	<p>The text data contained within the PBDOM_ATTRIBUTE object.</p> <p>If the element with an attribute is <code>&lt;abc ATTRIBUTE_1="My Attribute"&gt;</code>, then GetText returns "My Attribute".</p>
PBDOM_TEXT	<p>The text data contained within the PBDOM_TEXT object itself.</p> <p>For example, suppose there is the following element:</p> <pre>&lt;abc&gt;MY TEXT&lt;/abc&gt;</pre> <p>If there is a PBDOM_TEXT object to represent the text node "MY TEXT", then calling GetText on the PBDOM_TEXT returns the string "MY TEXT"</p>

DOM Object Type	Return Value
PBDOM_CDATA	<p>The string data that is contained within the CDATA section itself. For example, suppose there is the following CDATA:</p> <pre>&lt;![CDATA[ They're saying "x &lt; y" &amp; that "z &gt; y" so I guess that means that z &gt; x ]]&gt;</pre> <p>If there is a PBDOM_CDATA to represent the above CDATA section, then calling GetText on it returns the following string:</p> <pre>They're saying "x &lt; y" &amp; that "z &gt; y" so I guess that means that z &gt; x</pre>
PBDOM_COMMENT	<p>The string data that is contained within the COMMENT itself. For example, suppose there is the following COMMENT:</p> <pre>&lt;!--This is some comment. --&gt;</pre> <p>If there is a PBDOM_COMMENT to represent the above COMMENT, then calling GetText on it returns the following string:</p> <pre>This is some comment.</pre>

Throws	<p>EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – This PBDOM_OBJECT object is not associated with a derived PBDOM_OBJECT class object.</p> <p>EXCEPTION_MEMORY_ALLOCATION_FAILURE – Insufficient memory was encountered while executing this method.</p>
Usage	<p>This method returns meaningful data only if the PBDOM_OBJECT is of a type that can contain text nodes, CDATA sections, or basic text. These include:</p> <ul style="list-style-type: none"> <li>• PBDOM_ELEMENT</li> <li>• PBDOM_ATTRIBUTE</li> <li>• PBDOM_TEXT</li> <li>• PBDOM_CDATA</li> <li>• PBDOM_COMMENT</li> </ul>

The PBDOM\_TEXT, PBDOM\_CDATA, and PBDOM\_COMMENT objects are special cases that cause the GetText method to return the text data that is intrinsically contained within the objects. A PBDOM\_TEXT object is basically a DOM text node and therefore does not hold any child text nodes. A PBDOM\_CDATA object represents a DOM CDATA object, and therefore does not hold any child DOM nodes. The same rule applies to a PBDOM\_COMMENT object.

See also                      GetTextNormalize  
                                   GetTextTrim

## GetTextNormalize

**Description**                      Gets the text data that is contained in the current PBDOM\_OBJECT with all surrounding whitespace characters removed and internal whitespace characters normalized to a single space.

**Syntax**                              *pbdom\_object\_name*.GetTextNormalize()

Argument	Description
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT

**Return value**                      String. The normalized text content of the current PBDOM\_OBJECT, or an empty string if there is no text content.

**Throws**                              EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object is not associated with a derived PBDOM\_OBJECT class object.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

**Usage**                                This method returns meaningful data only if the PBDOM\_OBJECT is of a type that can contain text nodes or CDATA sections, or of a type that intrinsically contains basic text. These types are:

- PBDOM\_ELEMENT
- PBDOM\_ATTRIBUTE
- PBDOM\_TEXT
- PBDOM\_CDATA
- PBDOM\_COMMENT



The PBDOM\_TEXT, PBDOM\_CDATA, and PBDOM\_COMMENT classes are special cases that cause the GetTextNormalize method to return the intrinsic text data contained within their instances. A PBDOM\_TEXT object represents a DOM text node, therefore it does not hold any child DOM Nodes. PBDOM\_CDATA object is a representation of a DOM CDATA object and does not hold any child DOM Nodes. Nor does PBDOM\_COMMENT contain any child DOM Nodes.

The following table lists the return values based on the type of actual DOM Object contained within PBDOM\_OBJECT:

DOM Object Type	Return Value
PBDOM_ELEMENT	<p>The normalized text of the concatenation of the text values of all the TEXT Nodes and CDATA Sections contained within the PBDOM_ELEMENT.</p> <p>Suppose there is a PBDOM_ELEMENT defined as follows:</p> <pre>&lt;abc&gt;    Root    Element Data    &lt;data&gt;ABC Data &lt;/data&gt; now with extra info    &lt;/abc&gt;</pre> <p>GetTextNormalize returns Root Element Data now with extra info.</p> <p>Suppose there is a PBDOM_ELEMENT defined as follows:</p> <pre>&lt;abc&gt; Root    Element    Data &lt;/abc&gt;</pre> <p>GetTextNormalize returns Root Element Data.</p> <p>Suppose there is a PBDOM_ELEMENT defined as follows:</p> <pre>&lt;abc&gt;    Root    Element    Data    &lt;![CDATA[ with some CDATA text]]&gt;&lt;/abc&gt;</pre> <p>GetTextNormalize returns "Root Element Data with some cdata text".</p>
PBDOM_ATTRIBUTE	<p>The normalized text data contained within the PBDOM_ATTRIBUTE object.</p> <p>Suppose there is an element with an attribute as follows:</p> <pre>&lt;abc ATTRIBUTE_1=" My    Attribute    "&gt;</pre> <p>GetTextNormalize returns My Attribute.</p>
PBDOM_TEXT	<p>The normalized text data contained within the PBDOM_TEXT object itself.</p> <p>For example, suppose there is the following element:</p> <pre>&lt;abc&gt; MY TEXT &lt;/abc&gt;</pre> <p>If there is a PBDOM_TEXT object to represent the text node "MY TEXT", then calling GetTextNormalize on the PBDOM_TEXT returns the string MY TEXT.</p>

DOM Object Type	Return Value
PBDOM_CDATA	<p>The normalized string data that is contained within the CDATA section itself. For example, suppose there is the following CDATA:</p> <pre>&lt;![CDATA[    They're saying "x &lt; y" &amp; that "z &gt; y" so I guess    that means that z &gt; x    ]]&gt;</pre> <p>If there is a PBDOM_CDATA to represent the above CDATA section, then calling <code>GetTextNormalize</code> on it returns the string:</p> <pre>They're saying " x &lt; y " &amp; that "z &gt; y" so I guess that means that z &gt; x</pre> <p>Note that the initial spaces before “They’re” and the trailing space after the last “x” have been removed. Additionally, the spaces between the word “guess” and “that” have been reduced to just one space.</p>
PBDOM_COMMENT	<p>The normalized string data that is contained within the COMMENT itself. For example, suppose there is the following COMMENT:</p> <pre>&lt;!--    Comment    Here !--&gt;</pre> <p>Calling <code>GetTextNormalize</code> on the COMMENT returns the string <code>Comment Here !</code></p>

See also

`GetText`  
`GetTextTrim`

## GetTextTrim

Description

Gets the text data that is contained in the current PBDOM\_OBJECT with all surrounding whitespace characters removed.

Syntax

*pbdom\_object\_name*.GetTextTrim()

Argument	Description
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT

Return value

String. The trimmed text content of the current PBDOM\_OBJECT, or an empty string if there is no text content or only whitespace characters.

Throws

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object is not associated with a derived PBDOM\_OBJECT class object.

EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE – Insufficient memory was encountered while executing this method.

Usage This method returns meaningful data only if the PBDOM\_OBJECT is of a type that can contain TEXT NODEs or CDATA Sections, or of a type that intrinsically contains basic text. These types are:

- PBDOM\_ELEMENT
- PBDOM\_ATTRIBUTE
- PBDOM\_TEXT
- PBDOM\_CDATA
- PBDOM\_COMMENT

The PBDOM\_TEXT, PBDOM\_CDATA, and PBDOM\_COMMENT classes are special cases that cause the GetTextTrim method to return the intrinsic text data contained within their instances. A PBDOM\_TEXT object represents a DOM text node, so it does not hold any child DOM Nodes. PBDOM\_CDATA object is a representation of a DOM CDATA object and does not hold any child DOM Nodes, nor does PBDOM\_COMMENT contain any child DOM Nodes.

The following table lists the return values based on the type of actual DOM Object contained within PBDOM\_OBJECT:

DOM Object Type	Return Value
PBDOM_ELEMENT	<p>The trimmed concatenation of the text values of all the TEXT Nodes and CDATA Sections contained within the PBDOM_ELEMENT. Surrounding whitespace characters are removed.</p> <p>Suppose there is a PBDOM_ELEMENT defined as follows:</p> <pre>&lt;abc&gt; Root Element Data&lt;data&gt;ABC Data &lt;/data&gt; now with extra info &lt;/abc&gt;</pre> <p>GetTextTrim returns Root Element Data now with extra info.</p> <p>Suppose there is a PBDOM_ELEMENT defined as follows:</p> <pre>&lt;abc&gt; Root Element Data &lt;/abc&gt;</pre> <p>GetTextTrim returns Root Element Data.</p> <p>Suppose there is a PBDOM_ELEMENT defined as follows:</p> <pre>&lt;abc&gt;Root Element Data &lt;![CDATA[ with some cdata text]]&gt;&lt;/abc&gt;</pre> <p>GetTextTrim returns Root Element Data with some cdata text.</p>

DOM Object Type	Return Value
PBDOM_ATTRIBUTE	<p>The trimmed text data contained within the PBDOM_ATTRIBUTE object with surrounding whitespace characters removed.</p> <p>Suppose there is an element with an attribute as follows:</p> <pre data-bbox="417 340 973 361">&lt;abc ATTRIBUTE_1="My    Attribute    "&gt;</pre> <p>GetTextTrim returns:</p> <pre data-bbox="417 413 637 434">My    Attribute</pre> <p>Note, however, that the spaces between “My” and “Attribute” are still present.</p>
PBDOM_TEXT	<p>The trimmed text data contained within the PBDOM_TEXT object itself with surrounding whitespace characters removed.</p> <p>For example, suppose there is the following element:</p> <pre data-bbox="417 591 717 612">&lt;abc&gt; MY TEXT &lt;/abc&gt;</pre> <p>If there is a PBDOM_TEXT object to represent the text node “MY TEXT”, then calling GetTextTrim on the PBDOM_TEXT returns the string MY TEXT.</p>
PBDOM_CDATA	<p>The trimmed string data that is contained within the CDATA section itself with surrounding whitespace characters removed. For example, suppose there is the following CDATA:</p> <pre data-bbox="417 786 1147 838">&lt;![CDATA[    They're saying "x &lt; y" &amp; that "z &gt; y" so I guess    that means that z &gt; x    ]]&gt;</pre> <p>If there is a PBDOM_CDATA to represent the above CDATA section, then calling GetTextTrim on it returns the string:</p> <pre data-bbox="417 916 1174 968">They're saying " x &lt; y " &amp; that "z &gt; y" so I guess    that means that z &gt; x</pre> <p>Note that the initial spaces before “They’re” and the trailing space after the last “x” have been removed.</p>
PBDOM_COMMENT	<p>The trimmed string data that is contained within the COMMENT itself. For example, suppose there is the following COMMENT:</p> <pre data-bbox="417 1116 798 1137">&lt;!--    Comment    Here !    --&gt;</pre> <p>Note the spaces before the word “Comment” and after the exclamation mark “!”. Calling GetTextTrim on the COMMENT returns the string Comment Here !</p>

See also

GetText  
GetTextNormalize

## HasChildren

**Description** Determines whether the PBDOM\_OBJECT has any child objects.

**Syntax** `pbdom_object_name.HasChildren()`

Argument	Description
<code>pbdom_object_name</code>	The name of the PBDOM_OBJECT

**Return value** Boolean. Returns true if the current PBDOM\_OBJECT has at least one child PBDOM\_OBJECT, and false if it has none.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object is not associated with a derived PBDOM\_OBJECT class object.

**Examples** In the following example, a PBDOM\_DOCUMENT is created from a simple XML string. The root element abc has a child text node that encapsulates the text “abc data”. Calling HasChildren on the root element returns true. The message box displays Has Children. If the method returns false, the message box displays Has No Children

```
PBDOM_Builder pbdombuilder_new
pbdom_document pbdom_doc
pbdom_object pbdom_root_element
string strXML = "<abc>abc data</abc>"

pbdombuilder_new = Create PBDOM_Builder
pbdom_doc = pbdombuilder_new.BuildFromString (strXML)
pbdom_root_element = pbdom_doc.GetRootElement()
if (pbdom_root_element.HasChildren()) then
    MessageBox ("pbdom_root_element", "Has Children")
else
    MessageBox ("pbdom_root_element", "Has No
Children")
end if
Destroy pbdombuilder_new
```

**Usage** True is returned if the PBDOM\_OBJECT has at least one child, and false if there are no children.

## InsertContent

Description Inserts a new PBDOM\_OBJECT into the current PBDOM\_OBJECT.

Syntax *pbdom\_object\_name.InsertContent(pbdom\_object\_new, pbdom\_object\_ref)*

Argument	Description
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT
<i>pbdom_object_new</i>	The referenced name of a PBDOM_OBJECT you want to insert
<i>pbdom_object_ref</i>	The name of the PBDOM_OBJECT in front of which you want to insert the new PBDOM_OBJECT

Return value PBDOM\_OBJECT. The return value is the newly modified PBDOM\_OBJECT.

Throws EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object or the new PBDOM\_OBJECT or the reference PBDOM\_OBJECT is not associated with a derived PBDOM\_OBJECT class object.

EXCEPTION\_INVALID\_ARGUMENT – One of the input arguments is invalid. This can happen if the input argument has not been initialized properly or is a null object reference.

Usage When a new PBDOM\_OBJECT is inserted into the current PBDOM\_OBJECT, the new PBDOM\_OBJECT becomes a child node of the current PBDOM\_OBJECT. Also, the new PBDOM\_OBJECT is to be positioned specifically before another PBDOM\_OBJECT, designated using the second parameter.

If the second PBDOM\_OBJECT is specified as null, then the new PBDOM\_OBJECT is to be inserted at the end of the list of children of the current PBDOM\_OBJECT.

---

### Derived Classes

Methods of classes that derive from the PBDOM\_OBJECT class return trivial results when the derived classes can have no child objects and when the methods concern manipulation of child-node content.

---

See also AddContent  
GetContent  
RemoveContent  
SetContent

## IsAncestorObjectOf

Description	Determines whether the current PBDOM_OBJECT is the ancestor of another PBDOM_OBJECT.						
Syntax	<i>pbdom_object_name</i> .IsAncestorObjectOf( <i>pbdom_object_ref</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_object_name</i></td> <td>The name of the PBDOM_OBJECT</td> </tr> <tr> <td><i>pbdom_object_ref</i></td> <td>The PBDOM_OBJECT to check against</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT	<i>pbdom_object_ref</i>	The PBDOM_OBJECT to check against
Argument	Description						
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT						
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to check against						
Return value	Boolean. Returns true if the current PBDOM_OBJECT is the ancestor of the referenced PBDOM_OBJECT, and false otherwise.						
Throws	<p>EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – This PBDOM_OBJECT object is not associated with a derived PBDOM_OBJECT class object.</p> <p>EXCEPTION_INVALID_ARGUMENT – The input PBDOM_OBJECT is invalid. This can happen if it has not been initialized properly or it is a null object reference.</p>						
Examples	<p>The following code fragment uses the IsAncestorObjectOf method and creates a structured document. In the fragment, <i>pbdom_elem_1</i> represents the <i>pbdom_elem_1</i> element. Because it is an ancestor of <i>pbdom_elem_3</i>, which represents the <i>pbdom_elem_3</i> element, the call to IsAncestorObjectOf returns true.</p>						

```

PBDOM_ELEMENT pbdom_elem_1
PBDOM_ELEMENT pbdom_elem_2
PBDOM_ELEMENT pbdom_elem_3
PBDOM_ELEMENT pbdom_elem_root
PBDOM_DOCUMENT pbdom_doc1

pbdom_doc1 = Create PBDOM_DOCUMENT
pbdom_elem_1 = Create PBDOM_ELEMENT
pbdom_elem_2 = Create PBDOM_ELEMENT
pbdom_elem_3 = Create PBDOM_ELEMENT

pbdom_elem_1.SetName("pbdom_elem_1")
pbdom_elem_2.SetName("pbdom_elem_2")
pbdom_elem_3.SetName("pbdom_elem_3")

pbdom_elem_1.AddContent(pbdom_elem_2)
pbdom_elem_2.AddContent(pbdom_elem_3)

```

```
pbdom_doc1.NewDocument("", "", &
    "Root_Element_From_Doc_1" , "", "")
pbdom_elem_root = pbdom_doc1.GetRootElement()
pbdom_elem_root.AddContent(pbdom_elem_1)

IF (pbdom_elem_1.IsAncestorObjectOf(pbdom_elem_3))
THEN
    MessageBox ("Ancestry", &
        "pbdom_elem_1 Is The Ancestor Of pbdom_elem_3")
ELSE
    MessageBox ("Ancestry", &
        "pbdom_elem_1 Is NOT The Ancestor Of pbdom_elem_3")

END IF

destroy pbdom_elem_1
destroy pbdom_elem_2
destroy pbdom_elem_3
destroy pbdom_elem_root
destroy pbdom_doc1
```

The preceding code fragment creates the following document:

```
<!DOCTYPE Root_Element_From_Doc_1>
<Root_Element_From_Doc_1>
  <pbdom_elem_1>
    <pbdom_elem_2>
      <pbdom_elem_3 />
    </pbdom_elem_2>
  </pbdom_elem_1>
</Root_Element_From_Doc_1>
```

#### Usage

The `IsAncestorObjectOf` method determines whether the current `PBDOM_OBJECT` is the ancestor of another `PBDOM_OBJECT`.



## RemoveContent

Description	Removes a child PBDOM_OBJECT from the current PBDOM_OBJECT.						
Syntax	<i>pbdom_object_name.RemoveContent(pbdom_object_ref)</i>						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_object_name</i></td> <td>The name of the PBDOM_OBJECT</td> </tr> <tr> <td><i>pbdom_object_ref</i></td> <td>The PBDOM_OBJECT to remove</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT	<i>pbdom_object_ref</i>	The PBDOM_OBJECT to remove
Argument	Description						
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT						
<i>pbdom_object_ref</i>	The PBDOM_OBJECT to remove						
Return value	Boolean. Returns true if the content was removed, and false otherwise.						
Throws	<p>EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – This PBDOM_OBJECT object or the input PBDOM_OBJECT is not associated with a derived PBDOM_OBJECT class object.</p> <p>EXCEPTION_INVALID_ARGUMENT – The input PBDOM_OBJECT to be removed is invalid. This can happen if this object has not been initialized properly or is a null object reference.</p>						
Usage	When a new PBDOM_OBJECT is removed from the current one, all children under the removed PBDOM_OBJECT are also removed.						
See also	<p>AddContent</p> <p>GetContent</p> <p>InsertContent</p> <p>SetContent</p>						

## SetContent

Description Sets the entire content of the PBDOM\_OBJECT.

Syntax *pbdom\_object\_name*.SetContent(*pbdom\_object**pbdom\_object\_array*)

Argument	Description
<i>pbdom_object_name</i>	The name of the PBDOM object
<i>pbdom_object_array</i>	An array of PBDOM_OBJECT objects to be set as the contents of the PBDOM_OBJECT

Return value PBDOM\_OBJECT. Returns the newly modified PBDOM\_OBJECT.

Throws EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object is not associated with a derived PBDOM\_OBJECT class object.

Usage The supplied array contains PBDOM\_OBJECT objects that are legal for the particular derived PBDOM\_OBJECT that is associated with this PBDOM\_OBJECT.

For example, a PBDOM\_DOCUMENT accepts only an array that contains PBDOM\_ELEMENT, PBDOM\_COMMENT, PBDOM\_DOCTYPE, or PBDOM\_PROCESSINGINSTRUCTION objects. In addition, the array can contain only one PBDOM\_ELEMENT object that it sets as its root element, and only one PBDOM\_DOCTYPE object that is set as its DOCTYPE.

If illegal objects are included in the array, exceptions (specific to the particular derived PBDOM\_OBJECT) are thrown. For more details, please refer to the SetContent method of the objects derived from PBDOM\_OBJECT.

In the event of an exception, the original contents of this PBDOM\_OBJECT are unchanged, and the PBDOM\_OBJECT objects contained in the supplied array are unaltered.

See also  
AddContent  
GetContent  
InsertContent  
RemoveContent

## SetName

Description	Sets the name of the PBDOM_OBJECT.						
Syntax	<i>pbdom_object_name</i> .SetName(string <i>strName</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_object_name</i></td> <td>The name of the PBDOM_OBJECT</td> </tr> <tr> <td><i>strName</i></td> <td>The new name you want to set for PBDOM_OBJECT</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT	<i>strName</i>	The new name you want to set for PBDOM_OBJECT
Argument	Description						
<i>pbdom_object_name</i>	The name of the PBDOM_OBJECT						
<i>strName</i>	The new name you want to set for PBDOM_OBJECT						
Return value	Boolean. Returns true if the name of the PBDOM_OBJECT was changed, and false otherwise.						
Throws	<p>EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – This PBDOM_OBJECT object is not associated with a derived PBDOM_OBJECT class object.</p> <p>EXCEPTION_INVALID_ARGUMENT – Input name string is invalid. This can happen if the string has been specifically set to null.</p> <p>EXCEPTION_MEMORY_ALLOCATION_FAILURE – Insufficient memory was encountered while executing this method.</p> <p>EXCEPTION_INVALID_NAME – The input name string does not conform to the W3C standards for XML names.</p>						
Usage	<p>This name refers to the name of the particular derived PBDOM_OBJECT to which this PBDOM_OBJECT refers. Certain types of PBDOM_OBJECT do not have any name associated with them. See the description of GetName.</p> <p>For example, PBDOM_DOCUMENT does not have any name, so calling the SetName method returns false.</p>						
See also	GetName						

## SetParentObject

**Description** Sets the referenced PBDOM\_OBJECT as the parent of the current PBDOM\_OBJECT.

**Syntax** `pbdom_object_name.SetParentObject(pbdom_object pbdom_object_ref)`

Argument	Description
<code>pbdom_object_name</code>	The name of the PBDOM_OBJECT
<code>pbdom_object_ref</code>	The PBDOM_OBJECT to be set as the parent of the current PBDOM_OBJECT

**Return value** PBDOM\_OBJECT. The current PBDOM\_OBJECT is appended as a child node of the referenced parent.

**Throws**

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – This PBDOM\_OBJECT object or the input PBDOM\_OBJECT is not associated with a derived PBDOM\_OBJECT class object.

EXCEPTION\_INVALID\_ARGUMENT – The input PBDOM\_OBJECT is invalid. This can happen if it has not been initialized properly, or if it is a null object reference.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – The current PBDOM\_OBJECT already has a parent.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT is of a class that cannot have a legal parent-child relationship with this PBDOM\_OBJECT.

**Examples**

In the following code example, a PBDOM\_ELEMENT object is created and called `pbdom_elem_1`. Its parent is set to be the root element of the PBDOM\_DOCUMENT called `pbdom_doc`. Once this is done, `pbdom_elem_1` is immediately transferred to the `pbdom_doc` document and `pbdom_elem_1` is immediately appended as a child node of the root element of `pbdom_doc`.

The following method call returns the string “`pbdom_element`”, because the root element is a PBDOM\_ELEMENT:

```
pbdom_elem_1.GetParentObject().GetObjectClassString()
```

The following method call returns the string “`Root_Element`”, which is the name of the root element:

```
pbdom_elem_1.GetParentObject().GetName()
```

Here is the complete example:

```
PBDOM_ELEMENT pbdom_elem_1
PBDOM_ELEMENT pbdom_elem_root
PBDOM_DOCUMENT pbdom_doc1

pbdom_doc1 = Create PBDOM_DOCUMENT
pbdom_elem_1 = Create PBDOM_ELEMENT
pbdom_elem_1.SetName ("pbdom_elem_1")

pbdom_doc1.NewDocument ("", "", "Root_Element", "", "")
pbdom_elem_root = pbdom_doc1.GetRootElement()
pbdom_elem_1.SetParentObject (pbdom_elem_root)

MessageBox ("Parent Class", &
    pbdom_elem_1.GetParentObject(). &
    GetObjectClassString())
MessageBox ("Parent Name", &
    pbdom_elem_1.GetParentObject().GetName())

destroy pbdom_elem_1
destroy pbdom_elem_root
destroy pbdom_doc1
```

#### Usage

The caller is responsible for ensuring that the current PBDOM\_OBJECT and the referenced PBDOM\_OBJECT can have a legal parent-child relationship. The caller is also responsible for making sure pre-existing parentage is legal.

The PBDOM SetParentObject method differs from the JDOM SetParent method in that JDOM defines a setParent method for several specific classes, including Element, Comment, and CDATA. PBDOM implements the SetParentObject method in the base PBDOM\_OBJECT class to allow polymorphism.

See the SetParentObject documentation of derived PBDOM\_OBJECT classes for more details on implementation of specific classes.

#### See also

GetOwnerDocumentObject  
GetParentObject



# PBDOM\_PROCESSINGINSTRUCTION Class

About this chapter

This chapter describes the PBDOM\_PROCESSINGINSTRUCTION class.

## PBDOM\_PROCESSINGINSTRUCTION

Description

The PBDOM\_PROCESSINGINSTRUCTION class defines behavior for an XML processing instruction. Methods allow you to obtain the target of the processing instruction object as well as its data. You can always access the data as a string, and, where appropriate, as name/value pairs.

Note that the actual processing instruction of a processing instruction object is a string, even if the instruction is divided into separate name="value" pairs. PBDOM does support such a processing instruction object format. If the processing instruction object data does contain pairs, as is commonly the case, then PBDOM\_PROCESSINGINSTRUCTION parses them into an internal list of name/value pairs.

Methods

Some of the inherited methods from PBDOM\_OBJECT serve no meaningful objective, and only default or trivial functionalities result. These are described in the following table:

Method	Always returns
AddContent	Current PBDOM_PROCESSINGINSTRUCTION. Use AddValue instead.
GetContent	false. Use GetName and GetValue instead.
HasChildren	false.
InsertContent	Current PBDOM_PROCESSINGINSTRUCTION.
IsAncestorObjectOf	false.
RemoveContent	false. Use RemoveValue instead.
SetContent	Current PBDOM_PROCESSINGINSTRUCTION. Use SetData instead.

PBDOM\_PROCESSINGINSTRUCTION has the following methods:

Clone	GetTarget
Detach	GetText
Equals	GetTextNormalize
GetData	GetTextTrim
GetName	GetValue
GetNames	RemoveValue
GetObjectClass	SetData
GetObjectClassString	SetName
GetOwnerDocumentObject	SetParentObject
GetParentObject	SetValue

## Clone

Description

Creates and returns a clone of the current PBDOM\_PROCESSINGINSTRUCTION object.

Syntax

*pbdom\_pi\_name*.Clone(boolean *bDeep*)

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object.
<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are true for a deep clone and false for a shallow clone. This argument is currently ignored.

Return value

PBDOM\_OBJECT. A clone of the current PBDOM\_PROCESSINGINSTRUCTION object returned as a PBDOM\_OBJECT.

Throws

EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the internal implementation of this PBDOM\_PROCESSINGINSTRUCTION object is null. The occurrence of this exception is rare, but it can take place if severe memory corruption occurs.

Usage

The Clone method creates a new PBDOM\_PROCESSINGINSTRUCTION object that is a duplicate of, and a separate object from, the original. The clone of a PBDOM\_PROCESSINGINSTRUCTION object is always identical to its original whether *bDeep* is true or false, because a PBDOM\_PROCESSINGINSTRUCTION object contains no subtree of child PBDOM\_OBJECTs.

A PBDOM\_PROCESSINGINSTRUCTION clone has no parent, but it resides in the same PBDOM\_DOCUMENT as its original, and if the original PBDOM\_PROCESSINGINSTRUCTION object is standalone, so is the clone.



## Detach

**Description** Detaches a PBDOM\_PROCESSINGINSTRUCTION object from its parent PBDOM\_OBJECT.

**Syntax** *pbdom\_pi\_name*.Detach()

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object

**Return value** PBDOM\_OBJECT. This PBDOM\_PROCESSINGINSTRUCTION object detached from its parent object. This method does nothing if this PBDOM\_PROCESSINGINSTRUCTION object has no parent.

## Equals

**Description** Tests for the equality of the current PBDOM\_PROCESSINGINSTRUCTION object with the supplied PBDOM\_OBJECT.

**Syntax** *pbdom\_pi\_name*.Equals(*pbdom\_object\_ref*)

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object
<i>pbdom_object_ref</i>	A PBDOM_OBJECT for testing for equality with the current PBDOM_PROCESSINGINSTRUCTION object

**Return value** Boolean. Returns true if the current PBDOM\_PROCESSINGINSTRUCTION object is equivalent to the input PBDOM\_OBJECT, and false otherwise.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not a reference to an object derived from PBDOM\_OBJECT.

## GetData

Description Returns the raw data of the PBDOM\_PROCESSINGINSTRUCTION object.

Syntax *pbdom\_pi\_name*.GetData()

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object

Return value String. The data of the PBDOM\_PROCESSINGINSTRUCTION object.

Usage The processing instruction data is fundamentally a string and *not* a set of name="value" pairs.

## GetName

Description Obtains the name of the current PBDOM\_PROCESSINGINSTRUCTION object.

Syntax *pbdom\_pi\_name*.GetName()

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object

Return value String.

Examples Calling the GetName method on the following processing instruction returns works:

```
<?works document="hello.doc" data="hello.wks" ?>
```

Usage This method is similar to the GetTarget method. To PBDOM, the processing instruction target is synonymous with its name.

## GetNames

**Description** Retrieves a list of names taken from the part of the PBDOM\_PROCESSINGINSTRUCTION object's data that is factored into name="value" pairs. This method can be used in conjunction with the GetValue method.

**Syntax** `pbdom_pi_name.GetNames(string name_array[ ])`

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object
<i>name_array</i>	An unbounded string array filled with names

**Return value** Boolean. Returns true if a list of names is retrieved, and false otherwise. If there are no name/value pairs, this method returns false.

**Examples** Given the following PBDOM\_PROCESSINGINSTRUCTION object, GetNames returns three strings, a, b, and c, even though a occurs more than once:

```
<? dw-set_values a="1" b="2" c="3" a="4" ?>
```

When the GetValue method is called on a, the value 4 is returned, because it is the last value set for a.

**Usage** If a name is used more than once as the name of a name/value pair in a PBDOM\_PROCESSINGINSTRUCTION object, then the value set in the last occurrence of the name is used, and values declared in all previous occurrences of the name are discarded.

## GetObjectClass

**Description** Returns a long integer code that indicates the class of the current PBDOM\_PROCESSINGINSTRUCTION object.

**Syntax** `pbdom_pi_name.GetObjectClass()`

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_OBJECT

**Return value** Long. GetObjectClass returns a long integer code that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_pi\_name* is a PBDOM\_PROCESSINGINSTRUCTION object, the returned value is 10.

## GetObjectClassString

**Description** Returns a string form of the class of the PBDOM\_PROCESSINGINSTRUCTION object.

**Syntax** *pbdom\_pi\_name*.GetObjectClassString()

<b>Argument</b>	<b>Description</b>
<i>pbdom_pi_name</i>	The name of a PBDOM_OBJECT

**Return value** String. GetObjectClassString returns a string that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_pi\_name* is a PBDOM\_PROCESSINGINSTRUCTION, the returned string is “pbdom\_processinginstruction”.

## GetOwnerDocumentObject

**Description** Returns the owning PBDOM\_DOCUMENT of the current PBDOM\_PROCESSINGINSTRUCTION object.

**Syntax** *pbdom\_pi\_name*.GetOwnerDocumentObject()

<b>Argument</b>	<b>Description</b>
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object

**Return value** PBDOM\_DOCUMENT. If there is no owning PBDOM\_DOCUMENT, null is returned.

## GetParentObject

**Description** Returns the parent PBDOM\_OBJECT of the current PBDOM\_PROCESSINGINSTRUCTION object.

**Syntax** *pbdom\_pi\_name*.GetParentObject()

<b>Argument</b>	<b>Description</b>
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object

**Return value** PBDOM\_OBJECT. The parent of the PBDOM\_PROCESSINGINSTRUCTION object. If there is no parent, null is returned.

## GetTarget

**Description** Returns the target of the PBDOM\_PROCESSINGINSTRUCTION object.

**Syntax** `pbdom_pi_name.GetTarget()`

Argument	Description
<code>pbdom_pi_name</code>	The name of a PBDOM_PROCESSINGINSTRUCTION object

**Return value** String. The target of the PBDOM\_PROCESSINGINSTRUCTION object.

**Examples** Given the following PBDOM\_PROCESSINGINSTRUCTION object, calling the `GetTarget` method returns the string “xmlstylesheet”:

```
<?xml-stylesheet href="simple-ie5.xsl" type="text/xsl"
?>
```

Calling the `GetName` method returns the same string.

**See also** `GetName`

## GetText

**Description** Obtains text data that is contained within the current PBDOM\_PROCESSINGINSTRUCTION object.

**Syntax** `pbdom_pi_name.GetText()`

Argument	Description
<code>pbdom_pi_name</code>	The name of a PBDOM_PROCESSINGINSTRUCTION object

**Return value** String.

**Usage** The `GetText` method returns the text data of the current PBDOM\_PROCESSINGINSTRUCTION object. `GetText` is similar to `GetData`. However, the textual content of a processing instruction object is not a text node.

**See also** `GetData`  
`GetTextNormalize`  
`GetTextTrim`  
`SetData`

## GetTextNormalize

**Description** Obtains the text data that is contained within the current PBDOM\_PROCESSINGINSTRUCTION object with all surrounding whitespace characters removed and internal whitespace characters normalized to a single space.

**Syntax** *pbdom\_pi\_name*.GetTextNormalize()

<b>Argument</b>	<b>Description</b>
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object

**Return value** String. The normalized text content of the PBDOM\_PROCESSINGINSTRUCTION object. If no textual value exists for the current PBDOM\_OBJECT, or if only whitespace characters exist, an empty string is returned.

**See also** GetData  
GetText  
GetTextTrim  
SetData

## GetTextTrim

**Description** Obtains the text data that is contained within the current PBDOM\_PROCESSINGINSTRUCTION object with all surrounding whitespaces removed.

**Syntax** *pbdom\_pi\_name*.GetTextTrim()

<b>Argument</b>	<b>Description</b>
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object

**Return value** String. The trimmed text content of the PBDOM\_PROCESSINGINSTRUCTION object. If no textual value exists for the current PBDOM\_PROCESSINGINSTRUCTION object, or if only whitespace characters exist, an empty string is returned.

**See also** GetData  
GetText  
GetTextNormalize  
SetData

## GetValue

**Description** Returns the value for a specific name/value pair on the PBDOM\_PROCESSINGINSTRUCTION object. If no such pair is found for the PBDOM\_PROCESSINGINSTRUCTION object, an empty string is returned.

**Syntax** `pbdom_pi_name.GetValue(string strName)`

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object
<i>strName</i>	String name of name/value pair

**Return value** String. String name of the name/value pair to search for value.

**Examples** Given the following PBDOM\_PROCESSINGINSTRUCTION object, `GetValue("href")` returns the string "simple-ie5.xml":

```
<?xml-stYLESHEET href="simple-ie5.xml" type="text/xsl"
?>
```

**See also** `GetData`, `GetText`, `SetValue`

## RemoveValue

**Description** Removes the specified name/value pair.

**Syntax** `pbdom_pi_name.RemoveValue(string strName)`

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object
<i>strName</i>	String name of name/value pair to be removed

**Return value** Boolean. Returns true if the requested name/value pair is removed and false otherwise.

**Examples** Suppose the following PBDOM\_PROCESSINGINSTRUCTION object is given:

```
<?xml-stYLESHEET href="simple-ie5.xml" type="text/xsl"
?>
```

Then, `RemoveValue("href")` results in the PBDOM\_PROCESSINGINSTRUCTION object being transformed into the following:

```
<?xml-stYLESHEET type="text/xsl" ?>
```

## SetData

Description Sets the raw data for the PBDOM\_PROCESSINGINSTRUCTION object.

Syntax *pbdom\_pi\_name*.SetData(string *strData*)

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object
<i>strData</i>	New data for the PBDOM_PROCESSINGINSTRUCTION object

Return value PBDOM\_PROCESSINGINSTRUCTION. The PBDOM\_PROCESSINGINSTRUCTION object modified with the new data.

Throws EXCEPTION\_INVALID\_STRING – The input data is invalid. This can happen in the following circumstances:

- 1 The input data contains the sub-string “?>”. This violates the requirements for the data of a processing instruction.
- 2 If the processing instruction target name is `xml`, making this PBDOM\_PROCESSINGINSTRUCTION object an XML declaration processing instruction, this exception is thrown if the input data string does not conform to the following criteria:
  - The data must contain a name/value pair for the name `version`.
  - The data can contain a name/value pair for the name `encoding`.
  - The data can contain a name/value pair for the name `standalone`. If it does, the value for `standalone` must either be `yes` or `no`.
  - The data must not contain any other data in the form of name/value pairs or in any other form.

---

### Lowercase

The strings `xml`, `version`, `encoding`, `standalone`, `yes`, and `no` are all case sensitive and must be in lowercase.

---

Examples Suppose there is a PBDOM\_PROCESSINGINSTRUCTION object as follows:

```
<?xml-stylesheet href="simple-ie5.xsl" type="text/xsl"
?>
```



Then, `SetData("href=new.xsl")` results in the `PBDOM_PROCESSINGINSTRUCTION` object being transformed into the following:

```
<?xml-stylesheet href=new.xsl" ?>
```

The entire data for the `PBDOM_PROCESSINGINSTRUCTION` object is now reset.

#### Usage

Special processing is performed when the name of the processing instruction's target is `xml`, which indicates that it is an XML declaration. The valid instructions allowed in the input Data as part of the name in the name/value pairs are `version`, `encoding`, and `standalone`. The version instruction is mandatory before the processing instruction can be added to a document.

The XML specification expects the instructions to be in the specific order `version`, `encoding`, `standalone`. This function reorders the input data to conform to the specification, for example:

```
<? xml version="1.0" encoding="utf-8"
standalone="yes"?>
```

## SetName

#### Description

Sets the name of the current `PBDOM_PROCESSINGINSTRUCTION` object.

#### Syntax

```
pbdom_pi_name.SetName(string strName)
```

Argument	Description
<i>pbdom_pi_name</i>	The name of a <code>PBDOM_PROCESSINGINSTRUCTION</code> object
<i>strName</i>	The new name you want to set for the current <code>PBDOM_PROCESSINGINSTRUCTION</code> object

#### Return value

Boolean. Returns true if the name of the current `PBDOM_PROCESSINGINSTRUCTION` object was changed, and false otherwise.

#### Throws

`EXCEPTION_INVALID_NAME` – This exception is thrown if the name is invalid. The name can be `xml`, making this `PBDOM_PROCESSINGINSTRUCTION` object an XML declaration processing instruction. However, in this case, the name `xml` must be in lowercase, or the `EXCEPTION_INVALID_NAME` exception will be thrown.

EXCEPTION\_INVALID\_STRING – This exception is thrown if the name is `xml` and the current data of this PBDOM\_PROCESSINGINSTRUCTION object is not valid. The data is valid only under the following circumstances:

- It is an empty string.
- If it is not an empty string, it must contain a name/value pair for the name `version`.
- If it is not an empty string and it contains a name/value pair for the name `version`, it can also contain a name/value pair for the name `encoding`.
- If it is not an empty string and it contains a name/value pair for the name `version`, it can also contain a name/value pair for the name `standalone`. If it does, the value for `standalone` must be either `yes` or `no` (both are case sensitive).
- If it is not an empty string and it contains a name/value pair for the name `version`, it must not contain any other data (in name/value pair format or otherwise) except for `encoding` and `standalone`.

**Usage** This method is equivalent to setting the target of the processing instruction object. See the list of exceptions for information about the restrictions on the use of `xml` as the target.

## SetParentObject

**Description** Sets the referenced PBDOM\_OBJECT to be the parent of the current PBDOM\_PROCESSINGINSTRUCTION object.

**Syntax** `pbdom_pi_name.SetParentObject(pbdom_object pbdom_object_ref)`

Argument	Description
<code>pbdom_pi_name</code>	The name of a PBDOM_PROCESSINGINSTRUCTION object
<code>pbdom_object_ref</code>	A PBDOM_OBJECT to be set as the parent of the current PBDOM_PROCESSINGINSTRUCTION object

**Return value** PBDOM\_OBJECT. This PBDOM\_PROCESSINGINSTRUCTION object modified.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not a reference to an object derived from PBDOM\_OBJECT.

EXCEPTION\_HIERARCHY\_ERROR – If setting the input PBDOM\_OBJECT to be the parent of this PBDOM\_PROCESSINGINSTRUCTION object will cause the parent PBDOM\_OBJECT to be no longer well formed. For example, if this PBDOM\_PROCESSINGINSTRUCTION object is an XML declaration and the parent to be set is a PBDOM\_ELEMENT.

#### Usage

The PBDOM\_OBJECT that you set as the parent and the current PBDOM\_PROCESSINGINSTRUCTION object must have a legal parent-child relationship. Currently, only a PBDOM\_ELEMENT and a PBDOM\_DOCUMENT can be set as the parent of a PBDOM\_PROCESSINGINSTRUCTION object.

## SetValue

#### Description

Sets the value for the specified name/value pair.

#### Syntax

*pbdom\_pi\_name*.SetValue(string *strName*, string *strValue*)

Argument	Description
<i>pbdom_pi_name</i>	The name of a PBDOM_PROCESSINGINSTRUCTION object
<i>strName</i>	String name of a name/value pair
<i>strValue</i>	String value of a name/value pair

#### Return value

PBDOM\_PROCESSINGINSTRUCTION.

#### Throws

EXCEPTION\_INVALID\_STRING – The input *strName*/*strValue* is invalid. This can happen in the following circumstances:

- The input *strName*/*strValue* data contains the sub-string ?>. This violates the requirements for the data of a processing instruction.
- If the target name is `xml`, making this PBDOM\_PROCESSINGINSTRUCTION object an XML declaration processing instruction, this exception is thrown if the input data string does not conform to the following criterion: the data can contain a name/value pair for the name `standalone`. If it does, the value for `standalone` must either be `yes` or `no`. The strings `xml`, `standalone`, `yes`, and `no` are case sensitive and must be lowercase.

EXCEPTION\_INVALID\_NAME - The input *strName* is invalid. This can happen if the target name is `xml`, making this PBDOM\_PROCESSINGINSTRUCTION object an XML declaration processing instruction, and either of the following is true:

- The *strName* value is other than `version`, `standalone` or `encoding`.
- Either `standalone` or `encoding` is set without the `version` first being set.

## Examples

Consider the following PBDOM\_PROCESSINGINSTRUCTION object:

```
<?xml-stylesheet href="simple-ie5.xsl" type="text/xsl" ?>
```

`SetValue("href", "new.xsl")` transforms this processing instruction to the following, modifying the value for *href*:

```
<?xml-stylesheet href="new.xsl" type="text/xsl"?>
```

`SetValue("extra_info", "xalan")` transforms the processing instruction to the following, adding a new name/value pair for *extra\_info*:

```
<?xml-stylesheet href=new.xsl" type="text/xsl"
extra_info "xalan" ?>
```

Then `SetValue("extra_info_2", "")` transforms the processing instruction to the following, adding a new name/value pair for *extra\_info\_2* with an empty string as the value:

```
<?xml-stylesheet href=new.xsl" type="text/xsl"
extra_info="xalan" extra_info_2="" ?>
```

## Usage

If no value is found, the supplied pair is added to the processing instruction data. The appearance of name/value pairs in a PBDOM\_PROCESSINGINSTRUCTION object is not subject to any order. In this way, name/value pairs in a PBDOM\_PROCESSINGINSTRUCTION object are similar to attributes in an element. Attributes are specifically *not* ordered.

Special processing is performed when the name of the processing instruction's target is `xml`, which indicates that it is an XML declaration. The valid instructions allowed in the input Data as part of the name in the name/value pairs are `version`, `encoding`, and `standalone`. The `version` instruction is mandatory before the processing instruction can be added to a document.

The XML specification expects the instructions to be in this specific order: `version`, `encoding`, `standalone`. This function reorders the input data to conform to the specification, for example:

```
<? xml version="1.0" encoding="utf-8" standalone="yes"?>
```

About this chapter

This chapter describes the PBDOM\_TEXT class.

## PBDOM\_TEXT

Description

The PBDOM\_TEXT class represents a DOM Text Node within an XML document. It extends the PBDOM\_CHARACTERDATA class with a set of methods specifically intended for manipulating DOM text nodes.

The PBDOM\_TEXT class is derived from the PBDOM\_CHARACTERDATA class. PBDOM\_TEXT objects are commonly used to represent the textual content of a PBDOM\_ELEMENT or PBDOM\_ATTRIBUTE.

---

### Whitespace characters

The text in a PBDOM\_TEXT object can include whitespace characters such as carriage returns, linefeeds, tabs, and spacebar spaces.

---

Methods

Some of the inherited methods from PBDOM\_OBJECT serve no meaningful objective, and only default or trivial functionalities result. These are described in the following table:

Method	Always returns
AddContent	current PBDOM_TEXT
GetContent	false
GetName	a string “#text”
HasChildren	false
InsertContent	current PBDOM_TEXT
IsAncestorObjectOf	false
RemoveContent	false
SetContent	current PBDOM_TEXT
SetName	false

PBDOM\_TEXT has the following non-trivial methods:

Append	GetParentObject
Clone	GetText
Detach	GetTextNormalize
Equals	GetTextTrim
GetObjectClass	SetParentObject
GetObjectClassString	SetText
GetOwnerDocumentObject	

## Append

Description

The Append method is overloaded:

- Syntax 1 appends an input string to the text content that already exists within the current PBDOM\_TEXT object.
- Syntax 2 appends the text data of a PBDOM\_CHARACTERDATA object to the text content that already exists within the current PBDOM\_TEXT object.

Syntax

For this syntax	See
Append(string <i>strAppend</i> )	Append Syntax 1
Append(pbdom_characterdata <i>pbdom_characterdata_ref</i> )	Append Syntax 2

## Append Syntax 1

Description

Appends an input string to the text content that already exists within the current PBDOM\_TEXT object.

Syntax

*pbdom\_text\_name*.Append(string *strAppend*)

Argument	Description
<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object
<i>strAppend</i>	The string you want appended to the existing text of the current PBDOM_TEXT object

Return value

PBDOM\_CHARACTERDATA. The current PBDOM\_TEXT object modified and returned as a PBDOM\_CHARACTERDATA object.

## Append Syntax 2

Description	Appends the text data of a PBDOM_CHARACTERDATA object to the text content that already exists within the current PBDOM_TEXT object.						
Syntax	<i>pbdom_text_name</i> .Append( <i>pbdom_characterdata pbdom_characterdata_ref</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_text_name</i></td> <td>The name of a PBDOM_TEXT object</td> </tr> <tr> <td><i>pbdom_characterdata_ref</i></td> <td>The referenced PBDOM_CHARACTERDATA object whose text data is to be appended to the existing text of the current PBDOM_TEXT object</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object	<i>pbdom_characterdata_ref</i>	The referenced PBDOM_CHARACTERDATA object whose text data is to be appended to the existing text of the current PBDOM_TEXT object
Argument	Description						
<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object						
<i>pbdom_characterdata_ref</i>	The referenced PBDOM_CHARACTERDATA object whose text data is to be appended to the existing text of the current PBDOM_TEXT object						
Return value	PBDOM_CHARACTERDATA. The current PBDOM_TEXT object modified and returned as a PBDOM_CHARACTERDATA object.						
Throws	EXCEPTION_PBDOM_OBJECT_INVALID_FOR_USE – If the input PBDOM_CHARACTERDATA is not a reference to an object inherited from PBDOM_CHARACTERDATA.						
Usage	Note that JDOM does not define an Append method for its TEXT class. Because PBDOM implements its Append method in the base PBDOM_CHARACTERDATA class, a PBDOM_COMMENT object, a PBDOM_CDATA object, and a PBDOM_TEXT object can append their internal text data to each other, because they are all objects inherited from PBDOM_CHARACTERDATA.						

## Clone

Description	Creates and returns a clone of the current PBDOM_TEXT object.						
Syntax	<i>pbdom_text_name</i> .Clone( <i>boolean bDeep</i> )						
	<table border="1"> <thead> <tr> <th>Argument</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>pbdom_text_name</i></td> <td>The name of a PBDOM_TEXT object.</td> </tr> <tr> <td><i>bDeep</i></td> <td>A boolean specifying whether a deep or shallow clone is returned. Values are <code>true</code> for a deep clone and <code>false</code> for a shallow clone. This parameter is ignored.</td> </tr> </tbody> </table>	Argument	Description	<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object.	<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are <code>true</code> for a deep clone and <code>false</code> for a shallow clone. This parameter is ignored.
Argument	Description						
<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object.						
<i>bDeep</i>	A boolean specifying whether a deep or shallow clone is returned. Values are <code>true</code> for a deep clone and <code>false</code> for a shallow clone. This parameter is ignored.						
Return value	PBDOM_OBJECT. The return value is a clone of the current PBDOM_TEXT object returned as a PBDOM_OBJECT.						

## Examples

This example creates an XML document that, when serialized, appears as follows :

```
<!DOCTYPE root
[
<!ELEMENT root (child_1, child_2)>
<!ELEMENT child_1 (#PCDATA)*>
<!ELEMENT child_2 (#PCDATA)*>
]>
<root>
  <child_1>text for child.</child_1>
  <child_2>text for child.</child_2>
</root>
```

The definition of the DTD shows that the document is required to have the following composition:

- The document contains a root element with the name root.
- The root element contains a sequence of two child elements named child\_1 and child\_2.
- Both child\_1 and child\_2 contain only text.

The following PowerScript code creates a PBDOM\_TEXT object and assigns it a text value. It then creates a child\_1 element, adds the PBDOM\_TEXT object to it, creates a shallow clone of child\_1, and names the clone child\_2. After adding a clone of the text object to child\_2, the code adds both child objects to the root element:

```
PBDOM_BUILDER      pbdom_buildr
PBDOM_DOCUMENT     pbdom_doc
PBDOM_ELEMENT      pbdom_elem_child_1
PBDOM_ELEMENT      pbdom_elem_child_2
PBDOM_TEXT         pbdom_txt
string strXML = "<!DOCTYPE root [<!ELEMENT root
(child_1, child_2)><!ELEMENT child_1
(#PCDATA)><!ELEMENT child_2 (#PCDATA)>]><root/>"

try
  pbdom_buildr = Create PBDOM_BUILDER
  pbdom_doc = pbdom_buildr.BuildFromString (strXML)

  pbdom_txt = Create PBDOM_TEXT
  pbdom_txt.SetText ("text for child.")

  pbdom_elem_child_1 = Create PBDOM_ELEMENT
  pbdom_elem_child_1.SetName ("child_1")
```



```

pbdom_elem_child_1.AddContent (pbdom_txt)

pbdom_elem_child_2 = pbdom_elem_child_1.Clone(false)
pbdom_elem_child_2.SetName("child_2")
pbdom_elem_child_2.AddContent
(pbdom_txt.Clone(false))

pbdom_doc.GetRootElement().AddContent(pbdom_elem_chi
ld_1)
pbdom_doc.GetRootElement().AddContent(pbdom_elem_chi
ld_2)

pbdom_doc.SaveDocument ("sample.xml")

catch (PBDOM_EXCEPTION pbdom_except)
    MessageBox ("PBDOM_EXCEPTION",
pbdom_except.GetMessage())
end try

```

**Usage**

The Clone method creates a new PBDOM\_TEXT object that is a duplicate of, and a separate object from, the original. Whether true or false is supplied as the parameter to this function, a PBDOM\_TEXT clone is always identical to its original. This is because a PBDOM\_TEXT does not contain any subtree of children PBDOM\_OBJECTs.

A PBDOM\_TEXT clone has no parent. However, the clone resides in the same PBDOM\_DOCUMENT as its original, and if the original PBDOM\_TEXT object is standalone, the clone is standalone

**Detach****Description**

Detaches a PBDOM\_TEXT object from its parent PBDOM\_OBJECT.

**Syntax**

*pbdom\_text\_name*.Detach()

Argument	Description
<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object

**Return value**

PBDOM\_OBJECT. The current PBDOM\_TEXT object is detached from its parent.

**Usage**

If the current PBDOM\_TEXT object has no parent, nothing happens.

## Equals

**Description** Tests for the equality of the current PBDOM\_TEXT object and a referenced PBDOM\_OBJECT.

**Syntax** *pbdom\_text\_name*.Equals(*pbdom\_object pbdom\_object\_ref*)

Argument	Description
<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object
<i>pbdom_object_ref</i>	A reference to a PBDOM_OBJECT to test for equality with the current PBDOM_TEXT object

**Return value** Boolean. Returns true if the current PBDOM\_TEXT object is equivalent to the input PBDOM\_OBJECT, and false otherwise.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not a reference to an object derived from PBDOM\_OBJECT.

**Usage** True is returned only if the referenced PBDOM\_OBJECT is also a derived PBDOM\_TEXT object and refers to the same DOM object as the current PBDOM\_TEXT object. Two separately created PBDOM\_TEXT objects, for example, can contain exactly the same text but not be equal.

## GetObjectClass

**Description** Returns a long integer code that indicates the class of the current PBDOM\_OBJECT.

**Syntax** *pbdom\_object\_name*.GetObjectClass()

Argument	Description
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

**Return value** Long. GetObjectClass returns a long integer code that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_TEXT object, the returned value is 7.

**See also** GetObjectClassString

## GetObjectClassString

**Description** Returns a string form of the class of the PBDOM\_OBJECT.

**Syntax** `pbdom_object_name.GetObjectClassString()`

Argument	Description
<i>pbdom_object_name</i>	The name of a PBDOM_OBJECT

**Return value** String. `GetObjectClassString` returns a string that indicates the class of the current PBDOM\_OBJECT. If *pbdom\_object\_name* is a PBDOM\_TEXT object, the returned string is “pbdom\_text”.

**See also** `GetObjectClass`

## GetOwnerDocumentObject

**Description** Returns the owning PBDOM\_DOCUMENT of the current PBDOM\_TEXT object.

**Syntax** `pbdom_text_name.GetOwnerDocumentObject()`

Argument	Description
<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object

**Return value** PBDOM\_OBJECT.

**Usage** If there is no owning PBDOM\_DOCUMENT, null is returned.

## GetParentObject

**Description** Returns the parent PBDOM\_OBJECT of the current PBDOM\_TEXT object.

**Syntax** `pbdom_text_name.GetParentObject()`

Argument	Description
<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object

**Return value** PBDOM\_OBJECT.

**Usage** The parent is also an object inherited from PBDOM\_TEXT object. If the PBDOM\_TEXT object has no parent, null is returned.

**See also** `SetParentObject`

## GetText

**Description** Obtains the text data that is contained within the current PBDOM\_TEXT object.

**Syntax** `pbdom_text_name.GetText()`

Argument	Description
<code>pbdom_text_name</code>	The name of a PBDOM_TEXT object

**Return value** String. The GetText method returns the textual content of the current PBDOM\_TEXT object.

**Examples** If you have the element `<abc>MY TEXT</abc>`, and you have a PBDOM\_TEXT object to represent the text node “MY TEXT”, then calling GetText on the PBDOM\_TEXT object returns the string “MY TEXT”.

**See also** GetTextNormalize  
GetTextTrim  
SetText

## GetTextNormalize

**Description** Obtains the text data that is contained within the current PBDOM\_TEXT object, with all surrounding whitespace characters removed and internal whitespace characters normalized to a single space.

**Syntax** `pbdom_text_name.GetTextNormalize()`

Argument	Description
<code>pbdom_text_name</code>	The name of a PBDOM_TEXT object

**Return value** String.

**Examples** If you have a PBDOM\_TEXT object that represents the text node “ MY TEXT ”, calling GetTextNormalize returns the string “MY TEXT”. All surrounding whitespaces are removed, and the whitespaces between the words “MY” and “TEXT” are reduced to a single space.

**Usage** This method allows the caller to obtain the text data that is contained within the current PBDOM\_TEXT object with all surrounding whitespaces removed and internal whitespaces normalized to single spaces. If no textual value exists for the current PBDOM\_TEXT object, or if only whitespaces exist, an empty string is returned.

**See also** GetText, GetTextTrim, SetText

## GetTextTrim

**Description** Returns the textual content of the current PBDOM\_TEXT object with all surrounding whitespace characters removed.

**Syntax** `pbdom_text_name.GetTextTrim()`

Argument	Description
<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object

**Return value** String.

**Examples** If you have a PBDOM\_TEXT object that represents the text node “ MY TEXT ”, calling `GetTextNormalize` returns the string “MY TEXT”. All surrounding white spaces are removed. The whitespaces between the words “MY” and “TEXT” are preserved.

**Usage** This method allows the caller to obtain the text data that is contained within the current PBDOM\_TEXT object with all surrounding whitespaces removed. Internal whitespaces are preserved. If no textual value exists for the current PBDOM\_TEXT object, or if only whitespaces exist, an empty string is returned.

**See also** `GetText`  
`GetTextNormalize`  
`SetText`

## SetParentObject

**Description** Sets the referenced PBDOM\_OBJECT to be the parent of the current PBDOM\_TEXT object.

**Syntax** `pbdom_text_name.SetParentObject(pbdom_object pbdom_object_ref)`

Argument	Description
<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object
<i>pbdom_object_ref</i>	A PBDOM_OBJECT to be set as the parent of the current PBDOM_TEXT object

**Return value** PBDOM\_OBJECT.

**Throws** EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_USE – If the input PBDOM\_OBJECT is not referenced to an object derived from PBDOM\_OBJECT.

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_PARENT – If the current PBDOM\_TEXT object already has a parent.

EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT is of a class that does not have a proper parent-child relationship with the PBDOM\_TEXT class.

EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJECT – If the input PBDOM\_OBJECT requires a user-defined name and it has not been named.

Usage

The PBDOM\_OBJECT that you set to be the parent of the current PBDOM\_TEXT object must have a legal parent-child relationship with the current object. If it does not, an exception is thrown. Only a PBDOM\_ELEMENT is allowed to be set as the parent of a PBDOM\_TEXT object.

See also

GetParentObject

## SetText

Description

Sets the input string to be the text content of the current PBDOM\_TEXT object.

Syntax

*pbdom\_text\_name*.SetText(*strSet*)

Argument	Description
<i>pbdom_text_name</i>	The name of a PBDOM_TEXT object
<i>strSet</i>	The string you want set as the text of the PBDOM_TEXT object

Return value

String. If no DTD is referenced, an empty string is returned.

See also

GetText  
 GetTextNormalize  
 GetTextTrim

About this chapter

This chapter provides a quick reference to the methods of PBDOM base classes and additional methods provided by inherited classes.

## Summary of PBDOM classes and methods

### **PBDOM\_OBJECT inherited from PowerBuilder NonVisualObject**

---

`addcontent ( pbdom_object pbdom_object_ref )` returns `pbdom_object`  
`clone ( boolean bdeep )` returns `pbdom_object`  
`detach ( )` returns `pbdom_object`  
`equals ( pbdom_object pbdom_object_ref )` returns `boolean`  
`getcontent ( ref pbdom_object pbdom_object_array[] )` returns `boolean`  
`getname ( )` returns `string`  
`getobjectclass ( )` returns `long`  
`getobjectclassstring ( )` returns `string`  
`getownerdocumentobject ( )` returns `pbdom_document`  
`getparentobject ( )` returns `pbdom_object`  
`gettext ( )` returns `string`  
`gettextnormalize ( )` returns `string`  
`gettexttrim ( )` returns `string`  
`haschildren ( )` returns `boolean`  
`insertcontent ( pbdom_object pbdom_object_new, pbdom_object pbdom_object_ref )` returns `pbdom_object`  
`isancestorobjectof ( pbdom_object pbdom_object_ref )` returns `boolean`  
`removecontent ( pbdom_object pbdom_object_ref )` returns `boolean`  
`setcontent ( pbdom_object pbdom_object_array[] )` returns `pbdom_object`  
`setname ( string strname )` returns `boolean`  
`setparentobject ( pbdom_object pbdom_object_ref )` returns `pbdom_object`

## **PBDOM\_ELEMENT inherited from PBDOM\_OBJECT**

---

`addcontent ( string strtext )` returns `pbdom_element`  
`addnamespacedeclaration ( string strnamespaceprefix, string strnamespaceuri )` returns `pbdom_element`  
`getattribute ( string strname )` returns `pbdom_attribute`  
`getattribute ( string strname, string strnamespaceprefix, string strnamespaceuri )` returns `pbdom_attribute`  
`getattributes ( ref pbdom_attribute pbdom_attribute_array[] )` returns `boolean`  
`getattributevalue ( string strattributename )` returns `string`  
`getattributevalue ( string strattributename, string strdefaultvalue )` returns `string`  
`getattributevalue ( string strattributename, string strnamespaceprefix, string strnamespaceuri )` returns `string`  
`getattributevalue ( string strattributename, string strnamespaceprefix, string strnamespaceuri, string strdefaultvalue )` returns `string`  
`getchildelement ( string strelementname )` returns `pbdom_element`  
`getchildelement ( string strelementname, string strnamespaceprefix, string strnamespaceuri )` returns `pbdom_element`  
`getchildelements ( ref pbdom_element pbdom_element_array[] )` returns `boolean`  
`getchildelements ( string strelementname, ref pbdom_element pbdom_element_array[] )` returns `boolean`  
`getchildelements ( string strelementname, string strnamespaceprefix, string strnamespaceuri, ref pbdom_element pbdom_element_array[] )` returns `boolean`  
`getnamespaceprefix ( )` returns `string`  
`getnamespaceuri ( )` returns `string`  
`getqualifiedname ( )` returns `string`  
`hasattributes ( )` returns `boolean`  
`haschildelements ( )` returns `boolean`  
`isrootelement ( )` returns `boolean`  
`removeattribute ( pbdom_attribute pbdom_attribute_ref )` returns `boolean`  
`removeattribute ( string strattributename )` returns `boolean`  
`removeattribute ( string strattributename, string strnamespaceprefix, string strnamespaceuri )` returns `boolean`  
`removechildelement ( string strelementname )` returns `boolean`  
`removechildelement ( string strelementname, string strnamespaceprefix, string strnamespaceuri )` returns `boolean`  
`removechildelements ( )` returns `boolean`  
`removechildelements ( string strelementname )` returns `boolean`  
`removechildelements ( string strelementname, string strnamespaceprefix, string strnamespaceuri )` returns `boolean`  
`removenamespacedeclaration ( string strnamespaceprefix, string strnamespaceuri )` returns `boolean`  
`setattribute ( pbdom_attribute pbdom_attribute_ref )` returns `pbdom_element`  
`setattribute ( string strname, string strvalue )` returns `pbdom_element`  
`setattribute ( string strname, string strvalue, string strnamespaceprefix, string strnamespaceuri, boolean bverifynamespace )` returns `long`  
`setattributes ( pbdom_attribute pbdom_attribute_array[] )` returns `pbdom_element`  
`setdocument ( pbdom_object pbdom_document_ref )` returns `pbdom_element`  
`setnamespace ( string strnamespaceprefix, string strnamespaceuri, boolean bverifynamespace )` returns `long`  
`settext ( string strtext )` returns `pbdom_element`



**PBDOM\_ATTRIBUTE inherited from PBDOM\_OBJECT**

---

getbooleanvalue ( ) returns boolean  
getdatetimevalue ( string strdateformat, string strtimeformat ) returns datetime  
getdatevalue ( string strdateformat ) returns date  
getdoublevalue ( ) returns double  
getintvalue ( ) returns integer  
getlongvalue ( ) returns long  
getnamespaceprefix ( ) returns string  
getnamespaceuri ( ) returns string  
getownerelementobject ( ) returns pbdom\_element  
getqualifiedname ( ) returns string  
getrealvalue ( ) returns real  
gettimevalue ( string strtimeformat ) returns time  
getuintvalue ( ) returns unsignedinteger  
getulongvalue ( ) returns unsignedlong  
setbooleanvalue ( boolean boolvalue ) returns pbdom\_attribute  
setdatetimevalue ( datetime datetimevalue, string strdateformat, string strtimeformat ) returns pbdom\_attribute  
setdatevalue ( date datevalue, string strdateformat ) returns pbdom\_attribute  
setdoublevalue ( double doublevalue ) returns pbdom\_attribute  
setintvalue ( integer intvalue ) returns pbdom\_attribute  
setlongvalue ( long longvalue ) returns pbdom\_attribute  
setnamespace ( string strnamespaceprefix, string strnamespaceuri, boolean bverifynamespace ) returns long  
setownerelementobject( pbdom\_element pbdom\_element\_ref ) returns pbdom\_attribute  
setrealvalue ( real realvalue ) returns pbdom\_attribute  
settext ( string strtext ) returns pbdom\_attribute  
settimevalue ( time timevalue, string strtimeformat ) returns pbdom\_attribute  
setuintvalue ( unsignedinteger uintvalue ) returns pbdom\_attribute  
setulongvalue ( unsignedlong ulongvalue ) returns pbdom\_attribute

**PBDOM\_CHARACTERDATA inherited from PBDOM\_OBJECT**

---

append ( pbdom\_characterdata pbdom\_characterdata\_ref ) returns pbdom\_characterdata  
append ( string strappend ) returns pbdom\_characterdata  
settext ( string strtext ) returns pbdom\_characterdata

**PBDOM\_COMMENT inherited from PBDOM\_CHARACTERDATA**

---

No added methods.

**PBDOM\_TEXT inherited from PBDOM\_CHARACTERDATA**

---

No added methods.

**PBDOM\_CDATA inherited from PBDOM\_TEXT**

---

No added methods.

### **PBDOM\_DOCTYPE inherited from PBDOM\_OBJECT**

---

getinternalsubset ( ) returns string  
getpublicid ( ) returns string  
getsystemid ( ) returns string  
setdocument ( pbdom\_document pbdom\_document\_ref ) returns pbdom\_doctype  
setinternalsubset ( string strinternalsubset ) returns pbdom\_doctype  
setpublicid ( string strpublicid ) returns pbdom\_doctype  
setsystemid ( string strsystemid ) returns pbdom\_doctype

### **PBDOM\_DOCUMENT inherited from PBDOM\_OBJECT**

---

detachrootelement ( ) returns pbdom\_element  
getdoctype ( ) returns pbdom\_doctype  
getrootelement ( ) returns pbdom\_element  
hasrootelement ( ) returns boolean  
newdocument ( string strootelementname ) returns boolean  
newdocument ( string strootelementnamespaceprefix, string strootelementnamespaceuri,  
string strootelementname, string strdoctypepublicid, string strdoctypesystemid ) returns boolean  
savedocument ( string strfilename ) returns boolean  
setdoctype ( pbdom\_doctype pbdom\_doctype\_ref ) returns pbdom\_document  
setrootelement ( pbdom\_element pbdom\_element\_ref ) returns pbdom\_document

### **PBDOM\_ENTITYREFERENCE inherited from PBDOM\_OBJECT**

---

No added methods.

### **PBDOM\_PROCESSINGINSTRUCTION inherited from PBDOM\_OBJECT**

---

getdata ( ) returns string  
getnames ( ref string name\_array[] ) returns boolean  
gettarget ( ) returns string  
getvalue ( string strname ) returns string  
removevalue ( string strname ) returns boolean  
setdata ( string strdata ) returns pbdom\_processinginstruction  
setvalue ( string strname, string strvalue ) returns pbdom\_processinginstruction

### **PBDOM\_BUILDER inherited from PowerBuilder NonVisualObject**

---

buildfromdatastore ( datastore datastore\_ref ) returns pbdom\_document  
buildfromfile ( string strurl ) returns pbdom\_document  
buildfromstring ( string strxmlstream ) returns pbdom\_document  
getparseerrors( ref string strErrorMessageArray[]) returns boolean

### **PBDOM\_EXCEPTION inherited from PowerBuilder Exception**

---

getexceptioncode ( ) returns long

# Index

## A

AddContent method (PBDOM\_ATTRIBUTE) 65  
AddContent method (PBDOM\_DOCUMENT) 200  
AddContent method (PBDOM\_ELEMENT) 224  
AddContent method (PBDOM\_OBJECT) 296  
AddNamespaceDeclaration method  
    (PBDOM\_ELEMENT) 228  
AddToBypassList method (SoapConnection) 32  
Append method (PBDOM\_CDATA) 125  
Append method (PBDOM\_COMMENT) 176, 177  
Append method (PBDOM\_TEXT) 142, 143, 146,  
    336

## B

Begin method (EJBTransaction) 10  
BuildFromDataStore method (PBDOM\_BUILDER)  
    116  
BuildFromFile method (PBDOM\_BUILDER) 117  
BuildFromString method (PBDOM\_BUILDER) 120

## C

classes, EJB  
    EJBConnection 5  
    EJBTransaction 10  
    JavaVM 16  
classes, PBDOM  
    overview 61  
    PBDOM\_ATTRIBUTE 63  
    PBDOM\_BUILDER 115  
    PBDOM\_CDATA 123  
    PBDOM\_CHARACTERDATA 141  
    PBDOM\_COMMENT 175  
    PBDOM\_DOCTYPE 187  
    PBDOM\_DOCUMENT 199  
    PBDOM\_ELEMENT 223

PBDOM\_ENTITYREFERENCE 133  
PBDOM\_EXCEPTION 293  
PBDOM\_OBJECT 295  
PBDOM\_PROCESSINGINSTRUCTION 321  
PBDOM\_TEXT 335  
    quick reference 345  
classes, SOAP  
    SoapConnection 31  
    SoapException 44  
    SoapPBCookie 46  
classes, UDDIProxy 54  
Clone method (PBDOM\_ATTRIBUTE) 65  
Clone method (PBDOM\_CDATA) 125, 134  
Clone method (PBDOM\_CHARACTERDATA) 147  
Clone method (PBDOM\_COMMENT) 178  
Clone method (PBDOM\_DOCTYPE) 188  
Clone method (PBDOM\_DOCUMENT) 203  
Clone method (PBDOM\_ELEMENT) 230  
Clone method (PBDOM\_OBJECT) 296  
Clone method  
    (PBDOM\_PROCESSINGINSTRUCTION)  
        322  
Clone method (PBDOM\_TEXT) 337  
Commit method (EJBTransaction) 11  
ConnectToServer method (EJBConnection) 6  
conventions xiv  
CreateInstance method (SoapConnection) 32  
CreateJavaInstance method (EJBConnection) 7  
CreateJavaInstance method (JavaVM) 19  
CreateJavaVM method (JavaVM) 16

## D

Detach method (PBDOM\_ATTRIBUTE) 67  
Detach method (PBDOM\_CDATA) 127  
Detach method (PBDOM\_CHARACTERDATA) 149  
Detach method (PBDOM\_COMMENT) 180  
Detach method (PBDOM\_DOCTYPE) 188  
Detach method (PBDOM\_ELEMENT) 231

## Index

Detach method (PBDOM\_ENTITYREFERENCE) 136  
Detach method (PBDOM\_OBJECT) 297  
Detach method (PBDOM\_PROCESSINGINSTRUCTION)  
323  
Detach method (PBDOM\_TEXT) 339  
DisconnectServer method (EJBConnection) 8  
DynamicCast (SoapConnection) 34  
DynamicCast method (JavaVM) 20

## E

EJB classes  
EJBConnection 5  
EJBTransaction 10  
JavaVM 16  
EJBConnection class 5  
EJBTransaction class 10  
Equals method (PBDOM\_ATTRIBUTE) 69  
Equals method (PBDOM\_CDATA) 128  
Equals method (PBDOM\_CHARACTERDATA) 151  
Equals method (PBDOM\_COMMENT) 181  
Equals method (PBDOM\_DOCTYPE) 189  
Equals method (PBDOM\_ELEMENT) 231  
Equals method (PBDOM\_ENTITYREFERENCE) 136  
Equals method (PBDOM\_OBJECT) 298  
Equals method (PBDOM\_PROCESSINGINSTRUCTION)  
323  
Equals method (PBDOM\_TEXT) 340  
EXCEPTION\_DATA\_CONVERSION 291  
EXCEPTION\_HIERARCHY\_ERROR 292  
EXCEPTION\_ILLEGAL\_PBOBJECT 290  
EXCEPTION\_INAPPROPRIATE\_USE\_OF\_PBDOM\_OB  
JECT 289  
EXCEPTION\_INTERNAL\_XML\_ENGINE\_ERROR  
291  
EXCEPTION\_INVALID\_ARGUMENT 290  
EXCEPTION\_INVALID\_NAME 291  
EXCEPTION\_INVALID\_OPERATION 292  
EXCEPTION\_INVALID\_STRING 292  
EXCEPTION\_MEMORY\_ALLOCATION\_FAILURE  
291  
EXCEPTION\_MULTIPLE\_DOCTYPE 290  
EXCEPTION\_MULTIPLE\_ROOT\_ELEMENT 288  
EXCEPTION\_MULTIPLE\_XMLDECL 292  
EXCEPTION\_PBDOM\_NOT\_INITIALIZED 293

EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_  
OWNER 292  
EXCEPTION\_PBDOM\_OBJECT\_ALREADY\_HAS\_  
PARENT 289  
EXCEPTION\_PBDOM\_OBJECT\_INVALID\_FOR\_U  
SE 289  
EXCEPTION\_USE\_OF\_UNNAMED\_PBDOM\_OBJE  
CT 288  
EXCEPTION\_WRONG\_DOCUMENT\_ERROR 288  
EXCEPTION\_WRONG\_PARENT\_ERROR 290  
exceptions, PBDOM 287  
extensions  
about 1  
nonvisual, using 3  
third-party, finding 1  
visual, using 3

## F

findBusiness method (UDDIProxy) 55  
findService method (UDDIProxy) 57

## G

GetActualClass method (JavaVM) 24  
GetAttribute method (PBDOM\_ELEMENT) 233  
GetAttributes method (PBDOM\_ELEMENT) 235  
GetAttributeValue method (PBDOM\_ELEMENT)  
235  
GetBooleanValue method (PBDOM\_ATTRIBUTE)  
70  
getBusinessDetail method (UDDIProxy) 56  
GetChildElement method (PBDOM\_ELEMENT) 239  
GetChildElements method (PBDOM\_ELEMENT)  
241  
GetComment method (SoapPBCookie) 46  
GetCommentURI method (SoapPBCookie) 47  
GetContent method (PBDOM\_ATTRIBUTE) 72  
GetContent method (PBDOM\_DOCUMENT) 204  
GetContent method (PBDOM\_OBJECT) 298  
GetData method  
(PBDOM\_PROCESSINGINSTRUCTION)  
324

- GetDateTimeValue method (PBDOM\_ATTRIBUTE) 73
- GetDateValue method (PBDOM\_ATTRIBUTE) 72
- GetDocType method (PBDOM\_DOCUMENT) 206
- GetDocument method (PBDOM\_ATTRIBUTE) 78
- GetDocument method (PBDOM\_CHARACTERDATA) 152
- GetDocument method (PBDOM\_COMMENT) 182
- GetDocument method (PBDOM\_DOCTYPE) 191
- GetDocument method (PBDOM\_ELEMENT) 249
- GetDocument method (PBDOM\_ENTITYREFERENCE) 138
- GetDocument method (PBDOM\_OBJECT) 302
- GetDocument method (PBDOM\_PROCESSINGINSTRUCTION) 326
- GetDocument method (PBDOM\_TEXT) 341
- GetDoubleValue method (PBDOM\_ATTRIBUTE) 74
- GetEJBTransaction method (EJBConnection) 8
- GetElementsByTagName method (PBDOM\_DOCUMENT) 206
- GetExceptionCode method (PBDOM\_EXCEPTION) 293
- GetExpired method (SoapPBCookie) 47
- GetExpires method (SoapPBCookie) 47
- GetHttpOnly method (SoapPBCookie) 48
- GetInterfaces method (JavaVM) 25
- GetInternalSubset method (PBDOM\_DOCTYPE) 189
- GetIntValue method (PBDOM\_ATTRIBUTE) 74
- GetJavaClasspath method (JavaVM) 26
- GetJavaVMVersion method (JavaVM) 27
- GetLongValue method (PBDOM\_ATTRIBUTE) 74
- GetMessage method (SoapException) 44
- GetName method (PBDOM\_ATTRIBUTE) 75
- GetName method (PBDOM\_CHARACTERDATA) 155
- GetName method (PBDOM\_DOCTYPE) 189
- GetName method (PBDOM\_ELEMENT) 246
- GetName method (PBDOM\_ENTITYREFERENCE) 137
- GetName method (PBDOM\_OBJECT) 299
- GetName method (PBDOM\_PROCESSINGINSTRUCTION) 324
- GetName method (SoapPBCookie) 48
- GetNames method (PBDOM\_PROCESSINGINSTRUCTION) 325
- GetNamespacePrefix method (PBDOM\_ATTRIBUTE) 76
- GetNamespacePrefix method (PBDOM\_ELEMENT) 247
- GetNamespaceUri method (PBDOM\_ATTRIBUTE) 76
- GetNamespaceUri method (PBDOM\_ELEMENT) 247
- GetObjectClass method (PBDOM\_ATTRIBUTE) 77
- GetObjectClass method (PBDOM\_CDATA) 128
- GetObjectClass method (PBDOM\_CHARACTERDATA) 156
- GetObjectClass method (PBDOM\_COMMENT) 181
- GetObjectClass method (PBDOM\_DOCTYPE) 190
- GetObjectClass method (PBDOM\_DOCUMENT) 207
- GetObjectClass method (PBDOM\_ELEMENT) 248
- GetObjectClass method (PBDOM\_ENTITYREFERENCE) 137
- GetObjectClass method (PBDOM\_OBJECT) 300
- GetObjectClass method (PBDOM\_PROCESSINGINSTRUCTION) 325
- GetObjectClass method (PBDOM\_TEXT) 340
- GetObjectClassString method (PBDOM\_ATTRIBUTE) 77
- GetObjectClassString method (PBDOM\_CDATA) 129
- GetObjectClassString method (PBDOM\_CHARACTERDATA) 156
- GetObjectClassString method (PBDOM\_COMMENT) 182
- GetObjectClassString method (PBDOM\_DOCTYPE) 190
- GetObjectClassString method (PBDOM\_DOCUMENT) 208
- GetObjectClassString method (PBDOM\_ELEMENT) 249
- GetObjectClassString method (PBDOM\_ENTITYREFERENCE) 137
- GetObjectClassString method (PBDOM\_OBJECT) 301

## Index

- GetObjectClassString method  
(PBDOM\_PROCESSINGINSTRUCTION) 326
  - GetObjectClassString method (PBDOM\_TEXT) 341
  - GetOwnerDocumentObject method (PBDOM\_CDATA)  
129
  - GetOwnerElementObject method (PBDOM\_ATTRIBUTE)  
80
  - GetParent method (PBDOM\_ATTRIBUTE) 63
  - GetParentObject method (PBDOM\_CDATA) 129
  - GetParentObject method (PBDOM\_CHARACTERDATA)  
158
  - GetParentObject method (PBDOM\_COMMENT) 182
  - GetParentObject method (PBDOM\_DOCTYPE) 191
  - GetParentObject method (PBDOM\_ELEMENT) 250
  - GetParentObject method  
(PBDOM\_ENTITYREFERENCE) 138
  - GetParentObject method (PBDOM\_OBJECT) 303
  - GetParentObject method  
(PBDOM\_PROCESSINGINSTRUCTION) 326
  - GetParentObject method (PBDOM\_TEXT) 341
  - GetParseErrors method (PBDOM\_BUILDER) 121
  - GetPublicID method (PBDOM\_DOCTYPE) 191
  - GetQualifiedName method (PBDOM\_ATTRIBUTE) 81
  - GetQualifiedName method (PBDOM\_ELEMENT) 251
  - GetRealValue method (PBDOM\_ATTRIBUTE) 82
  - GetSecure method (SoapPBCookie) 48
  - GetStatus method (EJBTransaction) 12
  - GetSuperClass method (JavaVM) 27
  - GetSystemID method (PBDOM\_DOCTYPE) 192
  - GetTarget method  
(PBDOM\_PROCESSINGINSTRUCTION) 327
  - GetText method (PBDOM\_ATTRIBUTE) 82
  - GetText method (PBDOM\_CDATA) 130
  - GetText method (PBDOM\_CHARACTERDATA) 161
  - GetText method (PBDOM\_COMMENT) 183
  - GetText method (PBDOM\_ELEMENT) 251
  - GetText method (PBDOM\_OBJECT) 304
  - GetText method  
(PBDOM\_PROCESSINGINSTRUCTION) 327
  - GetText method (PBDOM\_TEXT) 342
  - GetTextNormalize method (PBDOM\_ATTRIBUTE) 84
  - GetTextNormalize method (PBDOM\_CDATA) 130
  - GetTextNormalize method  
(PBDOM\_CHARACTERDATA) 162
  - GetTextNormalize method (PBDOM\_COMMENT) 183
  - GetTextNormalize method (PBDOM\_ELEMENT)  
252
  - GetTextNormalize method (PBDOM\_OBJECT) 306
  - GetTextNormalize method  
(PBDOM\_PROCESSINGINSTRUCTION)  
328
  - GetTextTrim method (PBDOM\_ATTRIBUTE) 86
  - GetTextTrim method (PBDOM\_CDATA) 131
  - GetTextTrim method (PBDOM\_CHARACTERDATA)  
166
  - GetTextTrim method (PBDOM\_COMMENT) 184
  - GetTextTrim method (PBDOM\_ELEMENT) 252
  - GetTextTrim method (PBDOM\_OBJECT) 308
  - GetTextTrim method  
(PBDOM\_PROCESSINGINSTRUCTION)  
328
  - GetTextTrim method (PBDOM\_TEXT) 343
  - GetTimeStamp method (SoapPBCookie) 49
  - GetTimeValue method (PBDOM\_ATTRIBUTE) 88
  - GetUIntValue method (PBDOM\_ATTRIBUTE) 88
  - GetUlongValue method (PBDOM\_ATTRIBUTE) 89
  - GetURI method (SoapPBCookie) 49
  - GetValue method  
(PBDOM\_PROCESSINGINSTRUCTION)  
329
  - GetValue method (SoapPBCookie) 49
  - GetVersion method (SoapPBCookie) 49
- ## H
- HasAttributes method (PBDOM\_ELEMENT) 253
  - HasChildElements method (PBDOM\_ELEMENT)  
254
  - HasChildren method (PBDOM\_ATTRIBUTE) 89
  - HasChildren method (PBDOM\_CHARACTERDATA)  
170
  - HasChildren method (PBDOM\_DOCUMENT) 209
  - HasChildren method (PBDOM\_ELEMENT) 255
  - HasChildren method (PBDOM\_OBJECT) 311
  - HasRootElement method (PBDOM\_DOCUMENT)  
209

- I**
- InsertContent method (PBDOM\_ATTRIBUTE) 91
  - InsertContent method (PBDOM\_DOCUMENT) 209
  - InsertContent method (PBDOM\_ELEMENT) 256
  - InsertContent method (PBDOM\_OBJECT) 312
  - IsAncestorObjectOf method (PBDOM\_ATTRIBUTE) 93
  - IsAncestorOf method (PBDOM\_CHARACTERDATA) 171
  - IsAncestorOf method (PBDOM\_DOCUMENT) 212
  - IsAncestorOf method (PBDOM\_ELEMENT) 258
  - IsAncestorOf method (PBDOM\_OBJECT) 313
  - IsJavaVMLoaded method (JavaVM) 28, 29
  - IsRootElement method (PBDOM\_ELEMENT) 258
- J**
- JavaVM class 16
- L**
- Lookup method (EJBConnection) 9
- N**
- NewDocument method (PBDOM\_DOCUMENT) 212
  - nonvisual extensions, using 3
- P**
- PBDOM classes
    - overview 61
    - quick reference 345
  - PBDOM exceptions 287
  - PBDOM\_ATTRIBUTE class 63
  - PBDOM\_BUILDER class 115
  - PBDOM\_CDATA class 123
  - PBDOM\_CHARACTERDATA class 141
  - PBDOM\_COMMENT class 175
  - PBDOM\_DOCTYPE class 187
  - PBDOM\_DOCUMENT class 199
  - PBDOM\_ELEMENT class 223
  - PBDOM\_ENTITYREFERENCE class 133
  - PBDOM\_EXCEPTION class 293
  - PBDOM\_OBJECT class 295
  - PBDOM\_PROCESSINGINSTRUCTION class 321
  - PBDOM\_TEXT class 335
  - PBNI, not supported in .NET targets 2
  - PowerBuilder extensions
    - about 1
    - not supported in .NET targets 2
    - using 2
- R**
- RemoveAttribute method (PBDOM\_ELEMENT) 259
  - RemoveAuthentication (SoapConnection) 35
  - RemoveBypassList (SoapConnection) 35
  - RemoveChildElement method (PBDOM\_ELEMENT) 261
  - RemoveChildElements method (PBDOM\_ELEMENT) 263
  - RemoveContent method (PBDOM\_ATTRIBUTE) 93
  - RemoveContent method (PBDOM\_DOCUMENT) 216
  - RemoveContent method (PBDOM\_ELEMENT) 265
  - RemoveContent method (PBDOM\_OBJECT) 315
  - RemoveNamespaceDeclaration method (PBDOM\_ELEMENT) 266
  - RemoveValue method (PBDOM\_PROCESSINGINSTRUCTION) 329
  - Rollback method (EJBTransaction) 13
- S**
- SaveDocument method (PBDOM\_DOCUMENT) 217
  - SaveDocumentIntoString method (PBDOM\_DOCUMENT) 218
  - SetAttribute method (PBDOM\_ELEMENT) 267
  - SetAttributes method (PBDOM\_ELEMENT) 276
  - SetBasicAuthentication (SoapConnection) 36

## Index

- SetBooleanValue method (PBDOM\_ATTRIBUTE) 96
- SetBypassProxyOnLocal (SoapConnection) 37
- SetCertificateFile (SoapConnection) 37
- SetComment method (SoapPBCookie) 50
- SetCommentUri method (SoapPBCookie) 50
- SetContent method (PBDOM\_ATTRIBUTE) 96
- SetContent method (PBDOM\_DOCUMENT) 219
- SetContent method (PBDOM\_ELEMENT) 279
- SetContent method (PBDOM\_OBJECT) 316
- SetData method
  - (PBDOM\_PROCESSINGINSTRUCTION) 330
- SetDateTimeValue method (PBDOM\_ATTRIBUTE) 101
- SetDateValue method (PBDOM\_ATTRIBUTE) 100
- SetDocType method (PBDOM\_DOCUMENT) 220
- SetDocument method (PBDOM\_DOCTYPE) 193
- SetDocument method (PBDOM\_ELEMENT) 282
- SetDoubleValue method (PBDOM\_ATTRIBUTE) 102
- SetExpired method (SoapPBCookie) 50
- SetExpires method (SoapPBCookie) 51
- SetHttpOnly method (SoapPBCookie) 51
- setInquiryUrl method (UDDIProxy) 54
- SetInternalSubset method (PBDOM\_DOCTYPE) 193
- SetIntValue method (PBDOM\_ATTRIBUTE) 102
- SetLongValue method (PBDOM\_ATTRIBUTE) 102
- SetMessage method (SoapException) 44
- SetName method (PBDOM\_ATTRIBUTE) 103
- SetName method (PBDOM\_DOCTYPE) 194
- SetName method (PBDOM\_ELEMENT) 282
- SetName method (PBDOM\_ENTITYREFERENCE) 139
- SetName method (PBDOM\_OBJECT) 317
- SetName method
  - (PBDOM\_PROCESSINGINSTRUCTION) 331
- SetName method (SoapPBCookie) 52
- SetNamespace method (PBDOM\_ATTRIBUTE) 106
- SetNamespace method (PBDOM\_ELEMENT) 283
- setOption method (UDDIProxy) 55
- SetOptions method (SoapConnection) 38
- SetOwnerElementObject method (PBDOM\_ATTRIBUTE)  
109
- SetParent method (PBDOM\_ATTRIBUTE) 63
- SetParentObject method (PBDOM\_CDATA) 131
- SetParentObject method (PBDOM\_CHARACTERDATA)  
171
- SetParentObject method (PBDOM\_COMMENT) 184
- SetParentObject method (PBDOM\_DOCTYPE) 195
- SetParentObject method (PBDOM\_ELEMENT) 284
- SetParentObject method
  - (PBDOM\_ENTITYREFERENCE) 139
- SetParentObject method (PBDOM\_OBJECT) 318
- SetParentObject method
  - (PBDOM\_PROCESSINGINSTRUCTION)  
332
- SetParentObject method (PBDOM\_TEXT) 343
- SetProxyServer method (SoapConnection) 40
- SetProxyServerOptions method (SoapConnection) 41
- SetRealValue method (PBDOM\_ATTRIBUTE) 111
- SetRollbackOnly method (EJBTransaction) 14
- SetRootElement method (PBDOM\_DOCUMENT)  
221
- SetSecure method (SoapPBCookie) 52
- SetSoapLogFile method (SoapConnection) 42
- SetSystemID method (PBDOM\_DOCTYPE) 197
- SetText method (PBDOM\_ATTRIBUTE) 111
- SetText method (PBDOM\_CDATA) 132
- SetText method (PBDOM\_CHARACTERDATA)  
174
- SetText method (PBDOM\_COMMENT) 185
- SetText method (PBDOM\_ELEMENT) 285
- SetText method (PBDOM\_TEXT) 344
- SetTimeout method (SoapConnection) 42
- SetTimeValue method (PBDOM\_ATTRIBUTE) 112
- SetTransactionTimeout method (EJBTransaction) 15
- SetUIntValue method (PBDOM\_ATTRIBUTE) 113
- SetUlongValue method (PBDOM\_ATTRIBUTE) 113
- SetUri method (SoapPBCookie) 53
- SetUseDefaultProxySetting method (SoapConnection)  
32, 35, 37, 43
- SetValue method
  - (PBDOM\_PROCESSINGINSTRUCTION)  
333
- SetValue method (SoapPBCookie) 53
- SetVersion method (SoapPBCookie) 53
- SoapConnection class 31
- SoapException class 44
- SoapPBCookie class 46

## T

typographical conventions xiv



## **U**

- UDDIProxy class 54
- UseConnectionCache (SoapConnection) 43
- UseIntegratedWindowsAuthentication  
    (SoapConnection) 44

## **V**

- visual extensions, using 3

