XMetaL® 16 Developer

Programmer's Guide

About JustSystems

JustSystems is a leading global software provider with three decades of successful innovation in office productivity, information management, and consumer and enterprise software. With over 2,500 customers worldwide and annual revenues over $110M, the company is continuing a global expansion strategy that includes its enterprise software offering called xfy, its XMetaL content lifecycle solutions, and its pioneering work in the definition of the XBRL standard and commercialization of enabling technologies. A Gartner “Cool Vendor” selection in 2008, JustSystems is also a member of KMWorld’s 100 Companies that Matter in Knowledge Management for 2008 and the 2007 EContent 100. XMetaL is a 2008 KMWorld Trend-Setting Product. Major strategic partnerships include IBM, Oracle and EMC. For more information, please visit http://www.justsystems.com.

Contact Information:

Support:
North America: +1 866 647 2003

Sales:
North America: +1 866 793 1542

Office Locations:

XMetaL Sales & Support
Suite 3220
666 Burrard Street
Box 207
Vancouver, BC, Canada
V6C 2X8
T: 604-602-9928
Toll-Free Sales: 1-866-793-1542

Tokushima Head Office
Brains Park Kawauchi-cho
Tokushima-city Tokushima 771-0189
Japan
T: 088 666 1000
(+81 88 666 1000 from outside Japan)

Contents

Introduction

This guide describes XMetaL® Author and XMetaL® XMAX™ programming features.

Send your comments or questions about XMetaL documentation to *mailto:docs-feedback@xmetal.com*.

References

You can find more information about Windows Scripting and Microsoft XML Core Services (MSXML) from the Microsoft Developer's Network at *http://msdn.com*.

Specifications for XPath, XSLT, and the DOM are published by the World Wide Web Consortium. See *http://www.w3c.org* for details.

Scripting interface

The scripting interface provides access to XMetaL via an object model that is based on the Document Object Model (DOM) and the Microsoft® Office Word Visual Basic® for Applications (VBA) model.

You can run scripts in the following ways:

- From a macro. For most users, this is the most common way of running scripts.
- When you insert an element. You can specify a script to be executed when a particular element is inserted.

Script language

You can use any scripting language that conforms to the Microsoft Scripting Language Interface. Built-in support is provided for JScript and VBScript; this manual provides documentation and examples for these two languages. You must install script engines if you use other languages.

You must specify the language ID in macros and in the Customizations editor. This is the ProgID of the ActiveX control that implements the script engine, and can be found in the registry.

Scripts that are dropped or pasted must begin with the appropriate language comment string.

Note: The drop point is the cursor location at the time XMetaL lost focus. This means that regardless of where a script was dropped into the XMetaL Author editing window, the macro is executed upon the point where the cursor last was.

For VBScript the language comment string is:

```
' XMetaL Script Language VBScript:
```

For JScript it is:

```
// XMetaL Script Language JScript:
```

For an arbitrary language (for which a script engine must be installed) the string optionally starts with a delimiter (the best choice is the comment characters used by the language), followed by the following string, as indicated (but with `<progid>` replaced by the correct language ID):

```
XMetaL Script Language <progid>:
```

For example:

```
# XMetaL Script Language PERLSCRIPT:
```

```
# XMetaL Script Language Python:
```

The total length of the language comment string must be less than 128 characters.

If you include the language comment string in a macro (this is not required), the starting delimiter *must* be the language's comment delimiter.

For languages other than JScript and VBScript, you can check if you have the appropriate script engine installed using a simple script. For example:

```
# XMetaL Script Language PERLSCRIPT:$Selection->TypeText("Hello from Perl");
```

Copy the script, put the insertion point in the XMetaL document window that text is allowed, and then paste the script. If only the string in quotes is pasted, the scripting engine is working.

Basic scripting concepts

This manual refers to abstract data objects as *interfaces*. Certain *properties* and *methods* are supported. An instance of an interface is referred to in this manual as an *object* or, in the case of DOM objects that inherit the DOMNode interface, a *node*.

Both properties and methods can modify their objects. The basic distinction is that setting a property should not affect the value of any other property, whereas methods cause the object to perform some action that could change a property.

The XMetaL implementation contains the following interfaces:

- General interfaces. These are general purpose interfaces for document editing. They implement many functions available through the XMetaL interface.
- DOM interfaces, which implement the interfaces defined in the DOM Level 1 and DOM Level 2 specifications. These interfaces represent an XML document using a node structure and provide properties and methods for navigating and manipulating that structure.
- Command bar interfaces, for manipulating XMetaL menus and toolbars.
- Customization interfaces, for manipulating XACs.
- The top-level XMetaL XMAX interface, which implements the properties, methods, and events of the XMetaL XMAX control.

Global objects

Every script has access to globally defined objects.

Global Object	Available in XMetaL Author?	Available in XMetaL XMAX?
Application	Yes	No
ActiveDocument	Yes	Yes
Documents	Yes	No
ResourceManager	Yes	No
Selection	Yes	Yes

The Application object represents the currently running XMetaL instance.

In XMetaL Author, the ActiveDocument object is a Document object that represents the active document (the document in the active window). If ActiveDocument is assigned to a variable, the value of the variable changes if a different document becomes the active document. Each instance of XMetaL XMAX has its own ActiveDocument object. If you have two instances of XMetaL XMAX, (for example, `Control1` and `Control2`),

then both instances have their own ActiveDocument objects, and these two objects are different. The value of an ActiveDocument object does not change.

The Documents object is a collection of Document objects. It represents all of the currently open XMetaL documents.

The Selection object represents the user's current selection. This object can be accessed as `Application.Selection` in XMetaL Author, or `Document.Selection` in XMetaL XMAX, or `Selection` in both XMetaL Author and XMetaL XMAX.

By contrast, a Range object represents a selection that does not have to be the current selection. In XMetaL Author, the selection represented by a Range object can be in the same document as the selection, or in a different document. A Range object provides all the properties and methods of the Selection interface. Methods that modify the document structure (for example, inserting an element) should normally be used with Range objects rather than Selection objects, since Range objects can be placed in a greater number of locations in the document.

Macro format

XMetaL macro files are XML files. You can edit these files with the XMetaL script editor or with a text editor.

The contents of the file are enclosed in `<MACROS>` tags. Individual macros are enclosed in `<MACRO>` tags.

Many macros also have the `key` attribute, which specifies the macro's shortcut key. The `lang` attribute corresponds to the `ProgID` of the ActiveX control that implements the scripting language. The language comment string is optional.

```
<MACROS>
<MACRO name="Sample Macro" key="Ctrl+Alt+M" lang="JScript">
// XMetaL Script Language JSCRIPT:
ActiveDocument.Host.Alert("A typical macro.");
<MACRO>
...
</MACROS>
```

If you associate a macro with a toolbar button or menu item, other attribute values are generated automatically.

Complex customizations can contain hundreds utilities and tens of thousands line of code. XMetaL *.mcr files are written in xml format. Optional first children of "<MACRO>" tag are "<SCRIPT/>" elements.

<SCRIPT/>" elements allow users to specify the list of script files that XMetaL embeds into the macro during XMetaL start up initialization. The XMetaL Developer solution provides an efficient way of maintaining, native editing, navigation and script debugging in Visual Studio. It includes breakpoints, variable watching, XMetaL API intellisense support, etc.

<SCRIPT/>" elements contain information about script files' location, language and title (usually it is the name of the file that appears in Visual Studio during debugging). The "src" attribute contains the relative path to a referenced script file. The '$SQDIR' variable can be used at the beginning. XMetaL will resolve it as path to the …`\Author` folder.


```
<!DOCTYPE MACROS SYSTEM "macros.dtd">
<MACROS>
  <MACRO name="On_Macro_File_Load" lang="JScript" hide="true">
    <SCRIPT src="$SQDIR\Test\ExtensionsData.js" lang="JScript" name="ExtensionsData.js"></SCRIPT>
    <SCRIPT src="$SQDIR\Test\WhiteBoardService.js" lang="JScript" name="WhiteBoardService.js"></SCRIPT>
    <SCRIPT src="$SQDIR\Test\WhiteBoardImpl.js" lang="JScript" name="WhiteBoardImpl.js"></SCRIPT>
    <SCRIPT src="$SQDIR\Test\WhiteBoardDebug.js" lang="JScript" name="WhiteBoardDebug.js"></SCRIPT>
    <![CDATA[
]]>
  </MACRO>
  <MACRO name="Add to favorites..." lang="JScript" hide="false" desc="ABC">
    <![CDATA[
//  OpenAddToFavoritesDlg(...) defined in WhiteBoardImpl.js
  OpenAddToFavoritesDlg(1);
]]>
  </MACRO>
   <MACRO name="Test macro" lang="JScript"  hide= "true">
    <SCRIPT src="$SQDIR\XMDK\WhiteBoard\js\test.js" lang="JScript" name="test.js"></SCRIPT>
    <![CDATA[
//  doTest(...) defined in test.js
  doTest(1);
]]>
  </MACRO>
</MACROS>
```

Protecting markup

If your macro contains XML markup, (or strings with the same format as markup) you need to protect it with a CDATA section.

```
<MACRO name="Safe Macro" key="Ctrl+Alt+S" lang="JScript">
// XMetaL Script Language JSCRIPT:

<[CDATA[
Selection.PasteString("<Index/>");
ActiveDocument.Host.Alert("Press <Enter> to close this dialog box.");
]]>

<MACRO>
```

Hiding macros (XMetaL Author only)

You can prevent a macro from appearing in the **Macros** dialog box, **Macros** toolbar, or the **Toolbar** and **Menu Customization** dialog box by setting the optional hide attribute to True. This is useful for special event-driven macros, which are not intended to be run directly by users.

```
<MACRO name="On_Update_UI" lang="JScript" hide="true">
// ...code... <MACRO>
```

Sample macros

See the files `XMetaL\Author\Macros\journalist.mcr` and `XMetaL\Author\Macros\xmetal.mcr` for macro examples.

Omitting arguments

Some arguments can be omitted because they have been designated as optional. You may also want to supply an empty or null value for a required argument.

JScript and VBScript have different rules for omitting arguments.

JScript

Only trailing optional arguments can be omitted. For example, the `Selection.InsertImage` method has six arguments, the last five of which are optional:

```
InsertImage(strSrc, [strAlt], [strAlign], [strWidth], [strHeight], [strBorder])
```

If you want to supply a value for `strWidth` and `strHeight`, you cannot omit `strAlt` and `strAlign` (though you can omit `strBorder`, as a trailing argument). You can specify a null string (`""`) if you do not want to supply a value, for example:

```
Selection.InsertImage("donkey.gif","","","200","150");
```

Sometimes the semantics of a command are such that an undesirable result occurs if you specify a null string for an optional argument. For example:

```
Application.NoticeBox("Hello","Yes","","","Hi");
```

This results in a blank button on the dialog box. In this situation, the expression `null` can sometimes be used to represent the omitted value:

```
Application.NoticeBox("Hello","Yes",null,null,"Hi");
```

You cannot use null strings for optional non-string arguments. If such an argument is not a trailing argument, then a real value must be supplied, even though the argument is optional. For example, `Selection.HomeKey` has two optional boolean arguments:

```
HomeKey([intLocation=0], [intSelectionType=0])
```

The following call is *incorrect*, because a null string is used for `intLocation`:

```
Selection.HomeKey("",1);
```

VBScript

You can omit any optional argument as long as the comma that would normally follow it is still present, for example:

```
ans = Application.NoticeBox("Invalid paste.", "Continue?",,,"Paste text")
```

Trailing arguments can be omitted completely.

Null required arguments

Required arguments can never be omitted, but sometimes you can supply a null value, in both JScript and VBScript. There is no general rule governing this practice, and it should be approached with caution. Most of the time, if an argument is required, the semantics of the property or method require that a ‘real’ value be supplied.

Variable scope in XMetaL Author

If a variable is defined at the ‘top-level’ of a macro (that is, not within a subroutine or a function) then its value persists and is available to other macros that are written in the same scripting language until the next time the variable is set or XMetaL closes. This applies also to functions defined at the top level.

You can define a set of global variables and functions that are available when editing any document. You can define these functions at any time, but the On_Macro_File_Load macro and the On_Application_Open macro are particularly good places to do this.

You can also use the NameVariantProperties interface to store globally available objects (these can be strings, numbers, or anything else that can be stored in a variable). These objects are available to all macros, regardless of scripting language.

Variable persistence provides some programming flexibility but there are some things you should consider:

- Follow good programming practices. For example, if a macro expects a variable to have a zero or null value when it is invoked, make sure that another value does not persist from a previous invocation of this or another macro. The most reliable way to do this is to define the macro inside a function (JScript) or subroutine (VBScript). Here is an example:

```
<MACRO name="Macro1" lang="VBScript">
Sub Do_Macro1()
' ...code... End Sub
Call Do_Macro1
</MACRO>
```

- If you are depending on macros being called in a certain order, program accordingly.
- Memory usage problems could arise if many variables persist after the macro terminates.
- Run time errors can occur if a variable that has gone out of scope is referenced. This is particularly true of variables that represent objects (rather than strings or numbers). For example, if the document represented by `doc` in the following code has been closed, a run-time error occurs:

```
<MACRO name="abc" key="" lang="VBScript">
'XMetaL Script Language VBScript:
MsgBox(doc.IsValid) ' doc set up by some other macro
</MACRO>
```

- Run-time errors can occur if a variable is declared twice, in successive executions of a macro. For example, if you declare an array in a VBScript macro and run the macro twice, you get a type mismatch error.

```
<MACRO name="DimTest" lang="VBScript">
Dim A(10)
' ...more code... </MACRO>
```

  This macro should be written as a subroutine so that the array is not declared at the top level, and therefore does not persist between macro executions.

```
<MACRO name="DimTest" lang="VBScript">
Sub Do_DimTest()
Dim A(10)
' ...more code... End Sub
Call Do_DimTest
</MACRO>
```

Constants

The enumerated constants used by the scripting engine can be used in XMetaL macros.

These are constants that are used as arguments or return values for various properties and methods. These constants are documented with the property or method that uses them. Enumerated constants and their

numerical equivalents can be used interchangeably: your script may be more readable if you specify a constant by its name rather than by its value.

Additional scripting capabilities

Before working on scripts for XMetaL, you should familiarize yourself with the scripting capabilities of the scripting language(s) that you use. You should also be familiar with the Microsoft Scripting Run-time Library, the Windows Scripting Host and the Microsoft XML parser.

The Microsoft Scripting Run-time Library is a set of objects that extend the capabilities of JScript and VBScript. For example, for security reasons, these languages do not support reading and writing of files. This capability can be obtained from FileSystemObject, which supports properties and methods for file system access. Here is a sample that shows how to copy a file using the CopyFile method:

```
var fso = new ActiveXObject("Scripting.FileSystemObject");
fso.CopyFile(paramFile,newParamFile);
```

The Windows Scripting Host enables you to run Windows programs from a macro. The following example shows how to start a telnet session from XMetaL:

```
<MACRO name="Telnet" lang="JScript">
var WSHShell = new ActiveXObject("WScript.Shell");
WSHShell.Run("telnet");
</MACRO>
```

The Microsoft XML parser (MSXML) lets you incorporate XSLT processing into XMetaL macros. The On_Before_Document_Preview macro in the sample macros file (`Author\Startup\multipleOutput.mcr`) has an example of using MSXML in a macro.

DLL error handling

If you call a DLL from a macro, and the DLL is not registered properly, a run-time error occurs. You should set up error-handling procedures to avoid this.

Here is an example of an error-handling procedure:

```
' XMetaL Script Language VBSCRIPT:
On Error Resume Next
dim dlogSet
set dlogSet = CreateObject("YourCustom.DlogSet")
if Err.Number <> 0 Then
ActiveDocument.Host.Alert ("Cannot create YourCustom Object")
Else
dlogSet.showElementDlog
End If
set dlogSet = Nothing
```

XPath support

The XPath standard is partially supported by XMetaL.

Use the following APIs:

- DOMDocument.getNodesByXPath

- DOMElement.getNodesByXPath

Note: These APIs do not support pattern matching.

Scripting tips

The following tips provide helpful information for programmers.

It is good practice to check at the beginning of the macro which document view (Plain Text, Normal, Tags On) the document is displayed in. Many macros do not work the same in all views. Macros that use containers or DOM nodes do not work in Plain Text view. You can test the view type using `Document.ViewType`.

JScript

A backslash in literal text must be represented as a double backslash, "\\". For example:

```
Application.FileExists("c:\\data\\food.xml");
```

Method names must be followed by parentheses, even if they have no arguments. For example:

```
Selection.DeleteRow();
```

VBScript

Properties that return a value should be used only in code that accesses the returned value (in an assignment statement or conditional expression).

If a method does not return a value, parentheses around a non-null argument list are optional. For such methods, the 'Usage' sections in this manual omit the parentheses. For example, both calls below are correct:

```
Selection.TypeText "Hello"
```

```
Selection.TypeText("Hello")
```

If a method returns a value, parentheses around non-null argument lists can be omitted if the code does not use the return value (in an assignment statement or conditional expression), but are required if the code does use the return value. For such methods, the 'Usage' sections in this manual include the parentheses. For example, the `Selection.MoveToElement` method is defined as follows:

```
MoveToElement([strElementName], [boolForward=true])
```

If the return value is not used, both of these calls are correct:

```
Selection.MoveToElement("Para")
```

```
Selection.MoveToElement "Para"
```

The following call is *incorrect*, because the return value is used in an assignment:

```
ans = Selection.MoveToElement "Para"
```

The *correct* form is:

```
ans = Selection.MoveToElement("Para")
```

DOM interfaces

The Document Object Model (DOM) is an abstract definition of an API for manipulating XML document structures. XMetaL supports the DOM Level 1 specification and partially supports the DOM Level 2 specification.

The DOM was designed specifically to represent XML structures, but can represent SGML structures if they are also found in XML.

XMetaL implements the DOM using COM interfaces that enable you to access document structures using any scripting language that conforms to the Microsoft Scripting Language Interface. If you are programming for XMetaL XMAX, you can access document structures using any programming language that supports the ActiveX control. You only need to choose a scripting language that conforms to the Microsoft Scripting Language Interface when writing macros.

The DOM refers to abstract data objects as *interfaces*. This is because the DOM does not specify an underlying implementation (such as a class) for these data objects. The DOM does not consist of any particular language or implementation. Rather, it describes document structures in terms of the tools used to manipulate the structure: *attributes* (not to be confused with element attributes in the XML/SGML/XHTML sense), which we call *properties*, and *methods*. A concrete instance of an interface is referred to as an *object*.

The DOM interfaces represent a document as a logical node or tree structure (that is, users can think of the structure as node based, but the implementation details are the choice of the vendor and are considered to be unimportant to the end-user).

The distinction between properties and methods is sometimes blurred, since both types of operations can modify their objects. The basic distinction is that setting a property should not affect the value of any other property, whereas methods cause the object to perform some action which may or may not change one of its properties.

DOMNode interface

The basic DOM interface is the DOMNode interface. All DOM interfaces inherit this interface, with three exceptions: DOMNodeList, which is an ordered collection of DOMNodes; DOMNamedNodeMap, an unordered collection of DOMNodes; and DOMImplementation, a 'meta-object' that provides information about the specific DOM implementation being used.

You can think of each interface as a type of DOMNode, which may be qualified in either or both of the following ways:

- The interface may be augmented with some extra properties and methods that the generic DOMNode does not have. For example, DOMCharacterData inherits the DOMNode interface, and has two properties and five methods of its own.
- On the other hand, some of the methods and properties that an interface inherits from DOMNode may be syntactically valid but semantically not useful. For example, since DOMComment objects cannot have child nodes, any DOMNode properties that are used to navigate child nodes always return null if applied to such objects.

Each DOMNode has a nodeType property, which identifies what kind of node it is (DOMElement, DOMText, etc.).

Working with selections and nodes

You can work with a document structure via the Selection interface or by navigating and modifying the DOM representation of the document, in which document components are represented as nodes in a tree structure.

Sometimes you may need to communicate between the two representations of the document. If you are working with a selection (or range) and need to obtain the node (DOMNode) that contains the selection, you can use the `Selection.ContainerNode` property: this returns the DOMNode corresponding to the selection's container. Conversely, if you are working with a node and need to obtain a selection inside the element represented by that node, you can use the `Selection.SelectNodeContents` method, which selects all of the contents of the node. You can also select before or after the node using `Selection.SelectAfterNode` and `Selection.SelectBeforeNode`.

Note: A node can be the child of only one parent node at a time; therefore, if a node is already the child of some node, it is removed from that node's tree before being inserted in the tree of another node. Furthermore, nodes can be used only in the document that they were created in.

Rules checking

By default, rules checking stays on during DOM operations that modify the document. Therefore operations that would create an invalid document are disallowed. If you want to relax this restriction, you can do so via the `Document.DOMEnforceValid` property.

Document structure

A typical document may be represented using the DOM hierarchy of nodes.

The top level node of a document is a Document (DOMDocument) object. This node is the parent or 'root' node for a tree structure that represents the document. A Document object can have child nodes of the following types:

- DOMElement
- DOMProcessingInstruction
- DOMComment
- DOMDocumentType

For this discussion, we will consider in detail only the DOMElement node. The other nodes can contain important information, but none of them have child nodes of their own, and generally most of a document's structure is contained in its elements.

A Document object can have only one child node of type DOMElement. This node corresponds to the top-level element in the document (of which there is only one instance), and it is the starting point for any further navigations and manipulations of the document.

A DOMElement node can have the following child nodes, in any number and order:

- DOMElement
- DOMText
- DOMComment
- DOMProcessingInstruction
- DOMCDATASection
- DOMEntityReference

These objects correspond to the XML and SGML constructs that you typically find inside an element: sub-elements, text, comments, processing instructions, sections, and entity references.

Since DOMElement inherits the properties and methods of DOMNode, you can make use of DOMNode's properties for navigating in the node structure: childNodes, firstChild, lastChild, nextSibling, parentNode, and previousSibling. Using these operations, you can traverse the entire document.

Even though attributes correspond to a DOM interface (DOMAttr), they do not appear as child nodes of DOMElement. Rather, a DOMAttr is accessed via the getAttributeNode (not namespace aware) or getAttributeNodeNS (namespace aware) method of DOMElement, which returns a DOMAttr object. DOMElement can also access an attribute value as a string via the getAttribute method or the getAttributeNS method.

Namespaces

XMetaL supports XML namespaces. If your XML document uses namespaces, you will need to use namespace-aware APIs to perform document processing. These APIs should be used with XML Schema-based documents.

If your XML document uses namespaces, you will need to use namespace aware APIs to perform document processing.

For DTD-based documents, you should normally use APIs that are not namespace aware. If you using an element or attribute that does not belong to a namespace as a parameter to a namespace-aware API, use the empty string (" ") as the namespace URI parameter.

Note: Do not mix APIs that are namespace-aware with APIs that are not namespace aware. Unpredictable results may occur.

Command Bar interfaces

Microsoft Word and related Microsoft Office products offer significant and extensive information and examples regarding the use of their Command Bar interfaces. XMetaL emulates the Command Bar API that is provided by Microsoft Office.

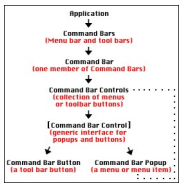
Most customizations of the XMetaL menu bar and toolbars can be carried out through the XMetaL graphical interface, rather than using the properties and methods. Choose **View** > **Toolbars** to carry out such customizations.

The On_Default_CommandBars_Complete event macro is a good place to carry out Command Bar customizations.

You can group menu commands and toolbar buttons by adding separators using the BeginGroup property.

Note: Separators cannot occur as the first item in a menu or toolbar.

The following diagram describes the relationships between the interfaces:



This code example further illustrates the relationships:

```
// XMetaL Script Language JSCRIPT:
// Get the CommandBars object
var cmdBars = Application.CommandBars;
// Get a single command bar (the main menu bar)
var menuBar = cmdBars.item("Menu bar");
// Get all of the menu bar's controls (all of the menus)
var menuBarCtrls = menuBar.Controls;
// Get a single control (the File menu)
// (a CommandBarPopup object)
var fileMenu = menuBarCtrls.item("File");
// Now get all of the controls on the File Menu
// (a CommandBarControls object)
var fileMenuCtrls = fileMenu.Controls;
// Add a new menu item (a CommandBarPopup object)
var newItem = fileMenuCtrls.Add(sqcbcTypePopup);
// Give the new menu item a label
newItem.Caption = "Choose Me!";
// Associate a macro with the new menu item
newItem.OnAction = "Macro1";
```

Top-level XMetaL XMAX interface

The top-level XMetaL XMAX interface represents the properties, methods, and events that are accessible directly from an XMetaL XMAX object.

IXMetaLControl is the interface that implements XMetaL XMAX properties and methods. _IXMetaLControlEvents is the top-level dispatch interface for XMetaL XMAX events.

APIs for the top-level XMetaL XMAX interface are classified in the following categories:

- XMetaL XMAX Document
- XMetaL XMAX Windowing
- XMetaL XMAX User Interface

Forms

To display a form in XMetaL Author, use the following code.

```
// XMetaL Script Language JSCRIPT:
// Call an instance of myform.xft
var dlg=Application.CreateFormDlg("C:\\myform.xft");
dlg.DoModal();
```

Content mapping model

You can use XFT-specific script events, properties, and methods to transfer data between an XML document and a form.

Only DOM node types of NODE_TEXT are mapped to the underlying XML document. If the DOM node associated with an XFT form control contains other node types, such as NODE_PROCESSING_INSTRUCTION or NODE_COMMENT, those types are deleted when XFT maps the form data back into the XML document.

XFT concatenates multiple NODE_TEXT items. For example, the following:

```
<A>Some text with<?xm-replace_text {Type Here}?> a PI</A>
```

Changes to:

```
<A>Some text with a PI</A>
```

Note: Due to the behavior of the Windows system and its own internal APIs, events may fire more than once. Therefore, you should not use display alerts, change document selection, or move focus in an way when in the context of an XFT-specific script event.

Table 1: XFT-specific script events

Name	Description
OnXftPutFormToDoc	Use this script event to create or delete child DOMNodes before XFT transfers the data from the form to the underlying XML document. XFT does not create or delete DOMNodes. The underlying content model of the form's anchor element may be so complex (containing, for example, OR groups, and required elements) that the form is unable to determine the correct content model. Therefore, maintaining the correct content model is the responsibility of the developer.
OnXftPutDocToForm	Use this script event to finish setting up the form. This event is called after the XFT system has transferred the XML document data to the form controls. An example use of this script is to apply rules for enabling or disabling the form controls based on the data in the controls.
OnXftGetXPath	Use this script event to modify a control's XPath property value. This script is run before the XFT system retrieves the underlying XML document's DOMNode that corresponds to the control's XPath property value. The XPath property value must be a relative path from the root DOMNode of the form. If the XPath property value is empty, the XFT

Name	Description
	system does not copy any data from the XML document to that particular control.
OnXftPutXmlValue	Use this script event to modify an XmlValue property value retrieved from the DOMNode. This script runs before XFT transfers the XmlValue property value into the form control. If there is no OnXftPutXmlValue script defined, XFT copies the DOMNode value to the form control XmlValue property value, and then uses the XmlValue as the control's current value. For static, text, edit box and multi-edit box controls, the current value is the text currently in the control. For combo box and list box controls, the current value is the currently selected item in the control. All other controls require this script if they are bound to XML content. An example use of this script is to handle the movement of data from an ActiveX control.
OnXftGetXmlValue(XmlValue)	Use this script to modify the XmlValue property value of a form. This script runs before XFT transfers the XmlValue property value from the control to the underlying DOMNode. If there is no OnXftGetXmlValue script defined, XFT copies the current value in the control to the XmlValue property value and then directly to the DOMNode value. For static, text, edit box, and multi-edit box controls, the current value is the text currently in the control. For combo box and list box controls, the current value is the currently selected item in the control. All other form controls require this script if they are bound to XML content. An example use of this script is to handle the movement of data to an ActiveX control.

Table 2: Control properties

Name	Description
XMLNodeRules	This property is reserved for future use.
XmlValue	This property is used by the XFT system as a placeholder for transferring XML data into and out of the control. XMLValue cannot contain markup.
XPath	This property is used by the XFT system to determine the underlying XML document's DOMNode to which the control's XmlValue is bound.

Table 3: Form property

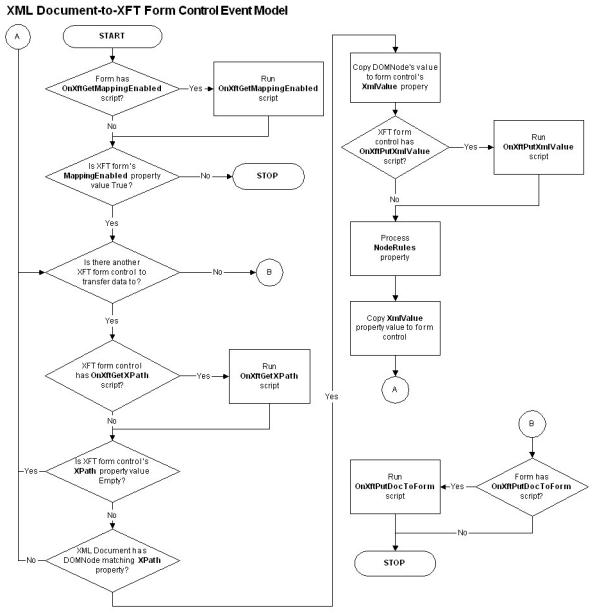
Name	Description
MappingEnabled	This property is a boolean value. When it is True, data is moved in and out of the form. When it is False, no data transfer to and from the XML document occurs. This property is usually set with the OnXftGetMappingEnabled script event.

Table 4: User form methods

Name	Description
GetNodeByXPath(XPath,node)	where node is parent XMetaL::DOMNode, from which XPath works; where XPath is string that represents relative to node path. Only an explicit path (without any search operations as *,?,|... etc.) is supported; where XPathNode is XMetaL::DOMNode that has valid value if XPath is found, or null if not.
CheckAllNodesExist (node);	where node is parent XMetaL::DOMNode from which to check if all nodes to be mapped to XML Document from the form exist. where bExist is true if nodes exist, false if nodes do not exist.
ReadFromXMLDoc();	This method forces XMetaL to update the form with data from the XML document.
SaveToXMLDoc();	This method forces XMetaL to update the XML document with data from the form.
SetAutoMap(node);	where node is parent XMetaL::DOMNode from which to map data to the XML document. This method forces XMetal to update the document and form automatically if any data has changed.

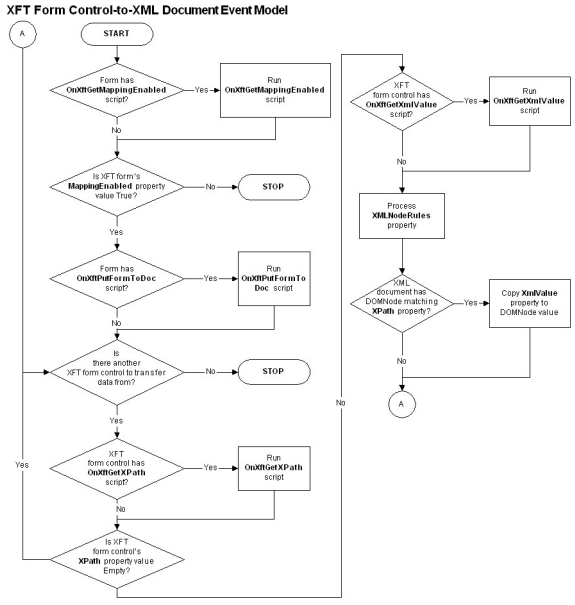
XML document to form control event model

The following diagram shows the programmatic flow followed by the underlying XFT systems when populating an XFT form with data from an XML document.



Form control to XML document event model

The following diagram shows the programmatic flow followed by the underlying XFT systems when moving data into an XML document from an XFT form.



Tips for working with forms

You can improve the performance and usability of your customization by carefully considering your approach to developing forms.

Mapping operations

One method to improve the performance of XFT forms is to enable mapping operations only when they are needed. For example, the following code controls the enabling or disabling of the mapping operations:

```
function TheView_OnXftGetMappingEnabled() {
TheView.MappingEnabled = getMappingEnabled();
try
   {
   // Hide frame border if user clicks on somewhere other than the form
   // only hide it if the form is read-only
   if (gDisabledAll
   && Application.Selection.ContainerNode != gSpecNode)
      {
      FrameBorder.Visible = false;
      }
   }
   catch (e)
   {
   }
}
```

Mapping operations do not take place when MappingEnabled is False. The getMappingEnabled function depends on some global variables to set the state of MappingEnabled.

An event handler for the OnChange event is used in many controls to provide information about whether the control has changed. You should look at the implementation of the OnChange event handler for different sample controls to see how they interact with the global variables on the form.

Creating new XML nodes

Normally you would use the XPath property of a control to transfer data from the form to an XML document. However, the XPath property has a limitation: the element you want to bind to the XFT form must exist in the document before data can be transferred using the XPath property. One solution to this problem is to use the event handler for the OnXftPutFormToDoc event to create the associated nodes. This is demonstrated in the following code:

```
function TheView_OnXftPutFormToDoc(domNode) {
try
   {
   // setting globals
   gSpecNode = domNode;
   gDisplayFeaturesNode = findChildNode(gSpecNode, "display.features");
   gMemoryFeaturesNode = findChildNode(gSpecNode, "memory.features");
   gCallFeaturesNode = findChildNode(gSpecNode, "call.features");
   // transfer data from form to doc
   modeFormToDoc();
   memoryFeaturesFormToDoc();
   displayFeaturesFormToDoc();
   callFeaturesFormToDoc();
   resetDirty();
   }
catch (e)
   {
   }
}
```

Note that this code depends on a number of global functions.

Setting the read-only property

You can disable a form whenever the associated XML element is read-only. You can use the Range interface to determine if an element is read-only.

Changing the save document prompts

By default, XMetaL prompts users to save frequently when the focus is in a form, regardless of whether the controls have been changed. Frequent prompts can impair the usability of a form.

Adjust the MaxLineLength and Text Layout properties

Every control that can display text on an XFT form (except a multi-edit box) shows a Carriage Return/Line Feed (CRLF, or ASCII 0A 0D) as a double ‘box’ character. You cannot put CRLF characters into controls directly (except for a multi-edit box); however, XMetaL Author or XMetaL XMAX can insert them. If the MaxLineLength customization property is set to be smaller than the line length (the default is 80 characters) the XML source for an element will contain CRLF characters. These characters are transferred to the control from XMetaL, and from the control back to XMetaL when the form is closed. To avoid this problem, increase the value for MaxLineLength in the customization file. You may also need to use the Text Layout properties to force the element onto its own line.

Programming for XMetaL XMAX

XMetaL XMAX is a system-registered ActiveX control. You can think of XMetaL XMAX as a container for XML documents that provides editing functionality such as inserting, deleting, cutting/pasting text. You must customize XMetaL XMAX if you want to perform more complex editing operations such as inserting elements.

XMetaL XMAX must be embedded in an application before use. This application is frequently called a *container application* in this manual. You can create a container application in the following ways:

- Use an Integrated Development Environment (IDE) such as Visual Studio .NET to add XMetaL XMAX to an application. You must work in a programming language that supports the ActiveX control.
- Add XMetaL XMAX to a Web page intended for use with Internet Explorer.

Choosing a programming language

The language that you use for the container application for XMetaL XMAX will likely be different from the scripting language you use to create macros.

The container application must be capable of embedding an ActiveX control, for example, Visual Basic. The only time you are likely to use a scripting language for the container application is when you are embedding XMetaL XMAX in an Internet Explorer Web page.

Using constants

Using constants helps to keep code readable and maintainable. A number of enumerated constants are provided in a type library for use in XMetaL XMAX.

In Visual Studio .NET, you can click on the XMetaLControlLib object to expand it, exposing all properties, methods, interfaces, and enumerated types associated with the object.

Enumerated constants are marked with the icon. If your container application for XMetaL XMAX is a custom application, you can use these constants automatically in your container application.

However, if your container application for XMetaL XMAX is Internet Explorer, you need to define constants in a script file. For example:

```
// A JSCRIPT example file containing enumerated constants
// SQDocViewType constants
var sqViewUnspecified = -1
var sqViewNormal = 0
var sqViewTagsOn = 1

// SQAlertLevel constants
var sqAlertsAll = -1
var sqAlertsNone = 0

// ... other constants ...
```

You can then load your script file in an HTML page.

Use constants in an HTML page

1. Click **View** > **Other Windows** > **Object Browser**.
2. Click **Customize...**.
3. Click **External Components and Libraries**, then click **Add...**.
4. Click the COM tab. Click XMetaL XMAX Type Library, then click **Select**.
5. Click **OK**.
6. Click **OK**.

Variable scope in XMetaL XMAX

If a variable is defined at the 'top-level' of a macro (that is, not within a subroutine or a function), then its value persists and is available to other macros in the same instance of XMetaL XMAX that are written in the same scripting language until the next time the variable is set or the document closes. This also applies to functions defined at the top level.

You can define a set of global variables and functions that are available when editing any document. You can define these functions at any time, but the On_Macro_File_Load macro and the On_Application_Open macro are particularly good places to do this.

You can also use the NameVariantProperties interface to store globally available objects (these can be strings, numbers, or anything else that can be stored in a variable). These objects are available to all macros, regardless of scripting language.

Note: Global variables defined using the above techniques can be accessed only by the instance of XMetaL XMAX that defined them. If you have an application with two instances of XMetaL XMAX (for example, `Control1` and `Control2`), the first cannot access the global variables of the second using the above techniques. If you need to define global variables that are shared among all instances of XMetaL XMAX, define these variables in the container application.

Variable persistence provides some programming flexibility but there are some things you should consider:

- Follow good programming practices. For example, if a macro expects a variable to have a zero or null value when it is invoked, make sure that another value does not persist from a previous invocation of this or another macro. The most reliable way to do this is to define the macro inside a function (JScript) or subroutine (VBScript). Here is an example:

```
<MACRO name="Macro1" lang="VBScript">
Sub Do_Macro1()
' ...code... End Sub
Call Do_Macro1
</MACRO>
```

- If you are depending on macros being called in a certain order, program accordingly.
- Memory usage problems could arise if many variables persist after the macro terminates.

- Run time errors can occur if a variable that has gone out of scope is referenced. This is particularly true of variables that represent objects (rather than strings or numbers). For example, if the document represented by doc in the following code has been closed, a run-time error occurs:

```
<MACRO name="abc" key="" lang="VBScript">
'XMetaL Script Language VBScript:
MsgBox(doc.IsValid) ' doc set up by some other macro
</MACRO>
```

- Run-time errors can occur if a variable is declared twice, in successive executions of a macro. For example, if you declare an array in a VBScript macro and run the macro twice, you get a type mismatch error.

```
<MACRO name="DimTest" lang="VBScript">
Dim A(10)
' ...more code... </MACRO>
```

This macro should be written as a subroutine so that the array is not declared at the top level, and therefore does not persist between macro executions.

```
<MACRO name="DimTest" lang="VBScript">
Sub Do_DimTest()
Dim A(10)
' ...more code... End Sub
Call Do_DimTest
</MACRO>
```

Event handling

XMetaL XMAX fires events to its container application in a number of situations. These events allow the container application to perform custom processing of events. For example, you would write a handler for the OnMessage event if you want to customize the appearance of messages.

If the event parameters are objects, you can modify writable event parameters directly. However, integer, boolean, and string parameters cannot be modified directly. Some special data types are used only for parameters to events: IntegerRefVar, BoolRefVar, and StringRefVar. These interfaces are wrapper interfaces for writable primitive (or immutable) event parameters.

The IntegerRefVar, BoolRefVar, and StringRefVar interfaces have one read/write property: value. When accessing a writable event parameter, use this property to get and set the value of the parameter. For example:

```
// JSCRIPT example
function myXMControl::myEvent(myIntegerRefVarParameter) {
var myInteger = myIntegerRefVarParameter.value;
myIntegerRefVarParameter.value = 38;
}
```

Note: If the boolean, integer, or string event parameter is read-only, treat it as a regular boolean, integer, or string.

Error handling

You should handle the OnMessage event to provide you with error information and ensure that error messages are displayed to users.

Note that many XMetaL XMAX APIs fail silently. This is particularly true of LoadFromFile, LoadFromString and LoadFromStringAsSGML. To prevent run-time errors in your program, you should always check that an API returns a non-null object before attempting to reference the return value.

Commonly used properties, methods, and events

When developing a container application, you must provide menu items, keyboard shortcuts, or toolbars required to perform any tasks not directly available in XMetaL XMAX.

The following is a short list of properties, methods, and events to get started writing a container application. Note that each property or method call should be invoked by a user interface event, such as a button click.

- **Open a document:** Use LoadFromFile to open an existing XML or SGML document.
- **Create an XML document:** UseLoadFromString to create a new XML document.
- **Create an SGML document:** Use LoadFromStringAsSGML to create a new SGML document.
- **Save a document:** Use `Document.Save` to save a document opened with LoadFromFile. Use `Document.SaveAs` to save a document created with LoadFromString, or to save a document opened with LoadFromFile in a different location or under a different name.
- **Close a document:** Use `Document.Close` to close a document.
- **Insert an element:** Use `Selection.CanInsertList` to get a list of elements that can be inserted into the current selection. There are many ways to insert an element, including `Selection.InsertElement`, `Selection.InsertElementNS`, `Document.createElement`, `Document.createElementNS`, `DOMNode.appendChild`, and `DOMNode.insertBefore`.
- **Change an element:** Use `Selection.CanChangeList` to get a list of elements that you can change the current selection into. Use `Selection.ContainerName` or `Selection.ContainerNameNS` to change the element.
- **Validate a document or selection:** Use `Document.Validate` to validate your document. Use `Selection.Validate` to validate your selection. Retrieve validation errors (if any) with `Document.ValidationErrorList` and `Selection.ValidationErrorList`.
- **Run a macro:** Use `Document.Host.Run` or `Document.Host.RunKeyedMacro` to run a macro.
- **Track revisions:** Use `Document.TrackRevisions` to turn revision tracking on or off. You can modify the insertion color, insertion style, deletion color, and deletion style. While revision tracking is turned on, you can accept or reject tracked changes and go to the next or previous change.
- **Create acontext menu:** Write a handler for the OnContextMenu event to create a context menu.

Working with XMetaL XMAX in Visual Studio .NET

You can use Visual Studio .NET to create a custom application that contains XMetaL XMAX.

Add XMetaL XMAX to the toolbox

You must add XMetaL XMAX to the Visual Studio .NET toolbox before you can add XMetaL XMAX to a form or HTML page.

1. Click **View** > **Toolbox**.
2. Right-click in the toolbox, and click **Customize Toolbox**.
3. Click the **COM Components** tab of the Customize Toolbox dialog box.

4. Enable the XMetaL XMAX check box, and click **OK**.

Add XMetaL XMAX to a form

Once you have added XMetaL XMAX to the Visual Studio .NET Toolbox, you can add the control to any form.

1. Create a new project, or open an existing project.
2. Open a new or existing form.
3. Drag the control from the toolbox to the form.

Add XMetaL XMAX to an HTML page

Once you have added XMetaL XMAX to the Visual Studio .NET Toolbox, you can add the control to any HTML page.

1. Create a new project, or open an existing project.
2. Open a new or existing HTML page.
3. Drag the control from the toolbox to the HTML page.

Modify the properties of XMetaL XMAX

Once you have added XMetaL XMAX to a form or HTML page, you can modify its properties.

There are a number of XMetaL XMAX-specific properties that can be set at design time, such as ChangeColorByAuthor, DeletionColor, DeletionStyle, InsertionColor, InsertionStyle, LicenseID, and MiniContextVisible. To view the Properties window, right-click on the XMetaL XMAX control in your form.

Note: If you are usingXMetaL XMAX Concurrent User License Server to grant licenses to end-users of your XMetaL XMAX container application, you must set the LicenseID property at design time. If you are not using XMetaL XMAX Concurrent User License Server, you must leave the LicenseID property blank.

View a sample Visual Basic container application

A sample Visual Basic container application for XMetaL XMAX is provided when XMetaL Developer is installed. You can view the sample Visual Basic application through the XMetaL Developer program group.

Developing an HTML-based container application

XMetaL XMAX can be embedded in an HTML page intended for display in Internet Explorer. Creating a HTML-based container application for XMetaL XMAX is probably the simplest way to create an application for remote authors to use.

Configuring Internet Explorer

XMetaL XMAX operates only in domains or HTML pages specified in the Trusted Sites or Local Intranet security zones.

You can add sites to a security zone using Internet Explorer.

If you are using the default security templates for the Trusted Sites or Local Intranet security zones, it is not necessary to modify them. However, if you are using a custom security template for these zones, you need to ensure that the following security settings are enabled:

- ActiveX controls and plug-ins:

- Download signed ActiveX control (required to download XMetaL XMAX as a CAB file)
- Run ActiveX controls and plug-ins
- Script ActiveX controls marked safe for scripting

- Scripting:
  - Active scripting
  - Run ActiveX controls and plug-ins
  - Script ActiveX controls marked safe for scripting

Other Internet Explorer settings may affect the operation of XMetaL XMAX, particularly the settings for offline web browsing and caching.

OBJECT tag

The CLASSID attribute must have the exact value shown. The CLASSID attribute is used by the system to access XMetaL XMAX. The ID attribute represents the name to use when referring to a specific instance of XMetaL XMAX in a Web page.

The following code must appear in the source code of the page containing XMetaL XMAX:

```
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myIDcode"
HEIGHT="500"
WIDTH="500">
</OBJECT>
```

In this manual, the ID attribute is frequently assumed to have a value of ‘myXMControl’. The HEIGHT and WIDTH attributes represent the height and width of XMetaL XMAX. You can enter an integer value, which represents the number of pixels, or a percentage, which represents the percentage of the screen to use to render XMetaL XMAX.

Download the XMetaL XMAX CAB file

End-users can install XMetaL XMAX from a CD or download and run the XMetaL XMAX CAB file.

Note: Do not download and execute the XMetaL XMAX CAB file on a machine with XMetaL XMAX installed from the CD. This causes problems with the uninstallation process.

To download and run the XMetaL XMAX CAB file from a HTML page, the following code must appear in the HTML source code:

```
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myIDcode"
CODEBASE="http://www.mydomainname.com/XMControl.cab#VERSION=1,0,3,xxx">
</OBJECT>
```

The CLASSID attribute must have the value shown. The CLASSID attribute is used by the system to access XMetaL XMAX. The ID attribute represents the name to use when referring to a specific instance of XMetaL XMAX in a Web page. In this manual, the ID attribute is frequently assumed to have a value of ‘myXMControl’. The CODEBASE attribute is used to download and run the XMetaL XMAX CAB file.

Note: : The ‘xxx’ portion of the version number (above) should be replaced with the actual version number. Specifying a version number equal to or smaller than the version of the currently installed XMAX CAB file results in no action taken. If you specify a number that is greater than the currently-installed CAB file, a newer version is sought and installed. You can use this mechanism to ensure that clients

viewing your HTML page are always up to date with the latest version of the XMAX CAB file (provided you update the page with the new version number). If you erroneously specify a number higher than the latest version of XMAX, the CAB file is repeatedly installed with each page-view.

Before deploying this HTML code to end users, you should:

- Ensure you have copied the XMetaL XMAX CAB file to a Web server or network location. This Web server or network location must be accessible to users of the HTML application.
- Check that the CODEBASE attribute refers to the correct network location and the correct version of the XMetaL XMAX CAB file.

To check the version number of the XMetaL XMAX CAB file:

- Open the XMetaL XMAX CAB file in a program capable of opening and extracting compressed files, such as WinZip.
- Extract `XMControl.dll` to a convenient location.
- Right click on the `XMControl.dll` file and select Properties from the pop-up menu.
- Click the Version tab.
- Select Product Version from the list of items in the Item name list box located in the Other version information area of the Version page.
- Note the Value of the Product Version item (displayed on the right hand side of the Other version information area of the Version page).

For an HTML container application that is not using XMetaL XMAX Concurrent User License Server: If the HTML container application references an XMetaL XMAX CAB file, end users must supply a registration number when the container application calls any of the following to open a file for the first time.

- LoadFromFile
- LoadFromString
- LoadFromStringAsSGML
- Xml

If the user does not supply a valid registration number, XMetaL XMAX does not work.

Configure your HTML application to use XMetaL XMAX Concurrent User License Server

If you are using XMetaL XMAX Concurrent User License Server, you must set the LicenseID parameter in the OBJECT element before attempting to use XMetaL XMAX.

If you do not set the LicenseID to the correct location of XMetaL XMAX Concurrent User License Server, XMetaL XMAX uses a locally installed licensed copy of XMetaL XMAX. During the use of a concurrent license, any loss of connection results in re-connection to the XMetaL XMAX Concurrent User License Server. No attempt is made to get a local license.

You can specify the license server in one of the following ways: by specifying the host name of the license server, or by specifying the IP address of the license server.The following examples illustrate the use of the LicenseID parameter:

```
<!-- example using the host name of the license server -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="myXMControl" HEIGHT="500"
 WIDTH="500">
<PARAM NAME="LicenseID" VALUE="myLicenseServerName">
</OBJECT>
```

```
<!-- example using the IP address of license server -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="myXMControl" HEIGHT="500"
```



```
 WIDTH="500">
<PARAM NAME="LicenseID" VALUE="127.0.0.1">
</OBJECT>
```

Load an XMetaL XMAX document with a XAC in an HTML page

You must use the LoadFromFile, LoadFromString and LoadFromStringAsSGML methods to load an XMetaL XMAX document in a HTML page. Place the API call inside a `SCRIPT` tag.

You cannot use the `<PARAM>` element inside the `<OBJECT>` element to load an XMetaL XMAX document with a XAC in an HTML page.

Using document customizations

A document customization is a collection of files that help to define the structure, display style, and other attributes of a XML document.

A document customization can include any of the following files:

- Compiled rules file
- DTD or schema
- Style sheets
- Macro files
- Customization files
- Document templates

Document customizations can bestored in a file with a XAC extension. Document customizations can also be provided as a collection of files in a folder. Refer to the *XMetaL Customization Guide* for more information about creating, modifying, and deploying customization files.

Choosing a deployment strategy

You can deploy a document customizations using a XAC file or a collection of customization files.

Here are some of the advantages of using a XAC to deploy a customization:

- Allows for remote distribution of customizations
- Completely encapsulates a collection of customization files in a single file

Loading a document customization

XMetaL XMAX applies search rules in a specific order when attempting to load a document customization.

If a customization is found, the search is stopped, and the document customization is returned. The following search rules apply in order:

1. The XAC associated with the XML or SGML document's document type declaration system identifier is loaded from XMetaL Developer. This happens if XMetaL Developer is installed, running, and a customization is being run in Debug mode (that is, if you push F5 to run a customization in XMetaL Developer).
2. The XAC file is loaded from the absolute file path specified in LoadFromFile, LoadFromString, LoadFromStringAsSGML, or Xac.

Note: If you specify a XAC in an API call, no further search rules apply. If the XAC does not exist in the location specified, a document customization will not be loaded.

3. OnResolveEntity fires. You can handle the `OnResolveEntity` event, and set a path to a document customization based on the information given in the document's document type declaration. If you do not handle OnResolveEntity, *OASIS* catalogs are used to set a path to a document customization, if possible. See the *XMetaL Customization Guide* for more information about catalogs. When OnResolveEntity or OASIS catalogs are used to set a path to a document customization, the following precedence applies to customization files:
   - If a XAC is located, the XAC will be used.
   - If a compiled rules file is located (an `.rlx`, `.rls`, or `.rld` file), the compiled rules file, along with any other customization files located in the same folder as the compiled rules file, is used.
   - If a DTD or schema is located, the DTD or schema, along with any other customization files located in the same folder as the DTD or schema, is used.

Once a customization has loaded, you can use Document.ResourceSet to access the Assets and Properties of a document customization.

Loading macro files

When a particular document is active in XMetaL XMAX, one set of macros is available. If different macros with the same name are defined in the same file, or in different files, the macro that is defined last, in the macro file that is loaded last, overrides the others.

Each DTD or schema can have a set of macros that are available if a document that uses that DTD or schema is active. For example, for the schema called `schemaname.xsd`, the macro file is `schemaname.mcr`. If you are using several DTDs or schemas, you can have a macro file for each.

Macro files are loaded when loading a document customization.

Macros are available only to the XMetaL XMAX objects that loaded the macro file. This is relevant if you have an application containing two or more XMetaL XMAX objects; the objects cannot share macros with each other. Each object must load its own instance of a macro file.

Reuse macros and scripts from previous versions of XMetaL in XMetaL XMAX

The APIs available to XMetaL XMAX are a subset of the XMetaL Author APIs. Application-level APIs are not applicable to XMetaL XMAX, as XMetaL XMAX is not an application in itself. XMetaL XMAX must be embedded in an application to function.

Scripts from previous versions of XMetaL will likely have to be modified for use with XMetaL XMAX. Only ActiveDocument and Selection objects are available in XMetaL XMAX. Many properties, methods, and event macros are not intended for use with XMetaL XMAX.

An XMetaL XMAX document can communicate with its host application through the DocumentHost interface. This interface contains a number of replacements for popular Application methods, such as Alert and MessageBox. You can acquire the DocumentHost interface with the Host property.

View a sample document customization

A sample document customization for XMetaL XMAX is provided when XMetaL Developer is installed. The Mini Journalist customization is a customization based on `journalist.dtd`, and it controls the display of

articles in XMetaL XMAX. You can view the sample customization through the XMetaL Developer program group.

Security

XMetaL XMAX is an ActiveX control. Like any ActiveX control, it needs to be secured before being allowed to run on an end-user system. XMetaL XMAX has its own security policy.

security zone	A security zone is a collection of Web sites and domains that are assigned the same level of trust.
security template	A security template is a collection of security policies that applies to a security zone.
security policy	A security policy determines whether a particular action is allowed or disallowed because of security concerns.

Security and ActiveX controls

An ActiveX control is a COM object. An ActiveX control has the same privileges as the user on the machine running the control.

This means the control can read from and write to the registry, and it can also access the local file system. ActiveX controls that are marked as safe for scripting can also run arbitrary program code on the end user's machine.

It is necessary to secure ActiveX controls to prevent them from damaging the end user's file system.

XMetaL XMAX security policy

XMetaL XMAX operates only in the Trusted Sites or Local Intranet security zones.

Note: If XMetaL XMAX is operational, it can load document customizations from any security zone, including the Restricted Sites and Internet zones.

If you are using Internet Explorer as the container application for XMetaL XMAX

If the domain or Web site that contains XMetaL XMAX is on the local intranet, Internet Explorer may automatically include the domain or Web site in the Local Intranet zone. However, it may be necessary to modify the Local Intranet zone to explicitly include the domain or web site. If the domain or Web site that contains XMetaL XMAX is on the Internet, you must modify the Trusted Sites zone to explicitly include the Web page that contains XMetaL XMAX.

In the Trusted Sites or Local Intranet zones, XMetaL XMAX is fully functional, provided that Internet Explorer is configured properly.

If you are using a custom container application for XMetaL XMAX

If your custom container application is stored and executed on the end-user's local machine, or inside a local intranet, XMetaL XMAX is fully functional. You may still be required to configure Internet Explorer to run ActiveX controls.

Using XMetaL XMAX Concurrent User License Server with a firewall

You need to perform some special configuration tasks to use XMetaL XMAX Concurrent User License Server with a firewall.

1. Install the License Server, and send the host information to JustSystems.
2. Run the license installation file provided by JustSystems.
3. Open a port (preferably with port number > 1024) on your firewall for use by the License Server, along with ports 27000 through 27009.
4. Open the `blastxml.lic` file.
5. Change the line: `VENDOR blastxml` to `VENDOR blastxml port=xxxx` where `xxxx` is the port you opened in step 3.
6. Save and close the `blastxml.lic` file.
7. Start the License Server (or reread the license file, if the License Server is already running).
8. You must specify the port number used by the License Server when using the LicenseID property in your XMetaL XMAX container application. For example, instead of `<PARAM NAME="LicenseID" VALUE="127.0.0.1">` use `<PARAM NAME="LicenseID" VALUE="127.0.0.1:xxxx">` where `xxxx` is the port you opened in step 3 above.

In-place controls

Both XMetaL Author and XMetaL XMAX enable you to use in-place ActiveX controls to represent elements in the Normal and Tags On document windows.

For example, you can:

- Display a calendar control for an element that is to contain a date
- Use a specialized image viewer to display a graphic in a format for which XMetaL Author does not have built-in support (such as CGM)
- Use a Web browser control to insert an HTML form in your document

These controls can communicate with XMetaL so that user actions performed with a control can cause the document content or markup to be modified, or conversely, changes to the document can be reflected in the in-place control.

Any ActiveX control (`.ocx` file), as defined by the Microsoft ActiveX standard, can be used as an in-place control.

ActiveX controls are embeddable, that is, they can function independently of a main application. Linked controls (those that continue to communicate with a main application) cannot be used with XMetaL. If a control displays its own menus, these are not available when the control is embedded in XMetaL.

In order to support event-callbacks, the control must support the IProvideClassInfo2 OLE interface.

Trapped keystrokes

When an in-place control has focus, XMetaL traps the following keystrokes, which are therefore not sent to the control:

- Left, Right, Up, and Down arrows
- Page Up and Down
- Home
- End

The Delete key is trapped by Windows.

Set-up in-place controls

To configure XMetaL to use an in-place control, you need to carry out the following steps.

1. Open a customization file (`.ctm` file) in XMetaL Developer.
   If the Properties window is not visible, click View Properties Window to display it.

   Note: Your customization file must be associated with a DTD or XML Schema before you can configure it to display an in-place control for an element.

2. From the list of elements that appears, choose the element that you want to associate with the control.
3. In the Display As area of the Properties window, click the **...** button next to the (Advanced Display Type) property. The Display As dialog box appears.

4. Enable the In-place control option in the Insert area of the Display As dialog box.
5. Click the **Browse** button to locate the in-place control to use.

   Select an in-place control from the list that appears, then click **OK**.You can also type the ProgId or the ClsId of the control into the ProgID/ClsID text box.
6. Type a Script Prefix.

   You will use this prefix when you write scripts that enable the control to communicate with XMetaL.
7. If you enable the Use Bitmap Printing check box, XMetaL uses a special printing mechanism when printing in-place controls. This option has been provided because some controls (for example, the Internet Explorer WebBrowser Control) do not print using the standard ActiveX mechanism. If one of your controls does not print using the default mechanism, enable the Use Bitmap Printing check box.
8. Click **OK**.
9. Save the customization file.
10. Now you need to create the macros that enable the control to communicate with XMetaL; these macros can be located in the application macro file (`xmetal.mcr`) if you intend to use them with more than one DTD or Schema, or in the DTD- or Schema-specific macro file (`dtdname.mcr`).

    The names of these macros start with the prefix that you chose earlier:

    - **prefix**_OnShouldCreate
    - **prefix**_OnInitialize
    - **prefix**_OnFocus
    - **prefix**_OnSynchronize

Specify multiple controls

Sometimes you may want to specify multiple controls for the same element. For example, you may be using the same image element to represent images in several different formats, each of which requires a different image viewer.

You can specify multiple controls for the same element in the following ways:

- Use the `DOMDocumentType.addInPlaceControlOverride` method to dynamically specify a control. This method should be called from the On_Document_Open_Complete event macro.
- Add an entry for an in-place control to the customizations file. Follow the steps below.

1. Open the customizations file `Rules\dtdname.ctm`.
2. Locate the OCXReplacements element. This element contains one or more OCXReplacement sub-elements.
3. Insert a new OCXReplacement element.
4. Inside the new OCXReplacement, insert the following sub-elements:

   - Selector
   - MacroPrefix
   - ClsId or ProgId
   - UseBmpPrinting (Optional. If this element is present, XMetaL uses the special printing mechanism when printing this control. This element is empty.)

   When the element is inserted, XMetaL calls the **prefix**_OnShouldCreate macro for each control, in the order that they were specified (in the customizations file, and/or dynamically) until one of these macros sets the ShouldCreate property to `true`.

Note: Only one control is listed in the “Treat As” tab.

Macros

You can create macros that enable XMetaL and an in-place control to communicate with each other.

Each of these macros uses the object returned by `Application.ActiveInPlaceControl`. Its properties give the macros access to the control itself.

In addition to these macros, you can create event-callback macros that carry out some action when the control signals one of its own events.

OnShouldCreate

The OnShouldCreate macro is called by XMetaL when an element that has been associated with an in-place control is inserted.

The purpose of this macro is to determine whether to create an instance of the control. For example, if a CGM-viewer control is associated with an image element, you would want to create the control only if the graphic file referred to by this element is actually a CGM file. Otherwise, XMetaL should process the element according to its assigned Treat As category (Image, List, Paragraph, General). Whether or not the control is created is determined by the macro setting the ShouldCreate property to True or False; the default is True. This macro is optional; if it is not present, XMetaL creates the control by default.

OnInitialize

The OnInitialize macro is called after the in-place control has been created in the XMetaL document window.

It performs any required one-time initialization, such as setting the default height and width of the control, or initializing the control with default values (for example, an image viewer control could be set to display a default image). This macro is optional; if it is not present, the control uses its own default settings.

OnFocus

The OnFocus macro is called when XMetaL detects the user moving into or out of an in-place control.

More specifically, it is called when:

- The user moves the insertion point into the control.
- The user moves out of the control. This includes manually moving the insertion point out of the control, switching documents, and switching to Plain Text view.

You can check the direction of user movement using the UserMovedIntoControl property.

The purpose of this macro is to make modifications to the control’s interface. You can use the control’s own properties and methods to do this, and you can also use the Height and Width properties. You cannot change the underlying XMetaL document inside this macro; for that you must use OnSynchronize.

Because OnSynchronize can carry out most of the same operations as OnFocus, you do not need to create OnFocus unless you have specific actions that must be taken when the user gains focus, or you need to use the Height and Width properties.

OnSynchronize

The OnSynchronize macro is usually the most important of the four macros. By contrast with OnFocus, OnSynchronize can modify the document.

OnSynchronize is called when:

- The document has changed.
- The user moves out of the control. This includes manually moving the insertion point out of the control, switching documents, and switching to Plain Text view.

You can check whether the document or the control needs to be updated by checking the UpdateFromDocument property.

If the content or markup of the element should determine the behavior of the control (for example, if an attribute determines which graphic is displayed), then the OnSynchronize macro needs to check whether the document has changed in a way that affects the control, and if so, update the control accordingly. You cannot use Height and Width properties inside OnSynchronize, but you can modify control using its own properties and methods.

Conversely, if the actions of the control should affect the document content or markup, then the OnSynchronize macro needs to check whether some properties of the control need to be reflected in the document (for example, if a new date has been chosen using a calendar control). It is a good idea to update the document only when the control has actually changed; otherwise, unnecessary undos may be added to the document's undo stack.

Note: Even though OnSynchronize is triggered when the user moves the selection out of the control an into the document, the macro actually runs before the location of the selection changes. Therefore, inside this macro, `ActiveDocument.Range` refers to a location within the control's corresponding element.

Event-callback macros

An in-place control signals XMetaL when one of the control's own events has occurred; this is called an *event-callback* (or sometimes an *EventSink*).

You can write a macro to carry out an action when some event is detected (you need a separate macro for each event that you want to respond to).

For an event called 'EventName', you would create a macro called EventName (with the assigned prefix for this control). For example, the IsoView CGM viewer control sends out an event whenever its FileName property changes. The corresponding event is called 'FileChanged'. If the assigned prefix is 'CGM', and you wanted to respond to this event, you would write a macro called 'CGM_FileChanged'.

Event-callback macros can also read parameters that the control passes with the event, using the `InPlaceControl.NextEventParam` property.

For a list of a control's events, consult the relevant documentation (for example, the MSDN library).

Note: In order to support event-callbacks, the control must support the IProvideClassInfo2 OLE interface.

Example

The macros in this example support the use of the Microsoft MonthView (Calendar) control. In this example the control is associated with an element that displays the date in the element content, in DD/MM/YYYY format. The OnShouldCreate macro is not required.

```
<MACRO name="Cal_OnInitialize" lang="JScript" >
<[CDATA[
var aipc = Application.ActiveInPlaceControl;
// Set the control width
aipc.Width = 400;
aipc.Height= 250;
// Initialize to an arbitrary date
// Day, Month, and Year are MonthView-specific properties
aipc.Control.Day = 16;
aipc.Control.Month = 4;
aipc.Control.Year = 1961;
]]
</MACRO>
<MACRO name="Cal_OnSynchronize" lang="JScript">
<[CDATA[
var aipc = Application.ActiveInPlaceControl;
var domnode = aipc.Node;
// If the document has changed, then update from it
if (aipc.UpdateFromDocument) {
var badDate = false;
// Get the contents of the "date" element
var dateStr = domnode.firstChild.nodeValue;
// Regular expression to match a syntactically correct date
var dateFmt = /([0-9]{1,2})\/([0-9]{1,2})\/([0-9]{4})/;
var result = dateStr.match(dateFmt);
if (result != null) {
// Get the components of the matched string
var day = result[1];
var month = result[2];
var year = result[3];
// Check the date semantics
if (day >= 1 && day <= 31
&& month >= 1 && month <= 12
&& year >= 1601) {
// Update the control with the date info
// from the document
aipc.Control.Day = day;
aipc.Control.Month = month;
aipc.Control.Year = year;
}
else {
badDate = true;
}
}
else {
badDate = true;
}
if (badDate){
// Set the control to the beginning of time
aipc.Control.Day = 16;
aipc.Control.Month = 4;
aipc.Control.Year = 1961;
}
}
// Otherwise, update document with the latest information
// from the control
else {
// Get the date specified by the control
var day = aipc.Control.Day;
var month = aipc.Control.Month;
```

```
var year = aipc.Control.Year;
var rg = domnode.ownerDocument.Range;
rg.SelectContainerContents();
// Update the element contents
rg.Text = day + "/" + month + "/" + year;
}
]]>
<MACRO>
```


Event macros

XMetaL Author defines events at which special user-defined event macros are called to perform some special processing.

In XMetaL Author, events are not explicitly associated with objects as such, as they are in some languages (for example, JScript and Visual Basic). You can think of XMetaL Author events as implicitly being associated with the application, though the macros that perform event processing usuallyhave the same names as the events themselves (for example, On_Click).

When an event macro is called, XMetaL Author first checks the macros defined for the current DTD or Schema (usually `Macros/dtdorschemaname.mcr`) and then the global XMetaL Author macro file (usually `Macros/xmetal.mcr`). If the macro in question is not defined, XMetaL Author continues with its normal operations.

XMetaL XMAX has both events and event macros. In XMetaL XMAX, events are explicitly associated with the XMetaL XMAX object. You would use events to control the appearance and behavior of the container application for XMetaL XMAX. The associated event macros provide event processing functionality within the XMetaL XMAX object itself. Events are fired before their related event macros are called.

When an event macro is called, XMetaL XMAX checks the macros defined for the current document customization. XMetaL XMAX does three things to look for macro files:

- XMetaL XMAX can acquire a macro file from a XAC, if a XAC was specified when the document was loaded.
- XMetaL XMAX can acquire the macro file from the folder (or a subfolder) of the document being loaded. In this case, the name of the macro file must have the same name as the DTD or Schema specified in the SYSTEM identifier of the document type declaration.

If the event macro in question is not defined, XMetaL XMAX continues with its normal operations.

In XMetaL Author, event macros are usually not intended to be run directly by users; they are called automatically by XMetaL when the associated events occur in an XMetaL session. You never attach these macros to a toolbar button or menu item. You may want to prevent them from appearing in the Macros dialog box and toolbar, by setting the optional `hide` attribute of the `<MACRO>` element to True. It should never be necessary to disable event macros.

Common event macros

Document-level event macros are applicable to both XMetaL Author and XMetaL XMAX.

When a document is opened or closed

The event macro On_Application_Before_Document_Open is run before a document opens, (provided it is not already open and it is being re-opened). If an application-level event macro named On_Application_Before_Document_Open exists and XMetaL Author is issued a user-driven Open Document command, XMetaL Author runs this event macro for each file that is not already opened in the list.

This macro is always run once *before* attempting to open each file in the list. It is run before the respective file is accessed or processed in any way by XMetaL Author. You can use the read-only property `Application.OpenFileName` to return the URL of the file that is currently being processed for opening. You can also set the read/write property `Application.CancelOpenDocument` to indicate to XMetaL

Author whether the Open Document operation is to be continued or aborted. The initial value of this flag property is False. In order for the Open Document operation to be discontinued the property `Application.CancelOpenDocument` has to be set to True before the end of the execution of the On_Application_Before_Document_Open macro.

The event macro On_Document_Open_Complete is run when any document is opened or created (new, open, or open template). This macro does not replace any action, but gives you access to the document at a point after it has been opened, but before it is displayed in a document window. The new document is available from the `ActiveDocument` object.

The event macro On_Document_Open_View is called once just after the document is opened and the view (Normal, Tags On, or Plain Text) is created internally by the application (but may not actually be displayed yet). This is a safe point to perform actions such as switching the view type. For example, you can use this macro to override the user's default view.

On_Document_Open_View is called after On_Document_Open_Complete.

```
// Put this code into On_Document_Open_View
// Always open documents in Plain Text view
if (ActiveDocument.ViewType!=2) {
ActiveDocument.ViewType=2;
}
```

The event macro On_Document_Close is called when a document closes.

Note: If On_Document_Close changes the document, the user is not prompted to save these changes. The changes should be saved from the macro itself.

When you open a document for the first time, the On_View_RefreshCssStyle macro is run. This macro lets you specify style overrides to the default document CSS which is determined by your customization (for example, journalist.css). A similar macro, On_StructureView_RefreshCssStyle, lets you override default formatting for the structure view (for example, journalist_structure.css).

These macros call `Document.RefreshCssStyleToAppend`, which provides a text string with your CSS rules to append. If you need to interrogate the document instance before appending the additional style information, use `Document.RefreshCssStyleDoc` instead of the ActiveDocument object (for example, `ActiveDocument.RefreshCssStyleDoc`) as it is possible ActiveDocument will be another document instance than the one being formatted.

The event macro On_Document_First_Draw is run when any document is first shown to the user. It is not subsequently run. This event is useful for showing users messages about specific document problems. Other events, such as On_Document_Open_Complete or On_Document_Open_View (among others) may be obscured to users by the splash screen.

When a simple element or attribute is validated

During validation, XMetaL calls the On_Check_Element_SimpleContent macro whenever it encounters an element with simple content. (Simple content is defined here as an element that allows only PCDATA.)

The On_Check_Attribute_Value macro is called whenever XMetaL encounters an attribute value. These two event macros are related to the OnCheckElementSimpleContent and OnCheckAttributeValue events in XMetaL XMAX.

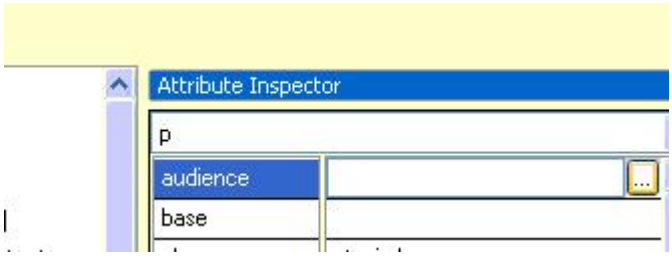
When using this macro, you can use the CheckData interface to return properties about the elements and attributes, and XMetaL's validation of them.

Note: Do not change the DOMNode during the execution of either of these event macros. Unrecoverable errors may result.

When an edit attribute button is clicked

During the On_Document_Open_Complete event-macro, your implementation will call the ShowEditButtonInAttributeInspector for each attribute where you want a custom editing UI. Each document can be uniquely configured for custom attribute editing. Each named attribute configured in this way will display a simple ellipse button beside its current value.

Here is an example:



Whenever the author clicks this ellipse button, the On_Edit_Attribute_From_AI event macro is fired. In the On_Edit_Attribute_From_AI event macro, you need to determine which attribute the user clicked upon as you may have a different custom editor depending on the attribute-type. XMetaL provides the following APIs to determine this:

- **ActiveDocument.EditAttributeNode.** Returns the DOMElement node currently selected in the dropdown list (in the above picture, this is ‘p’). This is the DOMElement owner node of the attribute and not a DOMAttribute node itself.
- **ActiveDocument.EditAttributeName.** Returns string value of the attribute name (in the above picture, this is ‘audience’).
- **ActiveDocument.EditAttributeValue.** Returns string value of the attribute value (in the above picture, this an is empty string).

Note: On_Edit_Attribute_From_AI is the only macro inside which these APIs may be called.

After your custom attribute editor has returned, you can set the new attribute by assigning it back to ActiveDocument.EditAttributeValue. If the user cancelled your custom attribute editor, leave the attribute value alone to avoid creating unnecessary undo operations on the command stack.

When the mouse is clicked

The On_Click event macro is called whenever the user clicks the mouse in the document window in Normal or Tags On view. Similarly, On_Double_Click is called when the user double-clicks. These macros are not called in Plain Text view. These two event macros are related to the XMetaL XMAX event OnClick.

On_Click is not called when you click a start-tag to collapse or expand it, or when you drag-and-drop a file. On_Double_Click is not called when you double-click an entity icon.

When these macros are called, the document's selection is the location of the mouse click. This enables the script to access information such as the current element name via the Selection interface. For example:

```
// put this code into On_Click
ActiveDocument.Host.Alert(Selection.ContainerName);
```

If both macros exist

If On_Click and On_Double_Click event macros exist, then they are both called when a user double clicks. The code in On_Click runs first followed by the code in On_Double_Click.

It is possible to place code in On_Click that prevents the user from clicking twice. In this case the code in On_Double_Click will never run.

If you define both On_Click and On_Double_Click macros, and On_Click contains `Application.Alert("message")`, `DocumentHost.Alert("message")` or `Application.Visible = false`, the code in On_Double_Click does not run.

When the view changes

The On_View_Change event macro is called when the user or a script switches between the Normal, Tags On, Plain Text, and Page Preview views.

You can obtain current and previous view types from the `Document.ViewType` and `Document.PreviousViewType` properties.

One application for this script is to restore the read-only property to parts of the document after switching from Plain Text to Normal or Tags On view. For example:

```
// put this code into On_View_Change
if ((ActiveDocument.ViewType==0 ||
ActiveDocument.ViewType==1) &&
ActiveDocument.PreviousViewType==2) {
// Code to restore read-only
// ... }
```

The event macro On_Before_View_Change, available in XMetaL Author only, fires before the user or script switches from Plain Text view to Normal or Tags On view. You can use this event macro to determine if the view change should be allowed, and to stop the view change if necessary.

One application for On_Before_View_Change is to ensure that the document is valid before allowing the user to exit from plain text view. For example:

```
' VBScript code
' place the following code in On_Before_View_Change
If (Not ActiveDocument.IsValid) Then
Dim prop
Set prop = ActiveDocument.CustomDocumentProperties.item("SQViewChangeIsOK")
prop.value = "false"
Set prop = Nothing
End If
```

When objects are dropped

You can use the `Application.AcceptDropFormat` and `Document.AcceptDropFormat` methods to enable an event macro to be called when a specific type of object is dropped into the XMetaL Author document window. When scripting forXMetaL XMAX, you can use the `Document.AcceptDropFormat` method.

These methods have two arguments: the first indicates the format name, as given by the application from which the object was dropped; the second argument is an string of your choice (no spaces) that is used in

the name of the related event macro. `Application.AcceptDropFormat` applies to all XMetaL Author documents, and should be called from the On_Application_Open macro. `Document.AcceptDropFormat` applies only to a single document, and should be called from the On_Document_Open_Complete macro.

For example, to enable an event macro when a selection is dragged and dropped from MS Word 2000, you could do the following:

```
//put this code into On_Application_Open
Application.AcceptDropFormat("HTML Format", "HTML");
```

This would cause a macro called On_Drop_HTML to be called when the selection was dropped. In general, the second argument to `AcceptDropFormat` is appended to `On_Drop_` to create the macro name.

The event macro needs to process the selection, and return a result that can be used by XMetaL. For example, converting an HTML selection into XML. This is generally done by calling a DLL that has been written for this purpose.

The following information is available inside these event macros:

- `Application.DropDataObject` or `Document.DropDataObject`: returns an IUnknown pointer to the object that was dropped. This pointer can be passed to a DLL.
- `Application.DropPoint` or `Document.DropPoint`: the Range object corresponding to the drop location.

Here is an example of this kind of macro, which calls a DLL.

```
'put this code into On_Drop_External_Text
' DLLName.MyObject is not a real DLL
Set obj = CreateObject("DLLName.MyObject")
str = obj.GetText(ActiveDocument.DropDataObject)
Set rng = ActiveDocument.DropPoint
rng.PasteString str
rng.Select
```

When a macro file is loaded

The On_Macro_File_Load macro is run when the macro file containing it is loaded in XMetaL.

You can have different versions of this macro in different macro files: for example, in the global XMetaL macro file (usually `Macros\xmetal.mcr` not available in XMetaL XMAX) and in DTD-specific macro files. This macro should not modify anything. It is intended only for defining functions and top-level variables for other macros in the file to use.

Note: Top-level variables and functions defined in On_Macro_File_Load are available only to the XMetaL XMAX object loading the macro file. This is relevant if you have an application containing two or more XMetaL XMAX objects the objects cannot share global variables defined in this manner with each other.

When objects are dragged over the editor window

The On_Drag_Over_format event is similar to OLEDragOver in Visual Basic where you might have On_Drop_XYZ defined and also On_Drag_Over_XYZ defined where `Application.DropDataObject`, `Document.DropDataObject`, and the current node are available.

Just as the Mouse_Over event gives access to the current node over which the user is hovering the mouse, the On_Drag_Over_format event gives the same type of access to that node.

The methods `Application.DropNotAllowed` and `Document.DropNotAllowed` are meant to be called only inside the On_Drag_Over_format macro and the related OnDragOver XMetaL XMAX event. The methods areused to make the cursor look like the 'no drop allowed' cursor.

The On_Drag_Over_format event is only called when dragging over a document window with the 'format' as specified by the `Application.AcceptDropFormat` or `Document.AcceptDropFormat` set for this format.

If the 'no drop allowed' cursor is showing when the drag is dropped then the On_Drop_format macro is not called.

When the context changes

The On_Update_UI event macro is called when the document context changes, and it might therefore be necessary for a script to update a toolbar button or menu item's enable status, so that you can enable/disable toolbar buttons and menu items that run macros.

The `Application.DisableMacro` and `DocumentHost.DisableMacro` methods are provided for this purpose. In XMetaL Author, these methods disable the toolbar button, menu item, and shortcut key associated with the macro (but does not remove it from the Macros dialog box or Macros toolbar). In XMetaL XMAX, `DocumentHost.DisableMacro` disables the shortcut key associated with the macro. The `Application.DisableMacro` method is available in XMetaL Author only. When On_Update_UI is called, any macros that were disabled the last time On_Update_UI was executed are automatically enabled again.

On_Update_UI is called when:

- XMetaL Author starts up
- The first document is opened in XMetaL Author
- A different document becomes the active document in XMetaL Author
- The last open document is closed in XMetaL Author
- The selection moves to a different element
- A command is executed
- A character is typed into an element (but not when subsequent characters are entered into the same element). This event is called after the initial On_Update_UI is fired as a result of a selection being moved to a different element.

On_Update_UI is not allowed to modify the document.

If On_Update_UI is not defined, all toolbars and menu items with macros are enabled.

Here is an example of disabling a macro in On_Update_UI:

```
' put this line into a macro called MyCutMacro
Selection.Cut
. . . ' put this code into On_Update_UI
if (Selection.IsInsertionPoint) Then
ActiveDocument.Host.DisableMacro("MyCutMacro")
End If
```

To see how this macro works:

- Add these macros to the global XMetaL macro file(usually `Macros\xmetal.mcr`) .
- Start XMetaL.
- Open a document.
- Choose **View** > **Toolbars** and associate the MyCutMacro macro with a toolbar and menu item.
- Make a selection in the document.

- Check that the button and menu item you just created are enabled.
- Click in text to make the selection an insertion point.
- Now check that the button and menu item are disabled.

Note: Since `Application.DisableMacro` and `DocumentHost.DisableMacro` do not remove the macro from the Macros dialog box or Macros toolbar, if you want this macro to be available only from its menu item, toolbar button, or shortcut key, you should set its 'hide' attribute to True (you should do this after you associate the macro with a toolbar and/or menu item).

To add new items to the style element drop down

Add the following to the On_Update_UI macro:

```
' This example is intended for use with XMetaL Author only.
Set se = Application.StyleElements
se.Insert -1, "New name"
```

This call puts the new value at the end of the list.

When text is dropped

There might be times when you want to view and change text that the user attempts to paste or drop into documents. You might also want to control the position that the text is pasted or dropped into documents. In addition, you might want to view what was pasted or dropped into a document to determine whether or not further checking or modification is necessary.

The On_Document_Before_DropText, On_Application_Document_Before_DropText, On_Document_After_DropText and On_Application_Document_After_DropText event macros are called when Windows causes text to be dropped into the XMetaL work area (the workspace). For example, when a user drags-and-drops text from Windows Notepad or another XMetaL document. This macro overrides any built-in default action that XMetaL may perform when the text is dropped.

The 'On_Before' events occur immediately after the user attempts to drop text. The 'On_After' events do not occur if the paste or drop action was not successful. All 'On_Before' and 'On_After' event macros can be called from the On_Application_Open event macro.

When two customizations are 'competing' for the text-dropping event, you may have to ensure programmatically that you are not compromising the operation of another script. You can use the On_Document_Before_DropText, On_Application_Document_Before_DropText, On_Document_After_DropText and On_Application_Document_After_DropText event macros to perform validation of the text before and after dropping to ensure you are performing an action that is appropriate.

An On_Document_Before_DropText, On_Application_Document_Before_DropText, On_Document_After_DropText, or On_Application_Document_After_DropText macro that processes text dropped onto the workspace must be in the application macro file (`xmetal.mcr`); a version that processes text dropped into a document must be in the document's DTD-specific macro file (`dtdname.mcr`).

In On_Document_Before_DropText, On_Application_Document_Before_DropText, On_Document_After_DropText and On_Application_Document_After_DropText you can use the `Application.DropPoint` property (available to DTD-specific versions only).

Stripping attributes

You can use the On_Document_Before_DropText and On_Document_After_DropText macros to modify strings that are moved or copied with copy-and-paste or drag-and-drop operations in a document.

A common need is to strip off ID attributes.

Here are two examples that use the journalist DTD to illustrate how the attribute stripping is done. The examples show two alternative methods of removing the ID attribute of all the copied `Para` elements.

Suppose the copied content is:

```
<Sect1>
<Para ID="a1">aaaaaaaaaaaaaaaa</Para>
<Para ID="a2">bbbbbbbbbbbbbbbb</Para>
<Para>cccccccccccccccc</Para>
</Sect1>
```

And the desired result is:

```
<Sect1>
<Para>aaaaaaaaaaaaaaaa</Para>
<Para>bbbbbbbbbbbbbbbb</Para>
<Para>cccccccccccccccc</Para>
</Sect1>
```

Here is one way of walking through DropRange to examine and modify the dropped content:

```
' put this code into On_Document_After_DropText
Set rng = ActiveDocument.DropRange
Set rngLeft = ActiveDocument.DropRange
Set rngRight = ActiveDocument.DropRange

rng.Collapse(1)
rngLeft.Collapse(1)
rngRight.Collapse(0)

rng.MoveToElement "Para"
While (rng.IsGreaterThan(rngLeft) And rng.IsLessThan(rngRight))
rng.ContainerAttribute("Id")=""
rng.MoveToElement "Para"
Wend
```

Here is how you might get the DropText before it is dropped, and use the DOM interface to modify the content:

```
var doc = new ActiveXObject("msxml2.DOMDocument");

// wrap the drop text with a dummy root element
// so that it is valid well-formed xml
// place this code in On_Document_Before_DropText

var xmlString = "<dummydocumentroot>";
xmlString += ActiveDocument.DropText;
xmlString += "</dummydocumentroot>";

doc.loadXML(xmlString);

var ndlist = doc.getElementsByTagName("Para");
var numParas = ndlist.length;

for (i = 0; i < numParas; i++) {
var para = ndlist.item(i);
var attrNode = para.getAttributeNode("Id");
if (attrNode != null) {
para.removeAttributeNode(attrNode);
}
}

var newxmlString = doc.documentElement.xml;

var str = newxmlString.replace("<dummydocumentroot>", "");
var str = str.replace("</dummydocumentroot>", "");
```



```
ActiveDocument.DropText = str;
```

Note: These event macros only occur in Tags On and Normal views.

When the mouse is moved

The event macro On_Mouse_Over is called whenever the user moves the mouse into a node (element, processing instruction, or comment) in the document window in Normal or Tags On view. Similarly, On_Mouse_Out is called when the user moves out of a node. These macros are not called in Plain Text view.

You can use `Application.MouseOverNode` or `Document.MouseOverNode` to obtain the node that the mouse is over or has just moved out of. You should use these properties only in On_Mouse_Over and On_Mouse_Out.

It is not safe for a macro to assume that the node (if any) returned by `MouseOverNode` in On_Mouse_Out is always the same as the node returned by `MouseOverNode` in the last-executed On_Mouse_Over. This can happen in the following circumstances, among others:

- The user may have selected an element (causing an On_Mouse_Over) and then pressed Delete. Since the node no longer exists, it cannot be returned by the next On_Mouse_Out.
- The document that was active when On_Mouse_Over was executed may not be active when the next On_Mouse_Out is executed. This happens if the user uses shortcut keys to close the document or switch to another document.

Four other properties that can be useful in On_Mouse_Over and On_Mouse_Out are `Application.SetCursor`, `DocumentHost.SetCursor`, `Application.SetStatusText`, and `DocumentHost.SetStatusText`.

```
// put this code into On_Mouse_Over
//
var curNode = ActiveDocument.MouseOverNode;
var curNodeName = curNode.NodeName;
if (curNodeName!="") {
ActiveDocument.Host.SetStatusText(curNodeName);
}
```

Note: Time- and processor-intensive code placed in this event macro can seriously degrade the performance of XMetaL. These effects increase when users open large documents. Be careful about adding code that may produce an unusable environment.

When a document is validated

Document validation events occur when a document or selection is validated or when a save or save as is performed.

Events occur in this order forXMetaL Author:

1. On_Application_Before_Document_Validate (all defined ones)
2. On_Before_Document_Validate

   or

   On_Application_Before_Selection_Validate (all defined ones)
3. On_Before_Selection_Validate
4. Regular XMetaL validation

5. On_Application_After_Document_Validate (all defined ones)
6. On_After_Document_Validate

   or

   On_Application_After_Selection_Validate
7. On_After_Selection_Validate

Events occur in this order for XMetaL XMAX:

1. On_Before_Document_Validate
2. On_Before_Selection_Validate
3. Regular XMetaL XMAXvalidation
4. On_After_Document_Validate
5. On_After_Selection_Validate

In XMetaL Author,you can monitor the validation inside of these macros to determine whether or not the document is valid by reading the `Application.ValidationSuccess` property. These properties return the validation status for a document up to the point in time at which it is called.

In XMetaL Author, you can also set the property Application.ValidationSuccess to False if the document is not valid from within these macros. You cannot set the ValidationSuccess to True.

In XMetaL Author, validation can be halted from within these macros by using the `Application.StopValidation` method.

Note: The calls to the On_..._Validate macros cannot be nested. Therefore, if one or more of these macros makes a call to `ActiveDocument.Validate` or any other validation methods, the event macros do not occur during that call.

When a document is spell-checked

During spell-checking, misspelled words can be replaced, at the discretion of the user, with suggestions from the dictionary.

Events exist for the moments just before and just after words in the current document are replaced with suggestions from the spell-checker:

- On_Before_SpellCheck_Replace
- On_After_SpellCheck_Replace

As their names suggest, these events surround the replace operation during spell-checking.

Because the behavior of replacing words using the spell-checker is affected by whether or not revision marking is turned on, you may want to use these events to control whether or not revision marks are created when spelling is checked in a document.

Use `Document.TrackRevisions` to turn revisions on or off.

Note: Modal dialogs called in either of these events (for example, `Application.Alert`) cause XMetaL to behave erratically, or cease functioning.

When a document is saved

The On_Document_Save event macro is called when a document is saved (by any means) under its current name. If the document is saved via the Save command (or an equivalent toolbar button or shortcut key) the macro File_Save is also called.

The event macro On_Document_SaveAs is called when a document is saved (by any means) under a new name. If the document is saved via the Save As command (or an equivalent toolbar button or shortcut key) the macro File_SaveAs is also called. Inside On_Document_SaveAs, `ActiveDocument.FullName` returns the new name of the document.

On_Before_Document_Save or On_Before_Document_SaveAs is called if a save (or save as) is initiated through any means. The macro is called just before the file is saved, and allows you to make modifications to the file (such as updating an element with the document's modification date) before the save takes place.

When a document is activated or deactivated

A document becomes active when the window displaying that document becomes the active window.

This happens when the document is created or opened, when the user activates the window containing the document, and when the document is activated via a script (see `Document.Activate`). The document is de-activated when it is closed or when the user or a script makes another document active.

Two event macros correspond to these events:

- On_Document_Activate is called when a document is activated
- On_Document_Deactivate is called when the current document is deactivated

Note: Do not call any API that displays a dialog box in On_Document_Activate or On_Document_Deactivate. APIs to avoid include `Application.Alert`, `DocumentHost.Alert`, `Application.Confirm`, `DocumentHost.Confirm`, `Application.CreateFormDlg`, `Application.MessageBox`, `DocumentHost.MessageBox`, `Application.NoticeBox`, `DocumentHost.NoticeBox`, `Application.Prompt`, and `DocumentHost.Prompt`.

When the rules file is loaded

In XMetaL Author, the On_DTD_Open_Complete event macro is run just after the rules are loaded but before the document is read in; this enables a script to (temporarily) make new elements and attributes available to the document (and all other documents using the same rules file). This macro must be in the global XMetaL macro file (usually `Macros\xmetal.mcr`).

In this macro, use `Application.NewDocumentType` to return the DOMDocumentType object corresponding to the loaded rules. Once you have this object, you can use the following DOMDocumentType methods to temporarily add new elements and attributes:

- addAttribute
- addElement
- addElementToInclusions
- addEnumeratedAttribute

XMetaL Author event macros

Application-level event macros, Element List event macros, Command Bar Macros and Attribute Inspector event macros are applicable to XMetaL Author only.

When files are dropped

The On_Drop_Files event macro is called in XMetaL Author when Windows causes files to be dropped into the document window, or into the background of the work area (the workspace); for example, when a user drags-and-drops files from Windows Explorer or the Desktop tab of the Resource Manager.

The On_Drop_Files event macro is also called when the user clicks the **Edit** > **Paste**.

This macro overrides any built-in default action that XMetaL may perform when the file is dropped (for example, image files are replaced by a defined image element that points to the file), thus allowing you to define some alternate action.

An On_Drop_Files macro that processes files dropped onto the workspacemust be in the application macro file (`xmetal.mcr`); a version that processes files dropped into a document must be in the document's DTD-specific macro file (`dtdname.mcr`).

In On_Drop_Files, you can use the following Application properties:

- DropPoint (available to DTD-specific versions only)
- DropFileCount
- DropFileName

These return the location that the files were dropped, and the number and names of the dropped files. You can also use `Document.DropPoint` in On_Drop_Files.

To carry out XMetaL's default on-drop action for files dropped into a document, you can use the `Selection.DropFile` method. This method does not trigger On_Drop_Files. The default action for files dropped into the workspace in XMetaL Author is to open them with `Documents.Open`.

On_Drop_Files is not called for HTML documents, which have their own drop file behavior.

```
' put this code into On_Drop_Files
Set rng = Application.DropPoint
Application.Alert "Dropped in element: " + rng.ContainerName
For cnt = 1 To Application.DropFileCount
Application.Alert "Dropped file: " + _
Application.DropFileName(cnt)
' Do the default drop action:
rng.DropFile(Application.DropFileName(cnt))
Next
```

File operations

Several **File** menu commands can be replaced by macros, enabling easy customization for content management systems. In all but one case, the macro can also use a COM equivalent method to call the default behavior, thereby allowing the macro to do something before and/or after the normal file operation.

Each macro replaces the action triggered by a particular menu item or its user interface equivalents, such as a toolbar button or shortcut key. However, they are not called if the action in question is triggered in a different way. For example, if from the **File** > **Save** menu is replaced by the File_Save macro, but the close file operation triggers a 'save', then the File_Save macro is not called. The On_Document_Save macro is called in this situation.

Command	Macro	COM equivalent
Open	File_Open	None
New (menu item)	File_Open_Template	Documents.OpenTemplate();
New.New (toolbar)	File_New	Documents.Add(false);
Save	File_Save	ActiveDocument.Save();
Save As	File_SaveAs	ActiveDocument.SaveAs();
Save All	File_SaveAll	Documents.Save();
Close	File_Close	ActiveDocument.Close();
Close All	File_CloseAll	Documents.Close();
Exit	File_Exit	Application.Quit(0);

The File_Open macro does not have an exact COM equivalent (which would basically be 'show the file open dialog'), but the method `Documents.Open` can be used (for example, after showing a custom **File Open** dialog box).

When the element list changes

The On_Update_ElementList event macro is called frequently by XMetaL and changes as the user's position inside a document changes. There are several properties and methods available to customizers that are intended for use only within the On_Update_ElementList macro.

Note: These event macros apply only to XMetaL Author. XMetaL XMAX does not support the ElementList interface.

Properties

- `Application.ElementList.Count`
- `Application.ElementList.Item`
- `Application.ElementList.NameToSelect`

Methods

- `Application.ElementList.InsertItem`
- `Application.ElementList.InsertSortedItem`
- `Application.ElementList.RemoveAllItems`
- `Application.ElementList.RemoveItem`

Using these properties and methods, you can work with the individual items in the element list. You can insert and remove items, as well as determine the name, description and other information about individual items (using the `Application.ElementList.Item` property).

The method `ElementList.SelectTab` must not be used in the On_Update_ElementList event macro.

Note: Because On_Update_ElementList is called frequently, it is not a good idea to create large or processor-intensive scripts in this event macro. XMetaL's performance is degraded substantially. This is because XMetaL runs an XML parser on each node location of the document to determine whether or not the document would remain valid for each item in the element list.

In the following event macros (called when the **Apply** button of the ElementList dialog box is pressed) the value of `Application.ElementList.SelectedName` is the actual name of the element list item selected.

- On_ElementList_Insert - called when the Insert radio button is selected.
- On_ElementList_Change - called when the Change radio button is selected.
- On_ElementList_Surround - called when the Insert radio button is selected and the user's selection is not empty.
- On_ElementList_Insert_NoRequired - only called if the XMetaL configuration file variable `include_required_elements` is set to False.

When the event macro is defined, the XMetaL does not perform the default behavior. The macros replace any default XMetaL behavior.

To get the original behavior for items in the list that XMetaL put there, add the following lines of code to the indicated macro:

In On_ElementList_Insert:

```
Selection.InsertWithTemplate Application.ElementList.SelectedName
```

In On_ElementList_Change:

```
Selection.ContainerName = Application.ElementList.SelectedName
```

In On_ElementList_Surround:

```
Selection.Surround Application.ElementList.SelectedName
```

In On_ElementList_Insert_NoRequired:

```
Selection.InsertElement Application.ElementList.SelectedName
```

When XMetaL starts or closes

The event macro On_Application_Open is called when XMetaL starts up, but before the workspace has been loaded (including opening previously-open documents), and toolbars have been set up. One use for this macro is to call the `Document.DisablePlainTextView` method. You can also use this macro to define a set of global variables and functions to be available in all documents.

The event macro On_Default_CommandBars_Complete is called after the workspace has been loaded and toolbars have been set up. If you want to make modifications to the menus and toolbars, you can do it here; however, you cannot affect the visibility of toolbars from this event. Instead, use the On_CommandBars_Activate or On_CommandBars_DeActivate events.

Note: Do not use `CommandBarControl.Delete` or `CommandBar.Delete` in this event macro. Also, do not create a new toolbar in this event macro. These methods are not intended to be used during the time that the toolbars are being set up. Unpredictable XMetaL behavior, including crashes, results.

The event macro On_Application_Open_Complete is called just after On_Default_CommandBars_Complete. This is a good place to open more documents.

The event macro On_Application_Close is called just before XMetaL closes down.

On_Application_Open and On_Application_Close should be defined in the global XMetaL macro file (usually `Macros/xmetal.mcr`).

When XMetaL is activated or deactivated

XMetaL is activated when its window becomes the active window as a user switches between applications. It is deactivated when another application becomes active.

The event macros On_Application_Activate and On_Application_Deactivate are called when these events occur. These macros should be defined in the global XMetaL macros file (usually `Macros\xmetal.mcr`).

One application for On_Application_Activate is to check whether a document that is open in XMetaL has been modified by another application since the last time XMetaL was active.

On_Application_Activate should not contain the following operations:

- Displaying a message box or other dialog box
- Creating new documents

On_Application_Activate and On_Application_Deactivate are not called when XMetaL starts or closes.

When a document is previewed

If you preview a document in XMetaL Author using the Page Preview view, the event macro On_Before_Document_Preview is called just before the document is previewed.

In this macro you can access the following Document properties:

- `BrowserApplication`: set or return the application used to browse
- `BrowserURL`: the document being browsed (sometimes this is a temporary file)

When the context menu is displayed

You can add menu items to the context (right-click pop-up) menu via the On_Context_Menu event macro.

This macro is called when the user right-clicks in the XMetaL document window. Inside this macro you can use the `Application.AppendMacro` method to add a new menu item: this method specifies a menu item label and a macro associated with the menu item. The menu item persists only for the current invocation of the context menu. You can also use the Selection object to identify the context of the user's selection. Use the On_Update_UI event macro to disable macros where necessary.

```
// put this code into On_Context_Menu
var rng = ActiveDocument.Range;
if (rng.ContainerName=="Graphic") {
Application.AppendMacro("I&mage Properties","ImgProps");
}
```

The application property `Application.ActiveContextMenu` can be used in the On_Context_Menu to return a `CommandBarPopup` representing the context menu that XMetaL is about to display.

When changing styles

By calling APIs, a developer can create customizations to change the tag of one element into another element.

The following APIs are intended to be used in the On_Style_Element and On_Update_UI event macros:

- `Selection.StyleElement`
- `Application.StyleElements`
- `Application.StyleElementName`
- `Application.StyleElementType`

Making this customization available in the XMetaL interface ensures that users can change the style of a current element even, in some cases, when the default behavior of XMetaL does not or cannot change the style.

The following example is how you can enable users of XMetaL to change a paragraph at the end of a section into a title of a new sibling section.

First, to check to see if the current selection is a paragraph (<Para>) at the end of a section, add this code to the On_Update_UI event macro:

```
//put this code into On_Update_UI
. . . var rng = ActiveDocument.Range;

if (rng.ContainerName == "Para") {
// the selection is in "Para"

var cntr = rng.ContainerNode;
var childs = cntr.childNodes;

if (childs && childs.length == 1) {

// only one child
var type = cntr.firstChild.nodeType;
if (type == 3 || type == 7) {
// the child is a text node or a processing instruction node
var parent = cntr.parentNode;

if (parent.nodeName == "Sect1" && parent.hasChildNodes()) {
// "Para" is in "Sect1"
// need to ignore spaces between element
var node = parent.lastChild;
while (node != null) {
if (node == cntr) {
// "Para" is the last non-text node of the "Sect1"
// Insert " Title" into style element box
var se = Application.StyleElements;
se.Insert(-1, "Sect1->Title");
break;
}
if (node.nodeType != 3) {
// "Para" is not the last non-text node
break;
}
node = node.previousSibling;
}
}
}
}
}
. . .
```

Note these two lines of the above example in particular:

```
var se = Application.StyleElements;
se.Insert(-1, "Sect1->Title");
```

`Application.StyleElements` returns a `StyleElements` object. The `StyleElements` object is then used to add the `Sect1->Title` style to the style element box:

```
StyleElements.Insert("Sect1->Title")
```

The next step is to trap the event when the user selects the style element box. Code must be added to check to see if the user chooses `Sect1->Title`. Add this code to the `On_Style_Element` event macro:

```
//put this code into On_Style_Element
var useDefault = true;
```



```
var name = Application.StyleElementName;
var type = Application.StyleElementType;

if (name == "Sect1->Title") {

var rng= ActiveDocument.Range;
var cntr = rng.ContainerNode;
var type = cntr.firstChild.nodeType;

if (type == 3) {

rng.SelectContainerContents();
var txt = rng.Text;
rng.SelectAfterContainer();
rng.SelectAfterContainer();
rng.InsertElement("Sect1");
rng.InsertElement("Title");
rng.typeText(txt);
cntr.parentNode.removeChild(cntr);
rng.Select();

} else if (type == 7) {

rng.SelectAfterContainer();
rng.SelectAfterContainer();
rng.InsertElement("Sect1");
rng.InsertElement("Title");
rng.InsertReplaceableText("Section Title");
cntr.parentNode.removeChild(cntr);
rng.Select();
}

} else {
Selection.Style = type;
}
```

In the above example, the property is used to find out what style the user selected in the style element box. If it is `Sect1->Title`, the changes to the document are made.

Selection.StyleElement vs. Selection.Style

If you want to make changes to the style of text without waiting for the user to select the style element box, use the `Selection.StyleElement` property instead.

The main difference between `Selection.StyleElement` and `Selection.Style` is that `Selection.StyleElement` changes the markup using the short description of the element (the element name specified on the General tab of the Customizations dialog box), while `Selection.Style` tries to change the currently-selected element to the tag name indicated.

When attributes are set in the Attribute Inspector

The On_Before_Set_Attribute_From_AI event macro is called after the user has input a value in the Attribute Inspector, but before the value is applied to the underlying element.

This allows the user input to be validated against business rules before insertion. All properties of the AttributeInspector interface are valid in this event macro, and the AttributeValueAfterChange property is readable and writable. All methods of the interface are usable.

The On_After_Set_Attribute_From_AI event macro is called after the user has input a value in the Attribute Inspector, and after the value is applied to the underlying element. All properties of the AttributeInspector interface are valid in this event macro, but the AttributeValueAfterChange property is read only. No AttributeInspector methods may be called from this event macro.

The following example prevents users from entering values with digits in the Attribute Inspector:

```
//XMetaL Script Language JSCRIPT:
//put this code in On_Before_Set_Attribute_From_AI event macro
AI = Application.AttributeInspector;
attrName = AI.AttributeName;
oldAttrValue = AI.AttributeValueBeforeChange;
newAttrValue = AI.AttributeValueAfterChange;
//Application.Alert("OLD: " + attrName + "=" + attrValue);
//Application.Alert("NEW: " + attrName + "=" + attrValue);

//prevent users from entering a value containing a number from 0-9
if (newAttrValue.search(/.*\d.*/) == 0)
{
AI.CancelChange();
Application.Alert("Value for [" + attrName + "] may not contain any digits.\r Please
re-enter the value for this attribute.");
}
```

When external entity references and URLs are resolved

The On_Application_Resolve_Entity event macro is provided so that you can resolve external entities.

ResolveEntityInfo

Here is a list of some circumstances when On_Application_Resolve_Entity event macro is fired:

- When references to a DTD, Schema, or parameter entity are resolved
- When an entity icon is double-clicked
- When a document or selection is validated

You can use the information contained in the `Application.ResolveEntityInfo` property to help resolve external entities. The ResolveEntityInfo object is readable and writable. Note that the `Application.ResolveEntityInfo` property is available for use only in the On_Application_Resolve_Entity event macro.

This mechanism does not bypass the catalog. The modified ResolveEntityInfo values are passed to the catalog for further resolution.

All macros named On_Application_Resolve_Entity in all application-level customizations are called. This allows for multiple integrations.

ResolveURL

The `Application.ResolveURL` property is only available for use in the On_Application_Resolve_Image_URL event macro. In addition, the On_Application_Resolve_Image_URL macro must be located in the Startup folder.

All macros named On_Application_Resolve_Image_URL in all application-level customizations are called. This allows for multiple integrations.

When command bars are activated and deactivated

You may want to enable or disable toolbars and menus using scripts.

For example, if users are switching focus between documents from different DTDs or schemas, and you have different customizations for each, you can change the toolbar appearances for each customization. This is accomplished using the following two events:

- On_CommandBars_Activate. This event occurs whenever a toolbar in XMetaL Author is about to activate, including a global common toolbar.
- On_CommandBars_DeActivate. This event occurs whenever a toolbar in XMetaL Author is about to deactivate, including a global common toolbar.

These events have value as well for different documents from the same DTD or schema; you can use code in these events to control the appearance of toolbars and menus based on the document content.

Note: Do not use `CommandBarControl.Delete` or `CommandBar.Delete` in this event macro. Also, do not create a new toolbar in this event macro. These methods are not intended to be used during the time that the toolbars are being set up. Unpredictable XMetaL behavior, including crashes, results.

On_Application_Query_Service

This XMetaL Author application-level event macro is triggered whenever the `ActiveDocument.Host.QueryService(BSTR)` is called. Note that application-level event macros are co-operative and another application-level customization may be affected.

Object

N/A

Type

Event macro

Parameters

`Application.QueriedServiceName`

`Application.QueriedServiceImpl`

Restrictions

N/A

Example

```
<MACRO name="On_Application_Query_Service">
var name = Application.QueriedServiceName;
if (name == "MyService") {
Application.QueriedServiceImpl = getMyService();
}
</MACRO>
```

Customizing using Java

Java programs can communicate with XMetaL via the COM interface using the XMetaL Java API Toolkit.

Using this toolkit, you can make API calls to XMetaL from within Java source code. The XMetaL Java API Toolkit provides a set of Java classes to make calls into the XMetaL API from the Java programming language. These classes are distributed as a `jar` file which must be added to your classpath. (See the Java API Toolkit readme file for a more complete description of the base classes used by the Java API Toolkit.)

Software requirements

In order to compile the generated Java files you will need the following:

- XMetaL 3.0 or higher, including all its requirements. See the XMetaL release notes for information about requirements for XMetaL. The Java API Toolkit is included with XMetaL Developer in XMetaL 4 or later.
- The Java Development Kit (JDK) version 1.2 or higher.

Installation

If you have already installed XMetaL, and you want to install the Java API Toolkit, consult the Maintenance Installation section of the Installation Guide (located on the XMetaL CD).The Installation Guide walks you through the process of modifying your installation of XMetaL to install additional components (such as the Java API Toolkit).

Getting started

The XMetaL Java API Toolkit exposes the following top-level classes:

- Application
- Document
- Global
- ResourceManager

The most commonly used class is the Application class, which serves as the most common entry point into the XMetaL API. For example, to access the list of currently open documents in XMetaL from Java:

```
// instantiate App with no parameters
Application xmetal = new Application();
Documents docs = xmetal.getDocuments();
// we can now do whatever we want with the set of documents
```

For more information on the XMetaL API interfaces, see the JavaDocs that accompany the Java API Toolkit, or refer to the Interfaces documentation.

Extending the XMetaL interface for your specialization

XMetaL supports specializations out-of-the-box. However, you may want to extend the XMetaL interface to support your specialization. This can include creating new menu items and toolbars, overriding settings for IDs, and controlling how properties are displayed for your specialized elements. You can customize XMetaL to support your specialization using the DitaSpecializationExtender interface.

When you create a document based on a DITA specialization, XMetaL loads and instantiates a DitaSpecializationExtender object from a JavaScript file that is saved in the same folder as the DTD for your customization. (This folder is named according to the system ID in the DTD.) The name of the JavaScript file follows the format:

```
<system_id>_<system_id>.js
```

For example, if your customization files are saved in a folder named ‘faq_shell’, the JavaScript file is named ‘faq_shell_faq_shell.js’.

Within the JavaScript file, you must define a function with the same base name of the DTD, for example,

```
function faq_shell_faq_shell() {
   return new MyDitaSpecializationExtender;
}
```

Note: To prevent possible errors as a result of the function name, the following characters are mapped to an underscore (‘_’) character:

```
;:'".&~?[]{}()|%*!=><-+^,/
```

The DitaSpecializationExtender interface is described in `..\XMetaL 9.0\Author\DITA\XACs\shared\specialize\js\DitaSpecializationExtender.js`. A sample implementation is located at `..\XMetaL 9.0\Author\DITA\XACs\{1.1 or 1.2}\ditabase`. (To test, remove ‘.off’ and start XMetaL.)

Methods

The DitaSpecializationExtender interface supports the following methods:

- **OnSpecializeCommandBars(cmdBars).** Allows you to customize menu items and toolbars for your specialization. For example, for the FAQ specialization, you may want to add a FAQ option to the Insert menu. For a description of how this method is used, refer to the Add method of the CommandBarControls object in the *XMetaL Programmer’s Guide*. Returns: void.
- **OnSpecializeCommandBars(cmdPopup).** Allows you to customize the right-click menu for your specialization. For a description of how this method is used, refer to the Add method of the CommandBarControls object in the *XMetaL Programmer’s Guide*. Returns: void.
- **makeUniqueId(domNode).** Allows you to override the auto-id generator settings for your specialization and to create your own ID algorithm. For an example implementation, refer to `..\XMetaL 9.0\Author\DITA\XACs\{1.1 or 1.2}\ditabase`. Returns: string. (If the method returns an empty string or null, the default ID generator settings are used.)
- **doProperties(range).** Allows you to control how properties are displayed. For example, you can choose to suppress the default dialog and provide your own dialog for specialized element types. You may also want to specify a domain name for your specialized elements or assign an ID of a specific format. The

range parameter is the current selection in the ActiveDocument on which the end-user asked to display properties. For an example implementation, refer to `..\XMetaL 9.0\Author\DITA\XACs\{1.1 or 1.2}\ditabase`. Returns: boolean. (If the return value is true, the default behavior will be suppressed and do nothing more. Return false if you want the default behavior to happen.

Conventions used in this document

This document is designed to give you the information necessary to begin writing scripts using the scripting interface. The following conventions facilitate this.

In the descriptions of properties, ‘Access’ states whether the property is both readable and writable, or readable only. ‘Returns’ specifies the type of value or object returned by a property or method.

When lists of arguments are specified for a property or method, the argument names describe both the semantics and the data type of the argument. For example:

```
FileExists(strPath)
```

`strPath` indicates that the argument is a string and that it should specify a file path.

Sometimes the argument is an object name, which implicitly describes both its type and semantics. For example:

```
removeAttributeNode(DOMAttr)
```

Optional arguments are enclosed in square brackets ([ ]); these characters are not part of the syntax. For example:

```
Alert(strMessage, [strTitle])
```

If an argument has a default value, it is expressed as in the following example:

```
InsertColumnsRight([longNumber=1])
```

Arguments with default values are always optional.

The expression `Interface_object` means ‘an object of type Interface’. For example:

```
DOMNode_object
```

This notation can represent any expression whose value is an object of the specified type: a variable, a global object such as `ActiveDocument`, or a compound expression such as `ActiveDocument.documentElement.firstChild`.

General interfaces

The general interfaces are general purpose interfaces for document editing.

These consist of global interfaces, utility interfaces, and special interfaces. Global interfaces have global objects available as shortcuts. Utility interfaces are accessible only through the properties and methods of global interfaces. Special interfaces provide some specialized utility functions.

Table 5: Global interfaces

Name	XMetaL Author	XMetaL XMAX
Application	Yes	No
Document	Yes	Yes
Documents	Yes	No
ResourceManager	Yes	No
Selection	Yes	Yes

Table 6: Utility interfaces

Name	XMetaL Author	XMetaL XMAX
Assets	Yes	No
CanElement	Yes	Yes
CanElementList	Yes	Yes
CheckData	Yes	Yes
Clipboard	Yes	No
Customizations	Yes	Yes
DocumentHost	Yes	Yes
DocumentProperties	Yes	Yes
DocumentProperty	Yes	Yes
ElementList	Yes	No
Find	Yes	Yes
InPlaceControl	Yes	No
NameVariantProperties		
NameVariantProperty		
Range	Yes	Yes
ResolveEntityInfo	Yes	Yes
ValidationError	Yes	Yes
ValidationErrorList	Yes	Yes

Table 7: Special interfaces

Name	XMetaL Author	XMetaL XMAX
DBImport	Yes	No

Name	XMetaL Author	XMetaL XMAX
SQExtras	Yes	No
Formatting Object	Yes	No

Application

The Application object represents the currently running instance of XMetaL. It provides information about the running application, file system information, and methods for displaying various kinds of dialog boxes. There can be only one Application object at any one time. It is globally available to all scripts only when scripting for XMetaL Author.

Properties

ActiveContextMenu

Returns a CommandBarPopup object that represents the context menu XMetaL is about to display. This property should be used only in the On_Context_Menu event macro. If it is used in any other macro, it returns null. This property is valid only for document context menus available in Plain Text, Tags On, and Normal view modes.

Applies To

XMetaL Author

Access

Read only

Returns

CommandBarPopup object

Usage in JScript

```
var p = Application.ActiveContextMenu;
```

Usage in VBScript

```
set p = Application.ActiveContextMenu
```

Example

```
// XMetaL Script Language JSCRIPT:
// Add "Insert Author" journalist command to end of context menu
// Add this code to the On_Context_Menu event macro

function OnContextMenu(){
// Valid only in Plain Text, TagsOn, and Normal
// view-modes... if (ActiveDocument.ViewType == sqViewTagsOn
|| ActiveDocument.ViewType == sqViewNormal) {
var popup = Application.ActiveContextMenu;

if (popup) {
var ctrls = popup.Controls;
var author = ctrls.Add(sqcbcTypePopup);
```

```
if (author != null) {
author.DescriptionText = "Insert &Author";
author.OnAction = "Insert Author";
var journalist2 = Application.CommandBars("Journalist 2");

if (journalist2) {
ctrls = journalist2.Controls;

if (ctrls.Item(3).OnAction == "Insert Author") {
author.FaceId = ctrls.Item(3).FaceId;
return;
}
}
author.FaceId = -1;
}

} else {
// Old API still works too!
Application.AppendMacro("Insert &Author", "Insert Author");
}
}
}
OnContextMenu();
```

ActiveDocument
Returns a Document object that represents the active document (the document in the active window). This object is similar, but not identical, to the object addressed simply as `ActiveDocument`. `Application.ActiveDocument` can be assigned to a variable, which then continues to represent the same document even when it is not the currently active document. However, if it is assigned to a variable, the value of the variable changes if a different document becomes the active document.

Applies To

XMetaL Author

Access

Read only

Returns

Document object

Usage in JScript

```
Application.ActiveDocument;
```

Usage in VBScript

```
Application.ActiveDocument
```

Example

```
// XMetaL Script Language JSCRIPT:
// bring up SaveAs dialog box for active document
Application.ActiveDocument.SaveAs();
// An equivalent form:
ActiveDocument.SaveAs();
```


ActiveInPlaceControl

Used in the OnShouldCreate, OnInitialize, OnFocus and OnSynchronize macros to obtain the IDispatch object for the in-place ActiveX control (if there is one) associated with the current element. If you are working with XMetaL XMAX, use the `Document.ActiveInPlaceControl` property.

Applies To

XMetaL Author

Access

Read only

Returns

InPlaceControl (IDispatch) object

Usage in JScript

```
Application.ActiveInPlaceControl;
```

Usage in VBScript

```
Application.ActiveInPlaceControl
```

Example

```
// XMetaL Script Language JSCRIPT:
// Control is the IsoView CGM viewer in this example
var aipc = Application.ActiveInPlaceControl;
var isoview = aipc.Control;
// Filename is a control-specific property!
isoview.Filename = "c:\\Images\\default.cgm";
```

AllUsersMacroFile

Returns the full path for the default XMetaL macro file for macros designated as available to all users (`xmetal.mcr`).

Applies To

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Application.AllUsersMacroFile;
```

Usage in VBScript

```
Application.AllUsersMacroFile
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert(Application.AllUsersMacroFile);
```

AlwaysUndoClearAfterSave

Clears the undo stack whenever `Document.Save` is called. Initially, this value is set to True.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
Application.AlwaysUndoClearAfterSave = (true | false);
```

Usage in VBScript

```
Application.AlwaysUndoClearAfterSave = (True | False)
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Application.AlwaysUndoClearAfterSave == true){
Application.Alert("You cannot undo after you save.");
}
else {
Application.Alert("You can undo after you save.");
}
```

Application

Returns an Application object that represents the XMetaL application. `Application` and `Application.Application` are synonymous.

Applies to

XMetaL Author

Access

Read only

Returns

Application object

Usage in JScript

```
Application.Application;
```

Usage in VBScript

```
Application.Application
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert("Hi there");
// You can access all methods and properties of
// the application object.

Application.Application.Alert(Application.Path);
```

AssetsPath

Returns the path for the XMetaL Assets folder.

Applies To

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Application.AssetsPath;
```

Usage in VBScript

```
Application.AssetsPath
```

Example

```
// XMetaL Script Language JSCRIPT:
// Selects the Calendar Builder folder in the upper (Explorer) pane of
the Resource Manager Assets tab
var resman = Application.ResourceManager.Assets;
astFldr = "\\Journalist DTD\\Calendar Builder";
resman.SelectFolder(Application.AssetsPath + astFldr);
```

CancelAllDocumentsAction

This property is reserved for internal use only.

CancelDocumentEventMacro
This property is reserved for internal use only.

CancelNewDocument
This property is reserved for internal use only.

CancelOpenDocument
This property can be changed only in the On_Application_Before_Document_Open event macro to cancel opening a document. Use the OpenFileName property to determine the name of the document being opened. The default value is False.

Applies to

XMetaL Author

Access

Read/write

Returns

Bool

Usage in JScript

```
var isCancel = Application.CancelOpenDocument;
Application.CancelOpenDocument = true;
```

Usage in VBScript

```
isCancel = Application.CancelOpenDocument
Application.CancelOpenDocument = True;
```

Example

```
// Put this script into On_Application_Before_Document_Open
var dn = Application.OpenFileName;
Application.Alert ("You are trying to open " + dn + ", but I will not open
 it.");
Application.CancelOpenDocument = true;
```

CancelSaveAsDocument
This property is reserved for internal use only.

CheckData
Returns the CheckData object. It is meant for use in the On_Check_Element_SimpleContent and On_Check_Attribute_Value event macros. Do not change the DOMNode during the execution of either macro; non-recoverable errors may result.

Applies to

XMetaL Author

Access

Read only

Returns

CheckData object

Usage in JScript

```
Application.CheckData;
```

Usage in VBScript

```
Application.CheckData
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put the following code
// in the On_Check_Element_SimpleContent macro

var cd = Application.CheckData;
var elem = cd.Element;
var name = elem.tagName;

// Check value.....
if(name == "Email") {
var value = cd.Value;

if (value != null) {
var re = new RegExp("^[^@ ]+@[^@ ]+$");

if (re.exec(value) == null) {
var msg = "Invalid email address value:"
msg += "'";
msg += value;
msg += "'";
cd.ValidationMsg = msg;
}
}
}


// Put the following code
// in the On_Check_Attribute_Value macro

var cd = Application.CheckData;
var elem = cd.Element;
var name = elem.tagName;
var attrName = cd.AttributeName;

if((name == "InlineGraphic" || name == "Graphic")
&& attrName == "FileRef") {
var value = new String(cd.Value);
var msg = null;

if(str.indexOf(" ")>=0) {
msg = "URL cannot have whitespace:"
}

else if (str.indexOf("\\")>= 0) {
msg = "URL cannot have backslash:"
}

if(msg) {
msg += cd.AttributeName;
msg += "='";
msg += value;
msg += "'";
```

```
cd.ValidationMsg = msg;
}
}
```

Clipboard
Returns the Clipboard object.

Applies to

XMetaL Author

Access

Read only

Returns

Clipboard object

Usage in JScript

```
var cobj = Application.Clipboard;
```

Usage in VBScript

```
set cobj = Application.Clipboard
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Get the clipboard object
Set co = Application.Clipboard
```

CommandBars
Provides access to the CommandBars interface, which you can use to customize menu bars and toolbars. You can group menu commands and toolbar buttons by adding separators using the BeginGroup property. (Separators cannot occur as the first item in a menu or toolbar.)

Applies to

XMetaL Author

Access

Read only

Returns

CommandBars object

Usage in JScript

```
Application.CommandBars;
```


Usage in VBScript

```
Application.CommandBars
```

Example

```
// XMetaL Script Language JScript:
// Get the object representing the main menu bar;
var cmdBars = Application.CommandBars;
var menuBar = CommandBars.item("Menu Bar");
```

CurrentUser
Returns the current user's login name.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Application.CurrentUser;
```

Usage in VBScript

```
Application.CurrentUser
```

Example

```
// XMetaL Script Language JScript:
Application.Alert(Application.CurrentUser);
```

CurrentUserInitials
Returns the current user's initials. By default, the initials correspond to the initials set in the user's system settings.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
vbl = Application.CurrentUserInitials;
Application.CurrentUserInitials = vbl;
```

Usage in VBScript

```
vbl = Application.CurrentUserInitials
Application.CurrentUserInitials = vbl
```

Example

```
// XMetaL Script Language JScript:
Application.CurrentUserInitials = "XML";
Application.Alert(Application.CurrentUserInitials);
```

CurrentUserMacroFile

Returns the full path for the default XMetaL macro file for macros designated as available to the current user only. The current user general macro file, `xmetal.mcr`, is saved to the personal settings folder.

Applies To

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Application.CurrentUserMacroFile;
```

Usage in VBScript

```
Application.CurrentUserMacroFile
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert(Application.CurrentUserMacroFile);
```

CurrentUserName

Returns the current user's name. By default, the name corresponds to the username set in the user's system settings.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
vbl = Application.CurrentUserName;
Application.CurrentUserName = vbl;
```

Usage in VBScript

```
vbl = Application.CurrentUserName
Application.CurrentUserName = vbl
```

Example

```
// XMetaL Script Language JScript:
Application.CurrentUserName = "XML Author";
Application.Alert(Application.CurrentUserName);
```

CurrentUserRole

Returns the current user's role.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
vbl = Application.CurrentUserRole;
Application.CurrentUserRole = vbl;
```

Usage in VBScript

```
vbl = Application.CurrentUserRole
Application.CurrentUserRole = vbl
```

Example

```
// XMetaL Script Language JScript:
Application.CurrentUserRole = "Administrator";
Application.Alert(Application.CurrentUserRole);
```

CustomProperties
Returns a NameVariantProperties object containing the custom properties for the application.

Applies to

XMetaL Author

Access

Read only

Returns

NameVariantProperties object

Usage in JScript

```
Application.CustomProperties;
```

Usage in VBScript

```
Application.CustomProperties
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the number of custom properties
var appProps;
appProps=Application.CustomProperties;
Application.Alert(docProps.Count);
```

DataObjectAsText
Asks the object on the clipboard to return object value as text. May be empty.

Applies to

XMetaL Author

Returns

String

Usage in JScript

```
var myString = Application.DataObjectAsText("strFormatName", objData);
```

`strFormatName` indicates the format to apply to the object when creating the text equivalent. `objData` represents the object to treat as text.

Usage in VBScript

```
dim myString = Application.DataObjectAsText("strFormatName", objData)
```


Example

```
// XMetaL Script Language JSCRIPT:
// treat the object on the clipboard as text
var data = Application.Clipboard.DataObject;
var txt = Application.DataObjectAsText("ASCII", data);
Application.Alert (txt);
```

DefaultSaveAsExtension
This property is reserved for internal use only.

DisplayAlerts
Enables or disables the displaying of alerts that can be issued by various operations, for example, output from the Alert method, find string not found, attempt to save an invalid file, or the Beep method. Allowed values are: sqAlertsNone (0) (= Turn off alerts), sqAlertsAll (-1) (= Turn on alerts). Use `DocumentHost.DisplayAlerts` when creating customizations that are intended to be used in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Access

Read/write

Returns

Integer

Usage in JScript

```
vbl = Application.DisplayAlerts;
Application.DisplayAlerts = intState;
```

Usage in VBScript

```
vbl = Application.DisplayAlerts
Application.DisplayAlerts = intState
```

Example

```
// XMetaL Script Language JSCRIPT:
// Turn off alerts
Application.DisplayAlerts = sqAlertsNone;
// Next two lines should display nothing
Application.Alert("Should not see this message!");
Selection.Find.Execute("xyzzyfvvkkp");
// Turn alerts back on
Application.DisplayAlerts = sqAlertsAll;
```

DocumentRuleSet
Returns the the RuleSet object, which represents a set of rules that are used by XMetaL Author to find the correct customization for the document being opened. This property is valid only in the On_Application_Set_Matching_Rules event macro.

Applies to

XMetaL Author

Access

Read only

Returns

RuleSet object

Usage in JScript

```
var rs = Application.DocumentRuleSet;
```

Usage in VBScript

```
set rs = Application.DocumentRuleSet
```

Example

```
// XMetaL Script Language JSCRIPT:
// put this code in On_Application_Set_Matching_Rules
// Retrieve the rule set
var rs = Application.DocumentRuleSet;
Application.Alert("There are " + rs.Count + " rules in the collection.")
```

Documents

Returns a Documents object that represents all the open documents. This object can be addressed simply as `Documents`.

Applies to

XMetaL Author

Access

Read only

Returns

Documents object

Usage in JScript

```
Application.Documents;
```

Usage in VBScript

```
Application.Documents
```

Example

```
// XMetaL Script Language JSCRIPT:
var doc;
```



```
var count=Application.Documents.Count;
// Iterate through Documents collection,
// and print file names
for (x=1; x <=count; x++) {
doc=Application.Documents.item(x);
Application.Alert(doc.FullName);
}
```

DropClipboardText
Returns True if the On_Document_Before_DropText event macro is called because text was pasted from the clipboard. Returns False if the macro is called because of a drag-and-drop action. This property should be used only in On_Document_Before_DropText.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Application.DropClipboardText;
```

Usage in VBScript

```
Application.DropClipboardText
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Find out if the action was a Paste or a Drag-and-Drop
If Application.DropClipboardText = True then
Msgbox ("Was pasted.")
Else
MsgBox ("Was dragged-and-dropped.")
End If
```

DropDataObject
Returns a pointer to the data object that was dropped. This property is available in the On_Drop_FmtType event macro. On_Drop_FmtType can then use this pointer to pass the object to a DLL to be interpreted. Use `Document.DropDataObject` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Access

Read only

Returns

IUnknown pointer

Usage in JScript

```
Application.DropDataObject;
```

Usage in VBScript

```
Application.DropDataObject
```

Example

```
'XMetaL Script Language VBSCRIPT:
'put this code into On_Drop_External_Text event macro
' DLLName.MyObject is not a real DLL
Set obj = CreateObject("DLLName.MyObject")
str = obj.GetText(Application.DropDataObject)
Set rng = Application.DropPoint
rng.PasteString str
rng.Select
```

DropFileCount

Use this property in the On_Drop_Files event macro to return the number of files that were dropped at a particular time.

Applies to

XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
Application.DropFileCount;
```

Usage in VBScript

```
Application.DropFileCount
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set rng = Application.DropPoint
Application.Alert "Dropped in element: " + rng.ContainerName

For cnt = 1 To Application.DropFileCount
Application.Alert "Dropped file: " _
+ Application.DropFileName(cnt)
' Do the default drop action:
```



```
rng.DropFile(Application.DropFileName(cnt))
Next
```

DropFileName
Provides access to an array containing the names of all the files that were dropped at a particular time in the On_Drop_Files event macro.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Application.DropFileName(intIndex);
```

`intIndex` is an integer from 1 to `Application.DropFileCount`.

Usage in VBScript

```
Application.DropFileName(intIndex)
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set rng = Application.DropPoint
Application.Alert "Dropped in element: " + rng.ContainerName

For cnt = 1 To Application.DropFileCount
Application.Alert "Dropped file: " + _
Application.DropFileName(cnt)
' Do the default drop action:
rng.DropFile(Application.DropFileName(cnt))
Next
```

DropInternalText
Returns True if the text is from this application and False if it is from another application. This property is intended for use in the On_Before_Drop_Text event macro.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Application.DropInternalText;
```

Usage in VBScript

```
Application.DropInternalText
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Find out if the text came from this application or another
If Application.DropInternalText = True then
MsgBox ("Came from this application.")
Else
MsgBox ("Came from another application.")
End If
```

DropPoint

If the current script was run by being dragged and dropped or pasted into a document, returns a Range object corresponding to the point at which the script was dropped; otherwise, returns a Range object corresponding to the user selection (the same as `ActiveDocument.Range`). This method is frequently used when creating asset templates and in the On_Drop_Files event macro. Use `Document.DropPoint` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Access

Read only

Returns

Range object

Usage in JScript

```
Application.DropPoint;
```

Usage in VBScript

```
Application.DropPoint
```

Example

```
// XMetaL Script Language JSCRIPT:
// Example from an asset template
var Dp = Application.DropPoint;
// Make the drop point the document's selection
Dp.Select();
var assetFolder = "\\Journalist DTD\\Calendar Builder";
```



```
Application.CopyAssetFile(Application.AssetsPath + assetFolder,
ActiveDocument.Path + assetFolder);
// Insert an image at the drop point
// Note that filename must be defined elsewhere in this script
Selection.InsertImage(filename);
```

DropRange
Returns the range that contains the contents that were pasted or dropped into the document. Use `Document.DropRange` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Access

Read only

Returns

Range

Usage in JScript

```
Application.DropRange;
```

Usage in VBScript

```
Application.DropRange
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Select the pasted or dropped text and
' surround it with paragraph tags
Application.DropRange
Selection.Surround("P")
```

DropText
Use this property to get or set the dropped text in the On_Application_Document_Before_DropText and On_Document_Before_DropText event macros. When you want to process text before it is dropped, you can use `Document.DropText`.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var strText = Application.DropText;
Application.DropText = "My text";
```

Usage in VBScript

```
dim strText = Application.DropText
Application.DropText = "My text"
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Get the dropped text and change it if it
' does not contain what we are looking for
txt = Application.DropText
If txt <> "This text was dropped." then
Application.DropText = "This text was dropped."
End If
```

DropTextIsMultiCell
If this property returns TRUE, it means the clipformat is a special XMetaL format that contains a series of table cell data. Currently, there is no API to get this table cell data or modify it. It also means that setting the DropText property has no effect.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

ElementList
Returns the ElementList object, which is the programmatic interface to the Element List. The ElementList object has its own set of properties and methods.

Applies to

XMetaL Author

Access

Read only

Returns

ElementList object

Usage in JScript

```
elmList = Application.ElementList;
```


Usage in VBScript

```
set elmList = Application.ElementList
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Add a tab to the Element List if inside
' a Journalist.dtd Article.
' Put this in On_Update_ElementList event macro.
If ActiveDocument.doctype.name = "Article" then
Set elmList = Application.ElementList
elmList.AddTab("MyTab")
End If
```

exeName

Returns the filename of the currently running XMetaL application.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var exeName = Application.exeName;
```

Usage in VBScript

```
dim exeName = Application.exeName
```

Example

```
//XMetaL Script Language JSCRIPT:
var fso = new ActiveXObject("Scripting.FileSystemObject");
var XMVersion = fso.GetFileVersion(Application.Path + "\\" +
Application.exeName);
Application.Alert(XMVersion);
```

InitComplete

Used by an external EXE to determine whether XMetaL has initialized. This property can also be used in the On_Document_Open_Complete event macro to test whether the document is being opened by the workspace (returns False) or by a user (returns True).

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
vbl = Application.InitComplete;
```

Usage in VBScript

```
vbl = Application.InitComplete
```

Example

```
' A Visual Basic Example
' NOTE: this code must reside in a standalone exe

Public Sub showOutline()
Set XMApp = CreateObject("XMetaL.Application")
XMApp.PreventExit("Outline", "Cannot exit")

Do While Not (XMApp.InitComplete)
' wait
Loop
set Doc = XMApp.ActiveDocument
XMApp.AllowExit("Outline")
Set XMApp = Nothing
' include more functionality if desired here
End Sub
```

IniVariable
Returns the value of the given XMetaL INI variable as a string.

Applies to

XMetaL Author

Access

Read/write

Returns

String

IsActivated
This method indicates whether XMetaL Author is activated with a license or operating in trial mode.

IsDocumentOpen
Returns True if the document represented by the specified Document object is currently open in XMetaL.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Application.IsDocumentOpen(Document_object);
```

Usage in VBScript

```
Application.IsDocumentOpen(Document_object)
```

Example

```
// XMetaL Script Language JSCRIPT:
// doc2 is a Document object
if (Application.IsDocumentOpen(doc2)){
Application.Alert(doc2.IsValid);
} else {
Application.Alert("Document is no longer open.");
}
```

IsMRUOpenDocumentAction

Determines if a document is open due to a selection in the most-recently-used menu item list. Use this property only in the On_Application_Before_Document_Open event macro.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Application.IsMRUOpenDocumentAction;
```

Usage in VBScript

```
Application.IsMRUOpenDocumentAction
```

Example

```
//XMetaL Script Language JSCRIPT:
if (!Application.IsMRUOpenDocumentAction)
Application.AddToRecentFileList(Application.Path +
"\\Samples\\Cameras\\CamerasInFocus.xml");
```

LastOpenDocumentPath
Gets or sets the last completed `File.Open` directory operation done by the user or (initially) the document_path variable in the XMetaL configuration file (xmetal.ini). Setting this property will cause the next `File.Open` (or equivalent) operation to start in that directory.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
Application.LastOpenDocumentPath = "strPathName";
```

Usage in VBScript

```
Application.LastOpenDocumentPath = "strPathName"
```

Example

```
//XMetaL Script Language JSCRIPT:
Application.LastOpenDocumentPath = Application.Path + "\\Samples\\Cameras"
```

LastOpenImagePath
Gets or sets the last completed `Choose.Image.File` directory operation done by the user or (initially) the image_path variable in the XMetaL configuration file (xmetal.ini). Setting this property will cause the next `Choose.Image.File` (or equivalent) operation to start in that directory.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
Application.LastOpenImagePath = "strPathName";
```

Usage in VBScript

```
Application.LastOpenImagePath = "strPathName"
```


Example

```
//XMetaL Script Language JSCRIPT:
Application.LastOpenImagePath = Application.Path +
"\\Samples\\Cameras\\images\\clipart"
```

MakeFaceId
Returns the face ID corresponding to the button specified. The face ID can then be assigned to a toolbar button using `CommandBarButton.FaceId` or to a menu item push button using `CommandBarPopUp.faceId`.

Applies to

XMetaL Author

Access

Read only

Returns

Long

Description

Note: The face ID is valid for use only in the current session of XMetaL, since face ID is a dynamic property that varies from session to session. Do not hard-code `CommandBarButton.FaceId` or `CommandBarPopUp.faceId` into a script, as this may produce unexpected results. Always use `Application.MakeFaceId` to generate a `CommandBarButton.FaceId` or `CommandBarPopUp.faceId`.

Usage in JScript

```
vbl = Application.MakeFaceId(strIconSet, longRow, longNum);
```

`strIconSet` is the name of the icon set, as displayed in the Choose Toolbar Button Image dialog box (to see this dialog box, choose **Tools** > **Macros** menu, and then click **Choose Image**). These names correspond to the folders containing the icon sets, under the `Icons` folder. Note that a row in this context does not refer to the row in which the icons are displayed in the Choose Toolbar Button Image dialog box, but rather a BMP file, underneath `Icons\strIconSet`, containing a number of icon images. For example, row 2 refers to the file `row2.bmp`.

Usage in VBScript

```
vbl = Application.MakeFaceId(strIconSet, longRow, longNum)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste into a "Journalist" DTD document
// Create a new button and assign an icon to it
var cmdBars = Application.CommandBars;
var stdBar = cmdBars.item("Standard");
var stdBarCtrls = stdBar.Controls;
var newBtn = stdBarCtrls.Add(sqcbcTypeButton);
// Get the fifth button in the first row
```

```
// of the "General (Custom)" set
var faceId = Application.MakeFaceId("General (Custom)",1,5);
newBtn.FaceId = faceId;
newBtn.OnAction="Insert Citation";
```

MouseOverNode
Returns the node corresponding to the element, comment, entity, or processing instruction that the mouse is currently over. This property should be used only inside the On_Mouse_Over and On_Mouse_Out event macros. Use `Document.MouseOverNode` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Access

Read only

Returns

DOMNode object

Usage in JScript

```
var curNode = Application.MouseOverNode;
```

Usage in VBScript

```
set curNode = Application.MouseOverNode
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the element name in the status bar
var curNode = Application.MouseOverNode;
var curNodeName = curNode.NodeName;

if (curNodeName!="") {
Application.SetStatusText = curNodeName;
}
```

NewDocumentType
Used in the On_DTD_Open_Complete event macro to obtain a DOMDocumentType object corresponding to the rules file for the document being opened. This object can then be used to temporarily add new elements and attributes to the rules.

Applies to

XMetaL Author

Access

Read only

Returns

DOMDocumentType

Usage in JScript

```
Application.NewDocumentType;
```

Usage in VBScript

```
Application.NewDocumentType
```

Example

```
 // XMetaL Script Language JSCRIPT:
var dt = Application.NewDocumentType;
dt.addElement("TechNote","Technical Note",true);
```

OpenFileName

Used in the On_Application_Before_Document_Open event macro to obtain the name of the document that XMetaL Author is about to open. You can use the `Application.CancelOpenDocument` property to cancel the opening operation.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var docName = Application.OpenFileName;
```

Usage in VBScript

```
docName = Application.OpenFileName
```

Example

```
// Put this script into On_Application_Before_Document_Open
var dn = Application.OpenFileName;
Application.Alert ("You are trying to open " + dn + ", but I will not open
 it.");
Application.CancelOpenDocument = true;
```

ParentURL

This property is reserved for internal use only.

Path

Contains the path for the currently running XMetaL application.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Application.Path;
```

Usage in VBScript

```
Application.Path
```

Example

```
// XMetaL Script Language JSCRIPT:
// Save the active document in XMetaL's
// "Document" folder
var curDoc = Application.ActiveDocument;
var xmPath = Application.Path;
var fName = curDoc.name;
curDoc.SaveAs(xmPath + "\\Document\\" + fName);
```

Properties

Returns the ResourcePropertyCollection object, which represents a collection of user-defined properties from all application packages. XMetaL searches this collection in place of variables defined in the XMetaL configuration file. If a variable is not found in this collection, XMetaL loads it from the configuration file as usual.

Applies to

XMetaL Author

Access

Read only

Returns

ResourcePropertyCollection object

Usage in JScript

```
var props = Application.Properties;
```

Usage in VBScript

```
set props = Application.Properties
```


Example

```
// XMetaL Script Language JSCRIPT:
// Get the user-defined properties from the application packages
var props = Application.Properties;
Application.Alert("There are " + props.Count + "items in the collection.");
```

QueriedServiceImpl
For use only during the event macro On_Application_Query_Service. The event macro implementor can provide an IDispatch implementation when it sees an interesting match.

Applies to

XMetaL Author

Access

Read/write

Returns

N/A

Usage in JScript

```
Application.QueriedServiceImpl;
```

Usage in VBScript

```
Application.QueriedServiceImpl
```

Example

```
var name = Application.QueriedServiceName;
if (name == "MyService") {
Application.QueriedServiceImpl = getMyService();
}
```

QueriedServiceName
For use only during the event macro On_Application_Query_Service. The event macro implementor can provide an IDispatch implementation when it sees an interesting match.

Applies to

XMetaL Author

Access

Read only

Returns

N/A

Usage in JScript

```
Application.QueriedServiceName;
```

Usage in VBScript

```
Application.QueriedServiceName
```

Example

```
var name = Application.QueriedServiceName;
if (name == "MyService") {
Application.QueriedServiceImpl = getMyService();
}
```

ResolveAttrName
This property is reserved for internal use only

ResolveAttrNS
This property is reserved for internal use only.

ResolveEntityInfo
Used to resolve external entity references. When scripting for the ResolveEntityInfo interface, create scripts only in the On_Application_Resolve_Entity event macro.

Applies to

XMetaL Author

Access

Read only

Returns

ResolveEntityInfo object

Usage in JScript

```
Application.ResolveEntityInfo;
```

Usage in VBScript

```
Application.ResolveEntityInfo
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put this code into On_Application_Resolve_Entity

function ResolveEntityURL(){
var url = Application.ResolveEntityInfo.systemID;
var re = /^iwts:.*/i; // test for interwoven URL prefix

if( !re.test(url) ){
return 0;
}
url=url.substr(5); // remove prefix
```

```
try {
var docPath=Application.ResolveEntityInfo.BasePath;
docPath=Application.URLToPath(docPath);
var newURL="c:\\entfiles" + url;
Application.ResolveEntityInfo.Filename=newURL.replace(/\//g,"\\");
} catch(e) {
Application.Alert("Unexpected exception=" + e.description);
return -1;
}
}
ResolveEntityURL();
```

ResolveNode
This property is reserved for internal use only.

ResolvePath
This property is reserved for internal use only.

ResolveURL
This property is intended to resolve a URL referencing an image that cannot be located by XMetaL. It is initialized to the value from the document that contains the entity reference. Examples: `C:\Samples\picture1.gif`, `http://www.someurl.com/images/picture1.gif`, `iwts:/pictures/picture1.gif`. You should change the value of this property only in the On_Application_Resolve_Image_URL event macro.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
Application.ResolveURL = "strURL";
```

Usage in VBScript

```
Application.ResolveURL = "strURL"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put this code in On_Application_Resolve_Image_URL
Application.ResolveURL = Application.PathToURL(
Application.Path + "\\Samples\\Cameras\\images\\clipart\\RJR-HC3000.jpg");
Application.Alert(Application.ResolveURL);
```

ResourceManager
Returns the ResourceManager object, which provides an interface for various customizations of the XMetaL Resource Manager. This object can be addressed simply as `ResourceManager`.

Applies to

XMetaL Author

Access

Read only

Returns

ResourceManager object

Usage in JScript

```
Application.ResourceManager;
```

Usage in VBScript

```
Application.ResourceManager
```

Example

```
// XMetaL Script Language JSCRIPT:
// Selects the Calendar Builder folder in the upper (Explorer) pane of
the Resource Manager Assets tab
var resman = Application.ResourceManager.Assets;
astFldr = "\\Journalist DTD\\Calendar Builder";
resman.SelectFolder(Application.AssetsPath + astFldr);
```

ResultsManager
This property is reserved for internal use only.

SaveAllDocumentsOption
This property is reserved for internal use only.

SaveAsFileName
This property is reserved for internal use only.

Selection
Returns a Selection object that represents the user's selection (which could be the insertion point) in the active document. This object can also be addressed simply as `Selection`. `Application.Selection`. It is evaluated dynamically and always takes its value from the active document. This is true even when it is represented by a variable. For example, if you define var sel=Application.Selection and a different document becomes active, a reference to `sel` returns the user's selection in the new active document. By contrast, a Range object represents the selection that was assigned to it, no matter which document is active.

Applies to

XMetaL Author

Access

Read only

Returns

Selection object

Usage in JScript

```
Application.Selection;
```

Usage in VBScript

```
Application.Selection
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display selected text
Application.Alert(Application.Selection.Text);
// Insert Para element
var select = Application.Selection;
select.insertElement("Para");
```

SkuName

Returns a string for the name of XMetaL Author variant currently executing.

Applies to

XMetaL Author

Access

Read only

Returns

N/A

Usage in JScript

```
Application.SkuName;
```

Usage in VBScript

```
Application.SkuName
```

Example

```
Application.Alert(Application.SkuName);
```

StyleElementName

Returns the name from the style element drop-down.

Applies to

XMetaL Author

Access

Read only

Returns

String

Description

This API is designed to work with `Selection.StyleElement`, `Application.StyleElements`, `Application.StyleElementType`, and `Application.StyleElementTypeNS`. By calling these APIs, a developer can create customizations to change the tag of one element into another element. Making this customization available in the XMetaL interface ensures that users can change the style of a current element even, in some cases, when the default behavior of XMetaL does not or cannot change the style. This should only be used in the On_Style_Element macro.

Usage in JScript

```
Application.StyleElementName;
```

Usage in VBScript

```
Application.StyleElementName
```

Example

```
// XMetaL Script Language JSCRIPT:

//If the user selects "Sect1->Title"
//Change the document to create a title
//Out of the last paragraph

var useDefault = true;
var name = Application.StyleElementName;
var type = Application.StyleElementType;

if (name == "Sect1->Title") {
var rng = ActiveDocument.Range;
var cntr = rng.ContainerNode;
var type = cntr.firstChild.nodeType;

if (type == 3) {
rng.SelectContainerContents();
var txt = rng.Text;
rng.SelectAfterContainer();
rng.SelectAfterContainer();
rng.InsertElement("Sect1");
rng.InsertElement("Title");
rng.typeText(txt);
cntr.parentNode.removeChild(cntr);
rng.Select();

} else if (type == 7) {
rng.SelectAfterContainer();
rng.SelectAfterContainer();
rng.InsertElement("Sect1");
rng.InsertElement("Title");
rng.InsertReplaceableText("Section Title");
cntr.parentNode.removeChild(cntr);
rng.Select();
}

} else {
Selection.Style = type;
}
```


StyleElements

This interface has this read-only property: `count` (the number of items in the list). The `StyleElements` interface has these read only methods: item(longIndex) - Returns the string at `longIndex` (ranges from 0 to count - 1). The string represents the name of the style element. Insert(longIndex, "value") - Inserts `"value"` at position `longIndex`. A negative index or an index greater than count results in the new value being added to the end of the list.

Applies to

XMetaL Author

Access

Read only

Returns

StyleElements object

Description

This API is designed to work with `Selection.StyleElement`, `Application.StyleElementName`, `Application.StyleElementType`, and `Application.StyleElementTypeNS`. By calling these APIs, a developer can create customizations to change the tag of one element into another element. Making this customization available in the XMetaL interface ensures that users can change the style of a current element even, in some cases, when the default behavior of XMetaL does not or cannot change the style. `StyleElements` should be used only in the On_Update_UI event macro.

Usage in JScript

```
Application.StyleElements;
```

Usage in VBScript

```
Application.StyleElements
```

Example

```
// XMetaL Script Language JSCRIPT:

// check to see if the current selection is a
// paragraph at the end of a section
var rng = ActiveDocument.Range;

if (rng.ContainerName == "Para") {
// the selection is in "Para"
var cntr = rng.ContainerNode;
var childs = cntr.childNodes;

if (childs && childs.length == 1) {
// only one child
var type = cntr.firstChild.nodeType;

if (type == 3 || type == 7) {
// the child is a text node or a processing instruction node
var parent = cntr.parentNode;

if (parent.nodeName == "Sect1" && parent.hasChildNodes()) {
```

```
// "Para" is in "Sect1"
// need to ignore spaces between element
var node = parent.lastChild;

while (node != null) {

if (node == cntr) {
// "Para" is the last non-text node of the "Sect1"
// Insert " Title" into style element box
var se = Application.StyleElements;
se.Insert(-1, "Sect1->Title");
break;
}

if (node.nodeType != 3) {
// "Para" is not the last non-text node
break;
}
node = node.previousSibling;
}
}
}
}
}
```

StyleElementType
Returns the element type name for the chosen element, if it is available. This property is not namespace aware.

Applies to

XMetaL Author

Access

Read only

Returns

String

Description

This API is designed to work with: `Selection.StyleElement`, `Application.StyleElements`, and `Application.StyleElementName`. By calling these APIs, a developer can create customizations to change the tag of one element into another element. Making this customization available in the XMetaL interface ensures that users can change the style of a current element even, in some cases, when the default behavior of XMetaL does not or cannot change the style. This should be used only in the On_Style_Element macro.

Usage in JScript

```
Application.StyleElementType;
```

Usage in VBScript

```
Application.StyleElementType
```


Example

```
// XMetaL Script Language JSCRIPT:

//If the user selects "Sect1->Title"
//Change the document to create a title
//Out of the last paragraph

var useDefault = true;
var name = Application.StyleElementName;
var type = Application.StyleElementType;

if (name == "Sect1->Title") {
var rng = ActiveDocument.Range;
var cntr = rng.ContainerNode;
var type = cntr.firstChild.nodeType;

if (type == 3) {
rng.SelectContainerContents();
var txt = rng.Text;
rng.SelectAfterContainer();
rng.SelectAfterContainer();
rng.InsertElement("Sect1");
rng.InsertElement("Title");
rng.typeText(txt);
cntr.parentNode.removeChild(cntr);
rng.Select();

} else if (type == 7) {
rng.SelectAfterContainer();
rng.SelectAfterContainer();
rng.InsertElement("Sect1");
rng.InsertElement("Title");
rng.InsertReplaceableText("Section Title");
cntr.parentNode.removeChild(cntr);
rng.Select();
}

} else {
Selection.Style = type;
}
```

StyleElementTypeNS
Returns the universal element type name for the chosen element, if it is available. This property is namespace aware.

Applies to

XMetaL Author

Access

Read only

Returns

String

Description

An example of a valid universal element name is `{http://www.mydomainname.com}P`. Universal element names always contain the namespace name URI in curly braces followed by the local name of the element. This API is designed to work with `Selection.StyleElement`, `Application.StyleElements`, and

`Application.StyleElementName`. By calling these APIs, a developer can create customizations to change the tag of one element into another element. Making this customization available in the XMetaL interface ensures that users can change the style of a current element even, in some cases, when the default behavior of XMetaL does not or cannot change the style. This should be used only in the On_Style_Element macro.

Usage in JScript

```
Application.StyleElementTypeNS = "{http://www.mydomainname.com}P";
var myString = Application.StyleElementTypeNS;
```

Usage in VBScript

```
Application.StyleElementTypeNS = "{http://www.mydomainname.com}P"
dim myString = Application.StyleElementTypeNS
```

Example

```
// XMetaL Script Language JSCRIPT:

//If the user selects "Sect1->Title"
//Change the document to create a title
//Out of the last paragraph
//Based on the Journalist schema
//assumes the namespace for Journalist schema is
http://www.mydomainname.com

var useDefault = true;
var name = Application.StyleElementName;
var type = Application.StyleElementTypeNS;

if (name == "Sect1->Title") {
var rng = ActiveDocument.Range;
var cntr = rng.ContainerNode;
var type = cntr.firstChild.nodeType;

if (type == 3) {
rng.SelectContainerContents();
var txt = rng.Text;
rng.SelectAfterContainer();
rng.SelectAfterContainer();
rng.InsertElementNS("http://www.mydomainname.com", "Sect1");
rng.InsertElementNS("http://www.mydomainname.com", "Title");
rng.typeText(txt);
cntr.parentNode.removeChild(cntr);
rng.Select();

} else if (type == 7) {
rng.SelectAfterContainer();
rng.SelectAfterContainer();
rng.InsertElementNS("http://www.mydomainname.com", "Sect1");
rng.InsertElement("http://www.mydomainname.com", "Title");
rng.InsertReplaceableText("Section Title");
cntr.parentNode.removeChild(cntr);
rng.Select();
}

} else {
Selection.StyleNS = type;
}
```

UnicodeSupported

Returns True if the current application is a version of XMetaL that supports Unicode™.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Application.UnicodeSupported;
```

Usage in VBScript

```
Application.UnicodeSupported
```

Example

```
// XMetaL Script Language JSCRIPT:
// check Unicode support
if (Application.UnicodeSupported == false) {
Application.Alert("UnicodeSupported property is false");
} else {
Application.Alert("UnicodeSupported property is true")
};
```

UseMultiUserLocalTBRs

This property is reserved for internal use only.

UserSettingMode

This property is reserved for internal use only.

ValidationSuccess

Returns True if the validation of the document is successful up to the point `Application.ValidationSuccess` is called. Can be set to False, but cannot be set to True. If it is set to False, XMetaL does not display the 'Validation Successful' message, even if it considers the document valid. This property is meant for use only in the On_ ... _Validate event macros.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
Application.ValidationSuccess;
```

Usage in VBScript

```
Application.ValidationSuccess
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Put this code into the On_After_Document_Validate
' Meant for a journalist.dtd article,
' Ensure users include an appendix. If they
' Do not, fail the validation and explain why.

If Application.ValidationSuccess Then
Set nds = ActiveDocument.getElementsByTagName("Appendix")

If nds.length = 0 Then
Application.ValidationSuccess = False
Application.StopValidation
MsgBox("Validation failed. Must have an appendix")
End If
End If
```

VersionNumber

Returns a string representing the version number of the currently running version of XMetaL.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
strVersionNumber = Application.VersionNumber;
```

Usage in VBScript

```
strVersionNumber = Application.VersionNumber
```

Example

```
// XMetaL Script Language JScript:
Application.Alert("The running version of XMetaL is version number " +
Application.VersionNumber + ".");
```

Visible

Determines whether the XMetaL application is hidden or visible. This property should be used only once XMetaL has completely initialized (that is, once `Application.InitComplete` returns True).

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
Application.Visible;
```

Usage in VBScript

```
Application.Visible
```

Example

```
// XMetaL Script Language JSCRIPT:
if (Application.InitComplete) {
Application.Alert("Hide XMetaL");
Application.Visible = false;
Application.Alert("Bring it back");
Application.Visible = true;
}
// else, do nothing, application has not yet initialized
```

WindowHandle

Provides access to the window handle of the main application window.

Applies to

XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
Application.WindowHandle;
```

Usage in VBScript

```
Application.WindowHandle
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert("The XMetaL application window handle is " +
Application.WindowHandle);
```

Methods

AcceptDropFormat

Enables the On_Drop_strFmtType event macro to be called when an object with the specified format is dropped into the XMetaL document window. A macro specified with this method applies to all documents. Generally this macro calls a DLL that converts the object into a format which XMetaL can use. If the object is dropped onto the background of the XMetaL work area, the macro On_Drop_Files is called instead. Use `Document.AcceptDropFormat` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies To

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.AcceptDropFormat("strFmtName", "strFmtType");
```

`strFmtName` is the name that the source application gives to that kind of object.

`strFmtType` is a string (no spaces) of your choice. Generally `AcceptDropFormat` is called in the On_Application_Open event macro.

Usage in VBScript

```
Application.AcceptDropFormat "strFmtName", "strFmtType"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Causes the On_Drop_HTML macro to be called
// when a selection is dragged from MS Word 2000
// into the XMetaL document window
ActiveDocument.AcceptDropFormat("HTML Format", "HTML");
```

AddChangedNodeKey

Used to add the changed node list identifier to a list kept by the application. This method is intended to be used prior to using any of the following: `Application.RemoveChangedNodeKey`, `Document.ClearAllChangedStatesbyKey`, `Document.ClearNodeChangedStates`, or `Document.ClearNodeChangedStatesbyKey`.

Applies To

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.AddChangedNodeKey("strKeyName");
```

`strKeyName` is the changed node list identifier.

Usage in VBScript

```
Application.AddChangedNodeKey "strKeyName"
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Add an entry to the changed node list
Application.AddChangedNodeKey "abc"
```

AddFileChangeMacro
Monitor file change events and run a macro when file content changes.

Applies to

XMetaL XMAX and XMetaL Author

Description

Allow monitoring file changes on the local file system and running XMetaL macros when events occur and XMetaL is in idle state.

The parameter, which specifies the local file system folder and file name, can include wildcard characters (for example, an asterisk (*) or a question mark (?)) in the file name.

The "file changed" event triggers the XMetaL macro "macroName" to be run. XMetaL sets the "$FileChangedPath" application-level custom property to the file path of the changed file. A VARIANT type optional parameter can be passed to **Application. AddFileChangeMacro (…)** as well. The assigned XMetaL application-level custom property contains the parameter's value when the XMetaL macro runs. Both properties are valid only during macro execution.

Returns

long - non zero if operation succeeded.

Usage in JScript

```
var ResourceData = {exInt: 0, exStr: "" };
Application.AddFileChangeMacro("filePath", "macroName", "propertyName", propertyValue);
```

The parameters are:

- `filePath` = string - local file system path to file for monitoring "file changed" event
- `macroName` = string - macro name to run
- `propertyName` = string - XMetaL Application level Custom Property name
- `propertyValue` = VARIANT - value of "propertyName" property. "propertyName" property is valid only during "macroName" macro execution.

Example

```
// XMetaL Script Language JSCRIPT:
var tempFilePath = "c:\\temp\\test\\test.xml";
var cnt = Application.AddFileChangeMacro(tempFilePath,
"MacroNameEventFileChanged", "AppPropertyNameEventFileChanged", {infoId,
 examplInt: 0, examplStr: "" });
```

AddNewTemplateDialogFileExtension
Allows extending the **File** > **New** dialog to show templates with file extensions other than .xml and .sgml.

Applies To

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.AddNewTemplateDialogFileExtension("strFileExt");
```

Usage in VBScript

```
Application.AddNewTemplateDialogFileExtension "strFileExt"
```

Example

```
' XMetaL Script Language JSCRIPT:
Application.AddNewTemplateDialogFileExtension( "ditamap" );
```

AddOpenFileDialogFilter
Extends the **File** > **Open** dialog by adding entries to the ‘Files of type’ dropdown list.

Applies To

XMetaL Author

Returns

No return value

Usage in JScript

```
AddOpenFileDialogFilter("strFilterEntry");
```

Usage in VBScript

```
AddOpenFileDialogFilter "strFilterEntry"
```


Example

```
' XMetaL Script Language JSCRIPT:
Application.AddOpenFileDialogFilter( "DITA Topics
(*.dita;*.xml)|*.dita;*.xml" );
```

AddToRecentFileList
Adds the full-path filename to the most-recently-used menu item list.

Applies to

XMetaL Author

Access

N/A

Returns

No return value

Usage in JScript

```
Application.AddToRecentFileList("strFileName");
```

Usage in VBScript

```
Application.AddToRecentFileList("strFileName")
```

Example

```
//XMetaL Script Language JSCRIPT:
Application.AddToRecentFileList("C:\\Program
Files\\XMetaL\\Author\\Samples\\Cameras\\CamerasInFocus.xml");
```

Alert
Displays an alert dialog box with the specified message and title, if supplied. Output from this method can be suppressed using `Application.DisplayAlerts`. Use `DocumentHost.Alert` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX. Do not call `Application.Alert` in the On_Document_Activate or On_Document_Deactivate event macros.

Applies To

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.Alert("strMessage", ["strTitle"]);
```

Usage in VBScript

```
Application.Alert "strMessage", ["strTitle"]
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert("Your Message Here", "Your Title");
```

AllowExit
If XMetaL has been prevented from exiting (using `Application.PreventExit`) while the application specified by `strAppName` is running, this method removes this restriction. This method is called from an external application.

Applies To

XMetaL Author

Returns

No return value

Usage in JScript

```
n/a
```

Usage in VBScript

```
n/a
```

Example

```
' A Visual Basic Example
' NOTE: this code must reside in a standalone exe

Public Sub showOutline()
Set XMApp = CreateObject("XMetaL.Application")
XMApp.PreventExit("Outline", "Cannot exit")

Do While Not (XMApp.InitComplete)
' wait
Loop
set Doc = XMApp.ActiveDocument
XMApp.AllowExit("Outline")
Set XMApp = Nothing
' include more functionality if desired here
End Sub
```

AppendMacro
Adds a menu item labeled `strLabel` to the context (right-click pop-up) menu, and associates it with the strMacroName macro. It is used only in the On_Context_Menu event macro. The menu item persists only for the current invocation of the context menu.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.AppendMacro("strLabel","strMacroName");
Application.AppendMacro("-","");
```

The ampersand (‘&’) in `strLabel` causes the character immediately following it to be the mnemonic for the menu item. To add a menu separator, use a hyphen (‘-’) as the `strLabel` value; in this case, `strMacroName` can be the null string.

Usage in VBScript

```
Application.AppendMacro "strLabel", "strMacroName"
Application.AppendMacro "-", ""
```

Example

```
// XMetaL Script Language JSCRIPT:
var rng = ActiveDocument.Range;

if (rng.ContainerName=="Graphic") {
Application.AppendMacro("-","");
Application.AppendMacro("I&mage Properties...","ImgProps");
}
```

Beep

Makes a ‘beep’ sound. This can be suppressed using `Application.DisplayAlerts`. Use `DocumentHost.Beep` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies To

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.Beep();
```

Usage in VBScript

```
Application.Beep
```

Example

```
// XMetaL Script Language JSCRIPT:
// Add this code to the On_Update_UI event macro
// However, macro can still be be run from the Macros dialog box
// or Macros toolbar while it is disabled!
if (!ActiveDocument.IsValid) {
Application.DisableMacro("SaveAsRTF");
}
```

```
// Add this code to the SaveAsRTF macro
// This will warn you if you try to run the macro while disabled
if (!ActiveDocument.IsValid) {
Application.Beep();
Application.Alert("You should not try to run this macro while the active
 document is invalid!");
}
```

Confirm
Displays a confirmation dialog box, with message and title (if specified) and containing **OK** and **Cancel** buttons. Clicking these buttons returns True and False, respectively. Use `DocumentHost.Confirm` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX. Do not call this method in the On_Document_Activate or On_Document_Deactivate event macros.

Applies To

XMetaL Author

Returns

Boolean

Usage in JScript

```
var myBoolean = Application.Confirm("strMessage", ["strTitle"]);
```

Usage in VBScript

```
dim myBoolean = Application.Confirm("strMessage", ["strTitle"])
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.DisplayAlerts = 0;
var closeFile = true;
var titleTxt = "";
var rng = ActiveDocument.Range;
var titleFound = rng.Find.Execute("<Title>");
Application.DisplayAlerts = -1;

if (titleFound) {
rng.SelectContainerContents();
titleTxt = rng.Text;
}
var dlgMsg = "Document has no title. Close anyway?";
var dlgTitle = "XML Editor";

if (!titleTxt) {
closeFile = Application.Confirm(dlgMsg,dlgTitle);
}

if (closeFile) {
ActiveDocument.Close();
}
```

CopyAssetFile
Copies the file specified to the target destination. This method creates folders if necessary.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
var myBoolean = Application.CopyAssetFile("strSrcFile", "strDestFile", [boolForceUpdate]);
```

`strSrcFile` can be a file at any location or a URL.

`strDestFile` can be in one of the following folders (and their sub-folders): the Assets folder, the folder containing the active document, the folder where documents are being opened from (the folder that the **Open** dialog box currently displays), the folder where documents are being saved to (the folder that the **Save As** dialog box currently displays).

It does not overwrite an existing file unless `boolForceUpdate` is True, and then only if the file is writable.

Usage in VBScript

```
dim myBoolean = Application.CopyAssetFile("strSrcFile", "strDestFile", [boolForceUpdate])
```

Example

```
// XMetaL Script Language JSCRIPT:
// Example from an asset template
var Dp = Application.DropPoint;
// Make the drop point the document's selection
Dp.Select();
var assetFolder = "\\Journalist DTD\\Calendar Builder";
Application.CopyAssetFile(Application.AssetsPath + assetFolder,
ActiveDocument.Path + assetFolder);
// Insert an image at the drop point
// Note that filename must be defined elsewhere in this script
Selection.InsertImage(filename);
```

CreateFormDlg

Launches the form specified. You can access the form's interface through the `Application.CreateFormDlg` in the On_Document_Activate or On_Document_Deactivate event macros. If you are working with XMetaL XMAX, use the `DocumentHost.CreateFormDlg` method.

Applies to

XMetaL Author

Returns

COM object that allows you to access controls

Usage in JScript

```
var myBoolean = Application.CreateFormDlg("strFileName", [nodeDOMNode]);
```

`strFileName` contains the full path name and filename.

The optional parameter `nodeDOMNode` indicates the DOMNode to map form data to when the form is closed.

Usage in VBScript

```
myBoolean = Application.CreateFormDlg("strFileName", [nodeDOMNode])
```

Example

```
// XMetaL Script Language JSCRIPT:
// Call an instance of myform.xft
var dlg=Application.CreateFormDlg("C:\\myform.xft");
dlg.DoModal();
dlg=null;
```

DataObjectAsText

Asks the object on the clipboard to return object value as text. May be empty.

Applies to

XMetaL Author

Returns

String

Usage in JScript

```
var myString = Application.DataObjectAsText("strFormatName", objData);
```

`strFormatName` indicates the format to apply to the object when creating the text equivalent. `objData` represents the object to treat as text.

Usage in VBScript

```
dim myString = Application.DataObjectAsText("strFormatName", objData)
```

Example

```
// XMetaL Script Language JSCRIPT:
// treat the object on the clipboard as text
var data = Application.Clipboard.DataObject;
var txt = Application.DataObjectAsText("ASCII", data);
Application.Alert (txt);
```

DisableDocumentContextMenu

Hides the context menu (right-click pop-up) from users when they attempt to view it for the active document.

Applies To

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.DisableDocumentContextMenu();
```

Usage in VBScript

```
Application.DisableDocumentContextMenu
```

Example

```
// XMetaL Script Language JSCRIPT:
// Hide the document context menu from the user
Application.DisableDocumentContextMenu();
```

DisableMacro

Disables the strMacroName macro by disabling its shortcut key, and graying out its toolbar button and menu item (if any). Use `DocumentHost.DisableMacro` when you are creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies To

XMetaL Author

Returns

No return value

Description

The strMacroName macro can still be run from the **Macros** dialog box and **Macros** toolbar. This method is intended to be used only in the On_Update_UI event macro. Whenever On_Update_UI is called, any macros that were disabled the last time On_Update_UI was executed are automatically enabled again. A macro that can be disabled in this way should check for conditions that may make it undesirable to be run from the **Macros** dialog box or **Macros** toolbar while it is disabled. For example, if the macro should not be run in this way, it can issue a 'beep' using `Application.Beep` . If you never want to run it from the Macros dialog box or Macros toolbar, set the macro's `Hidden` property to True.

Usage in JScript

```
Application.DisableMacro("strMacroName");
```

Usage in VBScript

```
Application.DisableMacro "strMacroName"
```

Example

```
// XMetaL Script Language JSCRIPT:

if (!ActiveDocument.IsValid) {
Application.DisableMacro("SaveAsRTF");
}
```

DisablePlainTextView
Disables the XMetaL Plain Text view. Designed for customizers who do not want to make this view available to users. This method should be used only in the On_Application_Open event macro.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.DisablePlainTextView();
```

Usage in VBScript

```
Application.DisablePlainTextView
```

Example

```
//XMetaL Script Language JSCRIPT:
// Place in On_Application_Open event macro to hide Plain Text view
// before workspace has been loaded
Application.DisablePlainTextView();
```

DisableToolbarContextMenu
Hides the context menu (right-click pop-up) from users when they attempt to view it for the toolbar.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.DisableToolbarContextMenu();
```

Usage in VBScript

```
Application.DisableToolbarContextMenu
```

Example

```
// XMetaL Script Language JSCRIPT:
// Hide the document context menu from the user
Application.DisableToolbarContextMenu();
```


DropNotAllowed

This method can be called only from within the On_Drag_Over_<Format> macro. When set, this changes the cursor to the 'drop not allowed' shape to provide the user with visual feedback about where objects can be dropped. Use `Document.DropNotAllowed` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.DropNotAllowed();
```

Usage in VBScript

```
Application.DropNotAllowed
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Put this code in the On_Application_Open:
Application.AcceptDropFormat "HTML Format", "HTMLFormat"

...

'Put this code in the On_Drop_HTMLFormat macro:
Set rng = Application.DropPoint
rng.PasteString "dropHTML"
rng.Select

...

'Put this code in the On_Drag_Over_HTMLFormat macro:
Set rng = Application.DropPoint

If rng.IsParentElement("Title") Then
Application.SetStatusText "Over the title element"
Application.DropNotAllowed
Else
Application.SetStatusText ""
End If
```

EmptyClipboard

Clears the Windows clipboard.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.EmptyClipboard();
```

Usage in VBScript

```
Application.EmptyClipboard
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Application.Clipboard.HasText){
Application.EmptyClipboard();
Application.Alert("Cleared text from the clipboard.");
}
```

EnableModeless

Enables or disables all currently visible modeless dialog boxes. Unexpected results may occur if one of the modeless dialog boxes becomes hidden between execution of `EnableModeless(false)` (disable) and `EnableModeless(true)` (enable); you should code so as to avoid this possibility.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.EnableModeless(booleanProp);
```

Usage in VBScript

```
Application.EnableModeless booleanProp
```

Example

```
// XMetaL Script Language JSCRIPT:
// disable modeless dialog boxes
Application.EnableModeless(false);
```

FileExists

Returns a boolean indicating whether the specified file exists.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
Application.FileExists("strPath");
```

Usage in VBScript

```
Application.FileExists("strPath")
```

Example

```
// XMetaL Script Language JSCRIPT:
var exists;
exists=Application.FileExists("c:\\data\\food.xml");
```

FileToString

Returns the contents of the file specified as a string. In effect, this enables you to read the file specified and assign it to a string variable.

Applies to

XMetaL Author

Returns

String

Usage in JScript

```
var myString = Application.FileToString("strPath");
```

Usage in VBScript

```
dim myString = Application.FileToString("strPath")
```

Example

```
// XMetaL Script Language JSCRIPT:
var docPath = "c:\\data\\mundo.txt";

if (Application.ReadableFileExists(docPath)) {
// This returns all the contents of the
// file in string format
var docToString= Application.FileToString(docPath);
}
else {
Application.Alert("Unable to read " + docPath + ".");
}
```

Help

Runs a command from the XMetaL **Help** menu as determined by the type specified.

Applies to

XMetaL Author

Returns

No return value

Description

Usage in JScript

```
Application.Help(intHelpType [, longContextID]);
```

Allowed values for `intHelpType` are:

- sqHelpContents (0)= **Contents**
- sqHelpSearch (1)= **Search for Help on...**
- sqHelpHowToUse (2)= **How to use Help**
- sqHelpTipOfTheDay (9)= **Tip Of The Day...**
- sqHelpAbout (7)= **About XMetaL Author...**
- sqHelpContext (8)= Displays a help topic with the corresponding `longContextId` in the XMetaL help file

Usage in VBScript

```
Application.Help intHelpType [, longContextID]
```

Example

```
// XMetaL Script Language JSCRIPT:
//brings up the Help Contents dialog box
Application.Help(0);
```

HelpContext

Displays the Help page containing the corresponding ID from the specified file.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.HelpContext("strHelpFile", longContextID);
```

`strHelpFile` can be an HTML Help (`.chm`) or WinHelp (`.hlp`) file.

`longContextID` is the ID of the Help page.

Usage in VBScript

```
Application.HelpContext "strHelpFile", longContextID
```


Example

```
// XMetaL Script Language JSCRIPT:
// Display help on "Printing"
Application.HelpContext("xmetal.chm", 138);
```

HelpFinder
Opens the help file specified.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.HelpFinder("strHelpFile");
```

`strHelpFile` can be an HTML Help (`.chm`) or WinHelp (`.hlp`) file.

Usage in VBScript

```
Application.HelpFinder "strHelpFile"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Open the XMetaL help file
Application.HelpFinder("xmetal.chm");
```

HideMiniContext
Hides the mini-context bar (displaying the hierarchy of the current element) from the user.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.HideMiniContext();
```

Usage in VBScript

```
Application.HideMiniContext
```

Example

```
// XMetaL Script Language JSCRIPT:
// Hide the mini-context bar from the user
Application.HideMiniContext();
```

HideSplashDialog
Hides the splash screen (XMetaL opening screen with copyright message) from the user at startup. This enables you to display a custom dialog box, prompt, or other alert when XMetaL Author opens. Note that the splash screen is not hidden if you do not supply an alternative dialog box in the On_Application_Open event macro. You can use this method only in the On_Application_Open event macro.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.HideSplashDialog();
```

Usage in VBScript

```
Application.HideSplashDialog
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Put this code into the On_Application_Open event macro:
Application.HideSplashDialog
MsgBox("On_Application_Open")
```

HideStatusBarPane
Hides the status bar pane (bar at the bottom of the XMetaL window) from the user. This method can be called only from the On_Application_Open event macro. The parameter `longValue` indicates which pane to hide, for example: 0 = Hide the leftmost pane, 4 = hide the rightmost pane.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.HideStatusBarPane(longValue);
```

Usage in VBScript

```
Application.HideStatusBarPane(longValue)
```


Example

```
// XMetaL Script Language JSCRIPT:
// Hide the left-most status bar pane
Application.HideStatusBarPane(0);
```

HideStructureView
Hides the structure view pane (the pane that displays the document structure) from the user. When the structure view pane is hidden, the UI control to open the structure view is also hidden. Users wanting to open the structure view pane can still use the **View** > **Structure View** menu item. This method can be used only inside the On_Application_Open event macro.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.HideStructureView();
```

Usage in VBScript

```
Application.HideStructureView
```

Example

```
// XMetaL Script Language JSCRIPT:
// Hide the structure view from the user
Application.HideStructureView();
```

HideViewModeButton
Hides one of the view mode buttons from the user, depending on the index number passed to it. This method can be called only from the On_Application_Open event macro.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.HideViewModeButton(intButton);
```

`intButton` indicates the index number of the button to hide: 0 = Normal view, 1 = Tags on view, 2 = Plain text view, 3 = Page preview.

Usage in VBScript

```
Application.HideViewModeButton intButton
```

Example

```
// XMetaL Script Language JSCRIPT:
// Hide the Normal view mode button from the user
var index=0
Application.HideViewModeButton(index);
```

IsProcessRunning

Check if a process is running.

Applies to

XMetaL XMAX and XMetaL Author

Returns

boolean true if process is running, otherwise false

Usage in JScript

```
Application.IsProcessRunning(ProcessID);
```

The parameters are:

- `ProcessID` = long - application process id

Example

```
// XMetaL Script Language JSCRIPT:
var is_true = Application.IsProcessRunning(ProcessID);
```

IsFileChangeMacro

Remove monitoring of a file change event.

Applies to

XMetaL XMAX and XMetaL Author

Returns

: long - number of objects that are monitoring "file changed" events. The object is identified by the "filePath + macroName" string.

Usage in JScript

```
Application.IsFileChangeMacro("filePath", "macroName");
```

The parameters are:

- `filePath` = string - local file system path to file for monitoring the "file changed" event

- `macroName` = string - macro name to run. If macroName is an empty string, then return the number of all macros associated with filePath.

Example

```
// XMetaL Script Language JSCRIPT:
var tempFilePath = "c:\\temp\\test\\test.xml";
var cnt = Application.IsFileChangeMacro(tempFilePath,
"MacroNameEventFileChanged");
```

MapXmlLangToWTDictionaryId

Maps and xml:lang attribute value to a specific WritingTools dictionary. More information is on our forum *here*.

Applies to

XMetaL Author

Returns

None

Usage in JScript

```
Application.MapXmlLangToWTDictionaryId("en-us", "US");
```

Usage in VBScript

```
Application.MapXmlLangToWTDictionaryId "en-us", "US"
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.MapXmlLangToWTDictionaryId("en-us", "US");
```

MessageBox

Displays a message box containing the text and, optionally, the title specified. Use `DocumentHost.MessageBox` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX. Do not call `Application.MessageBox` in the On_Document_Activate or On_Document_Deactivate event macros.

Applies to

XMetaL Author

Returns

Long

The return value depends on which button the user clicks.

Usage in JScript

```
var myLong = Application.MessageBox(strMessage, longButtons, [strTitle]);
```

Usage in VBScript

```
dim myLong = Application.MessageBox(strMessage, longButtons, [strTitle])
```

longButtons values

The button configuration is determined by the `longButtons` value. The first group of values (0 to 5) describes the number and type of buttons displayed in the dialog box; the second group (16, 32, 48, 64) describes the icon style; the third group (0,256,512) determines which button is the default; and the fourth group (0,4096) determines the modality of the message box. When adding numbers to create a final value for the argument buttons, use only one number from each group.

Value	Description
0	Display OK button only
1	Display OK and Cancel buttons
2	Display Abort, Retry, and Ignore buttons
3	Display Yes, No, and Cancel buttons
4	Display Yes and No buttons
5	Display Retry and Cancel buttons
16	Display Critical Message icon
32	Display Warning Query icon
48	Display Warning Message icon
64	Display Information Message icon
0	First button is default
256	Second button is default
512	Third button is default
0	Application modal; the user must respond to the message box before continuing work in the current application.
4096	System modal; all applications are suspended until the user responds to the message box.

Message box return values

Value	Button Pressed
1	OK
2	Cancel
3	Abort
4	Retry
5	Ignore
6	Yes
7	No

Example

```
// XMetaL Script Language JSCRIPT:
Application.DisplayAlerts = 0;
// Make default response "No"
var response = 7;
var rng = ActiveDocument.Range;
var titleFound = rng.Find.Execute("<Title");
Application.DisplayAlerts = -1;

if (!titleFound) {
var dlgMsg = "Document has no title. Insert one?";
var dlgTitle = "XML Editor";
// Display Ok, No, and Cancel buttons (3)
// and Warning Query icon (32)
var dlgConfig = 35;
response = Application.MessageBox(dlgMsg,dlgConfig,dlgTitle);
}

if (response==6) { // User clicked on [Yes]
rng.MoveToDocumentStart();
rng.FindInsertLocation("Title");
rng.InsertElement("Title");
ActiveDocument.Close();

} else if (response==7) { // User clicked on [No]
ActiveDocument.Close();
}
// Else response == 2 ([Cancel]): do nothing
```

MoveDropPoint
Moves the DropPoint to the insertion point specified. This method should be used in the On_Document_Before_DropText event macro. Use `Document.MoveDropPoint` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.MoveDropPoint(range);
```

`range` is an insertion point that specifies where the drop or paste occurs. It must be within the current document. If it is not an insertion point, the range is collapsed to the right, and the collapsed range is used as the new drop point.

Usage in VBScript

```
Application.MoveDropPoint range
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Set the drop point to the right of the
' user-selected text, and drop a file.
```

```
set sel=Application.Selection
Application.MoveDropPoint sel
Set rng = Application.DropPoint
' Do the default drop action:
rng.DropFile(Application.DropFileName("filename"))
```

NoticeBox
Displays a notice box containing the `strMessage` text. Use `DocumentHost.NoticeBox` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX. Do not call `Application.NoticeBox` in the On_Document_Activate or On_Document_Deactivate event macros.

Applies to

XMetaL Author

Returns

Long

Usage in JScript

```
var myLong = Application.NoticeBox("strMessage", "strBut1", ["strBut2"], ["strBut3"],
["strTitle"]);
```

Up to three buttons, with labels `strBut1`, `strBut2`, and `strBut3` can be specified. The return values are 1, 2, or 3, corresponding to the button clicked by the user. The first button is the initial default. If `strBut3` is omitted then a **Cancel** button with return value 3 is displayed. Pressing Esc or Alt+F4 always returns 3.

Usage in VBScript

```
dim myLong = Application.NoticeBox("strMessage", "strBut1", ["strBut2"], ["strBut3"],
["strTitle"])
```

Example

```
// XMetaL Script Language JSCRIPT:
var ans = Application.NoticeBox("myMessage", "Yes",
"No", "Abort", "Proceed?");
Application.Alert(ans);
```

PathToURL
Specifies a URL. Use `DocumentHost.PathToURL` if you are creating a customization that is intended to be used in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Returns

String

Usage in JScript

```
var myString = Application.PathToURL("strPath" [,"strBase"]);
```

If only `strPath` is provided, returns the URL corresponding to `strPath`. If `strBase` is specified, the return value is a URL corresponding to `strPath`, relative to `strBase`. `strBase` can be a path, relative URL, or `file:///` URL to a base document.

Usage in VBScript

```
dim myString = Application.PathToURL("strPath" [,"strBase"])
```

Example

```
// XMetaL Script Language JSCRIPT:
var u1, u2;
u1=Application.PathToURL("c:\\dir1\\dir2\\test.htm");
// Displays "file:///c|/dir1/dir2/test.htm"
Application.Alert(u1);
u2=Application.PathToURL("c:\\dir1\\images\\one.gif",
"c:\\dir1\\dir2\\test.htm");
// Displays "../images/one.gif"
Application.Alert(u2);
```

PostSetFocus
This method can be called only with the value of 1 passed to it. (Other values are reserved for future use.) When it is called, focus switches to the XMetaL Author editing window after the script containing this method has completed executing.

Applies to

XMetaL Author

Returns

N/A

Usage in JScript

```
Application.PostSetFocus(1);
```

Usage in VBScript

```
Application.PostSetFocus(1)
```

Example

```
// XMetaL Script Language JSCRIPT:
// First, make sure the editing window does not have focus
ResourceManager.Visible=1;
ResourceManager.SelectTab("Desktop");

// Now let Author know that we want focus on the
// editing window AFTER this script finishes running
Application.PostSetFocus(1);
Application.Alert("The editing window should have focus after this script
 is done.");
```

```
// ... Some more code ...
```

PreventExit
Prevents XMetaL from exiting while the strAppName application is running. If the user attempts to exit XMetaL, the strMessage appears. AllowExit removes this restriction. This method must be called from an external application. It should be used only in EXEs.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

n/a

Usage in VBScript

n/a

Example

```
' A Visual Basic Example
' NOTE: this code must reside in a standalone exe
Public Sub showOutline()
Set XMApp = CreateObject("XMetaL.Application")
XMApp.PreventExit("Outline", "Cannot exit")

Do While Not (XMApp.InitComplete)
' wait
Loop
set Doc = XMApp.ActiveDocument
XMApp.AllowExit("Outline")
Set XMApp = Nothing
' include more functionality if desired here
End Sub
```

Prompt
Displays a dialog box containing a text box with the string specified. Use `DocumentHost.Prompt` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX. Do not call `Application.Prompt` in the On_Document_Activate or On_Document_Deactivate event macros.

Applies to

XMetaL Author

Returns

String

Usage in JScript

```
var myString = Application.Prompt("strMessage", ["strText"], [intLength], [intMaxLength],
 ["strTitle"]);
```

The text box contains the default text `strText`, if specified. `intLength` and `intMaxLength` specify the size and maximum length of the text box. `strTitle` specifies a title for the dialog box. The dialog box also contains **OK** and **Cancel** buttons. The text in the entry box is returned if the user clicks **OK**; a null string is returned if the user clicks **Cancel.**

Usage in VBScript

```
dim myString = Application.Prompt("strMessage", ["strText"], ["intLength"],
["intMaxLength"], ["strTitle"])
```

Example

```
// XMetaL Script Language JSCRIPT:
// Displays a text entry box with "apple" as the default entry
var myFruit = ActiveDocument.Host.Prompt("Type the name of a fruit",
"apple");
```

PushInMacro

'Pushes in' the button for the macro specified. This method is intended to be used only in the On_Update_UI event macro. The default is for the toolbar button to not be pushed in. Use `DocumentHost.PushInMacro` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.PushInMacro("strMacroName");
```

Usage in VBScript

```
Application.PushInMacro "strMacroName"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Push the toolbar button

if (!ActiveDocument.IsValid) {
Application.PushInMacro("SaveAsRTF");
}
```

Quit

Quits XMetaL, and may save changes to documents, depending on the value specified.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.Quit([intSaveChanges]);
```

The allowed values of `intSaveChanges` are: sqPromptToSaveChanges (0) = prompt the user to save changes (the default), sqSaveChanges (1) = save changes without prompting, sqDoNotSaveChanges (2)= do not save changes.

Usage in VBScript

```
Application.Quit [intSaveChanges]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Quit and save changes
Application.Quit(1);
```

ReadableFileExists
Indicates whether the file specified exists and is readable. Returns False for nonexistent and new, unsaved documents.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
var myBoolean = Application.ReadableFileExists("strPath");
```

Usage in VBScript

```
dim myBoolean = Application.ReadableFileExists("strPath")
```

Example

```
// XMetaL Script Language JSCRIPT:
var readable;
var docPath = ActiveDocument.FullName;
readable = Application.ReadableFileExists(docPath);
```

RefreshMacros
Refreshes macros by reloading the current XMetaL macro and DTD or schema-specific macro files.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.RefreshMacros();
```

Usage in VBScript

```
Application.RefreshMacros
```

Example

```
//XMetaL Script Language JSCRIPT:
Application.RefreshMacros();
```

RegisterCRCL
This method is reserved for internal use only.

RemoveChangedNodeKey
Use this to remove changed node list identifier strings that were added using `Application.AddChangedNodeKey`.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.RemoveChangedNodeKey("strKeyName");
```

Usage in VBScript

```
Application.RemoveChangedNodeKey "strKeyName"
```

Example

```
// XMetaL Script Language JSCRIPT:
// put 'var mykey ="abc"' into On_Application_Open,
// and put the following in a local macro

if (ActiveDocument.ViewType == 2){
ActiveDocument.ViewType = 1;
}
var nds = ActiveDocument.ChangedNodesbyKey(mykey);
var l = nds.length;
var changenode = null;
var name = null;
```

```
var f = 0;
Application.Alert(l);

if (l==0){
Application.Alert("The method works!");
} else {

for (var i=0; i<=l;i++){
f++;
changenode=nds.item(i);
name = changenode.nodeName;
Application.Alert(name
+ "\n\nNewNode:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey, "NewNode",
 changenode)
+ "\nContentInserted:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"ContentInserted", changenode)
+ "\nContentDeleted:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"ContentDeleted", changenode)
+ "\nAttributesChanged:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"AttributesChanged", changenode));

if (f==nds.length){
break;
}
}
}

//Uncomment these methods one at a time running

//Application.RemoveChangedNodeKey(mykey);
//ActiveDocument.ClearAllChangedStatesbyKey(mykey);
//ActiveDocument.ClearNodeChangedStates(mykey, changenode, true);
//ActiveDocument.ClearNodeChangedStatesbyKey(mykey,changenode,false);
```

RemoveFileChangeMacro
Remove monitoring of a file change event.

Applies to

XMetaL XMAX and XMetaL Author

Returns

: long - number of objects that are monitoring "file changed" events. Object is identified by the "filePath + macroName" string.

Usage in JScript

```
Application.RemoveFileChangeMacro("filePath", "macroName");
```

The parameters are:

- `filePath` = string - local file system path to file for monitoring the "file changed" event
- `macroName` = string - macro name to run. If macroName is an empty string, then remove all macros associated with filePath.

Example

```
// XMetaL Script Language JSCRIPT:
var tempFilePath = "c:\\temp\\test\\test.xml";
```



```
var cnt = Application.RemoveFileChangeMacro(tempFilePath,
"MacroNameEventFileChanged");
```

RemoveFromRecentFileList
Removes the full-path filename from the most-recently-used menu item list.

Applies to

XMetaL Author

Access

N/A

Returns

No return value

Usage in JScript

```
Application.RemoveFromRecentFileList("strFileName");
```

Usage in VBScript

```
Application.RemoveFromRecentFileList("strFileName")
```

Example

```
//XMetaL Script Language JSCRIPT:
Application.RemoveFromRecentFileList(Application.Path +
"\\Samples\\Cameras\\CamerasInFocus.xml");
```

Run
Runs the strMacroName macro. Can be used to run both default XMetaL macros and DTD or schema-specific macros. Use `DocumentHost.Run` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.Run("strYourMacroName");
```

Usage in VBScript

```
Application.Run "strYourMacroName"
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Run("macroName");
```

RunAfterProcessDone
Run the XMetaL macro after a selected process terminates and XMetaL is in idle state.

Applies to

XMetaL XMAX and XMetaL Author

Description

XMetaL API Application.RunAfterProcessDone(…) runs the XMetaL macro after process (identified by "process ID") termination and XMetaL is in idle state.

A VARIANT type optional parameter can be passed to **Application. RunAfterProcessDone(…)**. The assigned XMetaL application-level custom property contains the parameter's value when XMetaL runs the macro.

Returns

No return value

Usage in JScript

```
var ResourceData = {exInt: 0, exStr: "" };
Application.RunAfterProcessDone(ProcessID, "MacroName", "ResourceName", ResourceData);
```

The parameters are:

- `ProcessID` = long - application process id for monitoring the "process terminated" event
- `MacroName` = string - macro name to run
- `ResourceName` = string - XMetaL Application level Custom Property name
- `ResourceData` = VARIANT - value of "ResourceName" property. "ResourceName" property is valid only during "MacroName" macro execution.

Example

```
// XMetaL Script Language JSCRIPT:
 Application.RunAfterProcessDone(appInfo.ProcessID,
"MacroNameEventProcessTerminated", "AppPropertyNameEventProcessTerminated",
 { exInt: 0, exStr: "" });
```

RunAfterProcessDone2
Run the XMetaL macro after a selected process terminates and XMetaL is in idle state.

Applies to

XMetaL XMAX and XMetaL Author

Description

XMetaL API Application.RunAfterProcessDone2(…) runs the XMetaL macro after process (identified by "process ID") termination and XMetaL is in idle state. Relies on correct process termination. On average, has better performance than RunAfterProcessDone().

A VARIANT type optional parameter can be passed to **Application. RunAfterProcessDone2(…)**. The assigned XMetaL application-level custom property contains the parameter's value when XMetaL runs the macro.

Returns

No return value

Usage in JScript

```
var ResourceData = {exInt: 0, exStr: "" };
Application.RunAfterProcessDone2(ProcessID, "MacroName", "ResourceName", ResourceData);
```

The parameters are:

- `ProcessID` = long - application process id for monitoring the "process terminated" event
- `MacroName` = string - macro name to run
- `ResourceName` = string - XMetaL Application level Custom Property name
- `ResourceData` = VARIANT - value of "ResourceName" property. "ResourceName" property is valid only during "MacroName" macro execution.

Example

```
// XMetaL Script Language JSCRIPT:
 Application.RunAfterProcessDone2(appInfo.ProcessID,
"MacroNameEventProcessTerminated", "AppPropertyNameEventProcessTerminated",
 { exInt: 0, exStr: "" });
```

RunMacroOnIdle

Runs the XMetaL macro when idle.

Applies to

XMetaL XMAX and XMetaL Author

Description

XMetaL API Application.RunMacroOnIdle(macroName, mlsDelay, propertyName, propertyValue) pushes the macro into the queue of macros that XMetaL runs when in idle state. An optional command line parameter can set a minimal delay interval in milliseconds before macro can run. Optional parameter "propertyValue" type of VARIANT can be passed to Application.RunMacroOnIdle(…). Assigned XMetaL Application level custom property contains parameter's value when XMetaL macro runs.

This API allows preventing interruptions to the execution of time-sensitive actions or macros (for example xmetal ON_UPDATE_UI event macros).

Returns

long - non zero if operation succeeded.

Usage in JScript

```
var propertyValue = {exInt: 0, exStr: "" };
Application.RunMacroOnIdle( macroName, mlsDelay, propertyName, propertyValue);
```

The parameters are:

- `macroName` = string - macro name to run
- `mlsDelay` = long - minimum delay interval in milliseconds before macro can run
- `propertyName` = string - XMetaL application-level custom property name
- `propertyValue` = VARIANT - value of "propertyName" property; "propertyName" property is valid only during "macroName" macro execution.

Example

```
// XMetaL Script Language JSCRIPT:
var delay = 1000;
var cnt = Application.RunMacroOnIdle("MacroName", delay, "AppPropertyName",
 {infoId, examplInt: 0, examplStr: "" });
```

RunKeyedMacro
Runs the macro that has the specified shortcut key. Can be used to run both default XMetaL macros and DTD or schema-specific macros. Use `DocumentHost.RunKeyedMacro` when you are creating a customization for use in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.RunKeyedMacro("strShortcutKey");
```

Usage in VBScript

```
Application.RunKeyedMacro "strShortcutKey"
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.RunKeyedMacro("Ctrl+Alt+G");
```

SetCursor
Sets the mouse cursor according to the value specified. To control the cursor shape within a document, use this method in the On_Mouse_Over and On_Mouse_Out event macros. To control the cursor shape when a

document is activated and deactivated, use this method in the On_Document_Activate and On_Document_Deactivate event macros (typically you would set the cursor to 0 in these macros). Use `DocumentHost.SetCursor` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Returns

No return value

Description

Usage in JScript

```
Application.SetCursor(intType);
```

Allowed values are:

- sqDefaultCursor (0): pointer
- sqCursorArrow (1): arrow
- sqCursorBusy (2): ‘busy’
- sqCursorIBeam (3): I-beam
- sqCursorHand (4): open hand
- sqCursorGrabHand (5): closed hand
- sqCursorParaArrow (6): paragraph arrow
- sqCursorRightArrow (7): right arrow
- sqCursorCrossHair (8): cross hair
- sqCursorLeftRightDrag (9): drag right or left
- sqCursorUpDownDrag (10): drag up or down
- sqCursorResizeNESW (11): resize (northeast to southwest orientation)
- sqCursorResizeNWSE (12): resize (northwest to southeast orientation)
- sqCursorVDouble (13): vertical double arrow
- sqCursorHDouble (14): horizontal double arrow
- sqCursorVArrow (15): vertical single arrow
- sqCursorMacroRecording (16): ‘macro recording’ (cassette)
- sqCursorNotAllowed (17): ‘not allowed’

XMetaL sometimes overrides the `SetCursor` setting, for example, when macro recording is taking place. You can define up to 10 extra cursors by setting **cursor_file0**, ..., `cursor_file9` in the XMetaL configuration file to the names of `.cur` files. These variables default to `Cursors\cursor0.cur`, ..., `Cursors\cursor9.cur` in the XMetaL folder. User-defined cursors have `intType` values 18 through 27. These correspond to the constants **sqUserDefinedCursor0**, ..., **sqUserDefinedCursor9**.

Usage in VBScript

```
Application.SetCursor intType
```

Example

```
// XMetaL Script Language JSCRIPT:
// Set the cursor to "not allowed"
// if mouse is over a read-only container,
// and to the default cursor otherwise. //
// Get the node the mouse is over
var curNode=Application.MouseOverNode;
var rng = ActiveDocument.Range;
// Select the contents of the node
rng.SelectNodeContents(curNode);
// Check the read-only status and
// set the cursor accordingly

if (rng.ReadOnly) {
Application.SetCursor(17);
} else {
Application.SetCursor(0);
}
```

SetForegroundWindow
Brings XMetaL Author to the foreground and activates it.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.SetForegroundWindow();
```

Usage in VBScript

```
Application.SetForegroundWindow()
```

Example

```
// XMetaL Script Language JSCRIPT:
// bring me to the front
Application.SetForegroundWindow();
```

SetStatusText
Sets the text at the left end of the status bar to the string specified. XMetaL sometimes overrides this setting, for example, when the mouse is over a toolbar button. Use `DocumentHost.SetStatusText` when creating customizations that are intended to run in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.SetStatusText("strText");
```

Usage in VBScript

```
Application.SetStatusText "strText"
```

Example

```
// XMetaL Script Language JSCRIPT:
// If the element that the mouse is
// over has a LINKEND attribute,
// display its value in the status bar. //
// Get the node the mouse is over
var curNode = Application.MouseOverNode;
var Linkend = curNode.getAttribute("LINKEND");
// Check if the attribute value is non-null
// and set the status text accordingly

if (Linkend) {
Application.SetStatusText(Linkend);
} else {
Application.SetStatusText("");
}
```

SetWorkbookMode

When set to True, workbook mode is enabled. When set to False (the default), workbook mode is disabled. Workbook mode is an XMetaL Author interface mode in which each open document has a tab at the bottom of the window, which can be used to identify and select the document.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.SetWorkbookMode([boolWorkbookModeEnabled]);
```

Usage in VBScript

```
Application.SetWorkbookMode [boolWorkbookModeEnabled]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Turn off the workbook mode
Application.SetWorkbookMode();
```

ShowPage

Displays the document specified using the default browser. This command is intended to enable you to easily create new **Help** menu commands.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ShowPage("strURL");
```

Usage in VBScript

```
Application.ShowPage "strURL"
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.ShowPage("c:\\Data\\PetFood\\about.xml");
```

ShowWindow

Changes the state of the XMetaL Author application window.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ShowWindow(longCmdShow);
```

For `longCmdShow`, use the following values:

- `SW_HIDE 0`
- `SW_SHOWNORMAL 1`
- `SW_NORMAL 1`
- `SW_SHOWMINIMIZED 2`
- `SW_SHOWMAXIMIZED 3`
- `SW_MAXIMIZE 3`
- `SW_SHOWNOACTIVATE 4`
- `SW_SHOW 5`
- `SW_MINIMIZE 6`
- `SW_SHOWMINNOACTIVE 7`
- `SW_SHOWNA 8`
- `SW_RESTORE 9`
- `SW_SHOWDEFAULT 10`
- `SW_FORCEMINIMIZE 11`

- SW_MAX 11

Usage in VBScript

```
Application.ShowWindow(longCmdShow)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Minimizes XMetaL Author
var SW_SHOWMINIMIZED = 2
Application.ShowWindow(SW_SHOWMINIMIZED);
```

StopValidation

Stops the validation process. This property is meant for use in the On_ ... Validate event macros.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.StopValidation;
```

Usage in VBScript

```
Application.StopValidation
```

Example

```

' XMetaL Script Language VBSCRIPT:
' Add this code to the On_Click event macro
' Prompt the user to stop validation if
' The mouse is clicked
response=MsgBox ("Do you want to stop validation?" , vbYesNo)

If response = vbYes then
Application.StopValidation
End If
```

TerminateProcessEx

Terminate running process.

Applies to

XMetaL XMAX and XMetaL Author

Returns

long - reserved internal use value.

Usage in JScript

```
Application.TerminateProcessEx(ProcessID, Parameter1, Parameter2, Parameter3);
```

The parameters are:

- `ProcessID` = long
- `Parameter1` = long
- `Parameter2` = long
- `Parameter3` = long

Combination of Parameter1, Parameter2, Parameter3:

0 : 0: 0 - "brute force" to terminate process. Uses windows API TerminateProcess(…)

-1 : 0 : 0 - find window by process id and send WM_CLOSE message

-1: -1: 0 - find window by process id and post WM_CLOSE message

Parameter1: contains window handle (> 0)

Parameter1: 0 : 0 - send WM_CLOSE message to window

Parameter1: -1: 0 - post WM_CLOSE message to window

Parameter2: > 0 use windows API SendMessageTimeout(…)

::SendMessageTimeout(xxx, WM_CLOSE, 0, NULL, Parameter3, Parameter2, …);

Example

```
// XMetaL Script Language JSCRIPT:
  Application.TerminateProcessEx(procID, -1, -1, 0); // terminate
Application process with id 'procID' by post message WM_CLOSE
```

UniqueFileName
Returns a filename, starting with the specified prefix, that is not present in the specified folder. The filename is assigned an extension if specified; otherwise it is assigned the `.tmp` extension.

Applies to

XMetaL Author

Returns

String

Usage in JScript

```
var myString = Application.UniqueFileName("strDirectory", "strPrefix", ["strExtension"]);
```

`strPrefix` can be up to three characters long.

Usage in VBScript

```
dim myString = Application.UniqueFileName("strDirectory", "strPrefix", ["strExtension"])
```


Example

```
// XMetaL Script Language JSCRIPT:
// Return a unique filename in C:\ that
// begins with "b" and has the extension "sys"
var str = Application.UniqueFileName("C:", "b", "sys");
Application.Alert(str);
```

UnregisterCRCL
This method is reserved for internal use only.

URLToPath
Returns a URL. Use `DocumentHost.URLToPath` when creating a customization for use in both XMetaL Author and XMetaL XMAX.

Applies to

XMetaL Author

Returns

String

Usage in JScript

```
var myString = Application.URLToPath("strURL" [,"strBase"]);
```

If only `strURL` is specified, returns a path corresponding to the `strURL`. `strBase` is an optional parameter specifying a path to a document. If `strURL` is a relative URL, and `strBase` is specified, returns a full path corresponding to `strURL`, using `strBase` as the base path.

Usage in VBScript

```
dim myString = Application.URLToPath("strURL" [,"strBase"])
```

Example

```
// XMetaL Script Language JSCRIPT:
var p1, p2;
p1=Application.URLToPath("file:///c|/data/index.html");
// Displays "c:\data\index.html"
Application.Alert(p1);
p2=Application.URLToPath("../images/one.gif",
"d:\\dir1\\dir2\\test.htm");
// Displays "d:\dir1\images\one.gif"
Application.Alert(p2);
```

WritableDirExists
Indicates whether the folder specified exists and is writable.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
var myBoolean = Application.WritableDirExists("strPath");
```

Usage in VBScript

```
dim myBoolean = Application.WritableDirExists("strPath")
```

Example

```
// XMetaL Script Language JSCRIPT:
var exists;
var appPath = Application.Path;
exists = Application.WritableDirExists(appPath);
```

WritableFileExists

Indicates whether the file specified exists and is writable.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
var myBoolean = Application.WritableFileExists("strPath");
```

Usage in VBScript

```
dim myBoolean = Application.WritableFileExists("strPath")
```

Example

```
// XMetaL Script Language JSCRIPT:
var docPath = ActiveDocument.FullName;
Application.Alert(Application.WritableFileExists(docPath));
```

Assets

The Assets interface enables some configurations of the Resource Manager Assets tab. To obtain an Assets object, use the `ResourceManager.Assets` property. This interface is available only when scripting for XMetaL Author.

Assets API's are unsupported. The API or feature may be removed in a future release and usage should be avoided.

Properties

WebBrowser (unsupported)

Returns the IDispatch object of the Web browser control in the lower (Contents) pane of Assets tab. This object gives you access to the Web browser control's properties and methods.

Applies to

XMetaL Author

Access

Read only

Returns

IDispatch object

Usage in JScript

```
Assets_object.WebBrowser;
```

Usage in VBScript

```
Assets_object.WebBrowser
```

Example

```
// XMetaL Script Language JSCRIPT:
browser = ResourceManager.Assets.WebBrowser;
// Use a browser-specific method
browser.Navigate("www.xmetal.com");
```

Methods

Refresh (unsupported)

Refreshes the upper and lower panes of the Assets tab.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Assets_object.Refresh();
```

Usage in VBScript

```
Assets_object.Refresh
```

Example

```
// XMetaL Script Language JSCRIPT:
ResourceManager.Assets.Refresh();
```

SelectFolder (unsupported)

Selects the folder specified in the upper (Explorer) pane of the Resource Manager Assets tab.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Assets_object.SelectFolder("strPath");
```

`strPath` must be an absolute (full) path.

Usage in VBScript

```
Assets_object.SelectFolder "strPath"
```

Example

```
// XMetaL Script Language JSCRIPT:
var resman = ResourceManager.Assets;
astFldr = "\\Journalist DTD\\Calendar Builder";
resman.SelectFolder(Application.AssetsPath + astFldr);
```

AttributeInspector

Enables developers to intercept changes that users make in the Attribute Inspector and apply business logic to them. Developers can inspect values before and after the changes, and can cancel changes if necessary.

```
// XMetaL Script Language JSCRIPT:
// Add this code to On_Before_Set_Attribute_From_AI event macro
// Make sure a user's change does not go against business rules
var ai = Application.AttributeInspector;
var element = ai.Element;
var attrName = ai.AttributeName;
var beforeValue = ai.AttributeValueBeforeChange;
var afterValue = ai.AttributeValueAfterChange;
if (attrName == "Color") {
if (afterValue == "Blue") {
// blue is not a valid color
s1 = "Blue is not an accepted color. Would you like to choose Green instead?";
response = Application.MessageBox(s1, 32+4);
if (response == 6) { // 6 is "Yes"
ai.AttributeValueAfterChange = "Green";
} else {
// Alert user again that business rules are being violated
s1 = "Do you really want to change " + beforeValue + " to Blue?";
response = Application.MessageBox(s1, 32+4+256);
```



```
if (response == 7) { // 7 is "No"
ai.CancelChange();
} else {
// Let the change happen, but alert the user. s1 = "Business rules overridden. Blue has
been accepted.";
Application.MessageBox(s1, 32+4);
}
}
}
}
```

Properties

AttributeName

The name of the attribute that is being changed. It is valid only in the On_Before/After_Set_Attribute_From_AI event macros. The name is the qualified name in the form prefix:local name. For documents with no namespace, the prefix and the colon are omitted.

Applies to

XMetaL Author

Access

Read only

Returns

BSTR *prop

Usage in JScript

```
Application.AttributeInspector.AttributeName;
```

Usage in VBScript

```
Application.AttributeInspector.AttributeName
```

AttributeValueAfterChange

The attribute value after the change. This property is readable during the On_Before/After_Set_Attribute_From_AI event macros, and writable only in the On_Before_Set_Attribute_From_AI event macro.

Applies to

XMetaL Author

Access

Read/write

Returns

BSTR *prop

Usage in JScript

```
Application.AttributeInspector.AttributeValueAfterChange;
```

Usage in VBScript

```
Application.AttributeInspector.AttributeValueAfterChange
```

AttributeValueBeforeChange

The attribute value before the change. This property is valid only during the On_Before/After_Set_Attribute_From_AI event macros.

Applies to

XMetaL Author

Access

Read only

Returns

BSTR *prop

Usage in JScript

```
Application.AttributeInspector.AttributeValueBeforeChange;
```

Usage in VBScript

```
Application.AttributeInspector.AttributeValueBeforeChange
```

Element

The DOMElement whose attribute is being changed. This property is valid only in the On_Before/After_Set_Attribute_From_AI event macros.

Applies to

XMetaL Author

Access

Read/write

Returns

DOMElement **prop

Usage in JScript

```
Application.AttributeInspector.Element;
```

Usage in VBScript

```
Application.AttributeInspector.Element
```


Visible
Determines whether or not the Attribute Inspector is displayed (in view).

Applies to

XMetaL Author

Access

Read only

Returns

VARIANT_BOOL* pVal

Usage in JScript

```
Application.AttributeInspector.Visible;
```

Usage in VBScript

```
Application.AttributeInspector.Visible
```

Methods

CancelChange
Cancels the current change in the Attribute Inspector. It can be called only in the On_Before_Set_Attribute_From_AI event macro.

Applies to

XMetaL Author

Usage in JScript

```
Application.AttributeInspector.CancelChange()
```

Usage in VBScript

```
Application.AttributeInspector.CancelChange()
```

CanElement

Provides detailed information about a specific item in a CanElementList.

Properties

description
Returns the description of a CanElement. Use this property instead of `ElementListItem.Description` when scripting for XMetaL XMAX.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
CanElement_object.description;
```

Usage in VBScript

```
CanElement_object.description
```

Example

```
// XMetaL Script Language JSCRIPT:
// Displays all elements (with their descriptions) that can
// be inserted into the current selection

// pre: myXMControl has previously been defined as an XMetaL XMAX object


function ShowCanInsert() {
var sel = myXMControl.Selection;
var list = sel.CanInsertList;
var num = list.count;
var str = "";

for (i=0; i<num; i++) {
str += list.item(i).name;
str += "-";
str += list.item(i).description;
str += "\n";
}
ActiveDocument.Host.Alert(str);

}
```

DisplayName

Returns the short description of the element type as configured in the CTM file. Use this property instead of `ElementListItem.DisplayName` when scripting for XMetaL XMAX.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
CanElement_object.DisplayName;
```

Usage in VBScript

```
CanElement_object.DisplayName
```

Example

```
// XMetaL Script Language JSCRIPT:
// Displays all elements using their nice-names (and the number of times
// each element appears in the active document) that can
// be inserted into the current selection

// pre: myXMControl has previously been defined as an XMetaL XMAX object

function showCanInsert() {
var sel = myXMControl.Selection;
var list = sel.CanInsertList;
var num = list.count;
var str = "";

for (i=0; i<num; i++){
str += list.item(i).DisplayName;
str += "is used ";
str += list.item(i).UsageCount;
str += " times in the active document.\n";
}
ActiveDocument.Host.Alert(str);

}
```

name
Returns the name of a CanElement. Use this property instead of `ElementListItem.Name` when scripting for XMetaL XMAX.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
CanElement_object.name;
```

Usage in VBScript

```
CanElement_object.name
```

Example

```
// XMetaL Script Language JSCRIPT:
// Displays all elements (and the number of times
// each element appears in the active document) that can
// be inserted into the current selection

// pre: myXMControl has previously been defined as an XMetaL XMAX object

function showCanInsert() {
var sel = myXMControl.Selection;
var list = sel.CanInsertList;
var num = list.count;
var str = "";

for (i=0; i<num; i++){
str += list.item(i).name;
str += "is used ";
str += list.item(i).UsageCount;
str += " times in the active document.\n";
}
ActiveDocument.Host.Alert(str);

}
```

Required

Returns True if the CanElement is required to be inserted into the selection, and False otherwise. Use this property instead of `ElementListItem.Required` when scripting for XMetaL XMAX.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
CanElement_object.Required;
```

Usage in VBScript

```
CanElement_object.Required
```

Example

```
// XMetaL Script Language JSCRIPT:
// Displays all elements (and whether or not each
// element is required) that can
// be inserted into the current selection

// pre: myXMControl has previously been defined as an XMetaL XMAX object

function ShowCanInsert() {
var sel = myXMControl.Selection;
var list = sel.CanInsertList;
```

```
var num = list.count;
var str = "";

for (i=0; i<num; i++) {
str+= list.Item(i).Name;

if (list.Item(i).Required)
str += "- Required";
else
str += "- Not required";

str += "\n";
}
ActiveDocument.Host.Alert(str);

}
```

UsageCount
Returns a long integer representing the number of times CanElement appears in the active document. Use this property instead of `ElementListItem.UsageCount` when scripting for XMetaL XMAX.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
CanElement_object.UsageCount;
```

Usage in VBScript

```
CanElement_object.UsageCount
```

Example

```
// XMetaL Script Language JSCRIPT:
// Displays all elements (and the number of times
// each element appears in the active document) that can
// be inserted into the current selection

// pre: myXMControl has previously been defined as an XMetaL XMAX object

function showCanInsert() {
var sel = myXMControl.Selection;
var list = sel.CanInsertList;
var num = list.count;
var str = "";

for (i=0; i<num; i++){
str += list.item(i).Name;
str += "is used ";
str += list.item(i).UsageCount;
```

```
str += " times in the active document.\n";
}
ActiveDocument.Host.Alert(str);

}
```

CanElementList

The CanElementList interface is a collection of CanElement objects.

You can acquire the interface from one of the following properties: `Selection.CanChangeList` or `Selection.CanInsertList`. CanElementList represents a list of elements that you could validly use to update the current selection. You can change the current selection into any element listed in `Selection.CanChangeList`. You can insert any element listed in `Selection.CanInsertList` into the current selection.

Use the CanElementList interface instead of the ElementList interface when scripting for XMetaL XMAX.

Properties

Count

Returns the number of elements in the CanElementList. Use this property instead of `ElementList.Count` when scripting for XMetaL XMAX.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
Selection.CanChangeList.Count;
```

Usage in VBScript

```
Selection.CanChangeList.Count
```

Example

```
// XMetaL Script Language JSCRIPT:
// display a list of all items that you can
// insert into the current selection

// pre: myXMControl has previously been defined as an XMetaL XMAX object

function showCanInsert() {
```



```
var sel = myXMControl.Selection;
var list = sel.CanInsertList;
var num = list.Count;
var str = "";

for (i=0; i<num; i++) {
str+= list.item(i).name;
str += "\n";
}
ActiveDocument.Host.Alert(str);

}
```

item
Returns the CanElement located at the specified position in the CanElementList. Use this method instead of `ElementList.item` when scripting for XMetaL XMAX.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

CanElement object

Usage in JScript

```
var myItem = Selection.CanChangeList.item(0);
```

`intIndex` is an integer between 0 and the value represented by `CanElementList.count-1`. If you pass an integer outside this range to `item`, it will return null. It will also return null if the CanElement located at `intIndex` is null.

Usage in VBScript

```
set myItem = Selection.CanChangeList.item(0)
```

Example

```
// XMetaL Script Language JSCRIPT:
// display a list of all items that you can
// insert into the current selection

// pre: myXMControl has previously been defined as an XMetaL XMAX object

function showCanInsert() {
var sel = myXMControl.Selection;
var list = sel.CanInsertList;
var num = list.Count;
var str = "";

for (i=0; i<num; i++) {
str+= list.item(i).name;
str += "\n";
}
```

```
ActiveDocument.Host.Alert(str);
}
```

CheckData

The CheckData interface enables events and event macros to examine the data being validated at the time of validation.

The `Application.CheckData` property, available only in XMetaL Author, returns a CheckData object. Its properties are intended for use in On_Check_Element_SimpleContent and On_Check_Attribute_Value event macros.

In XMetaL XMAX, the CheckData object is provided as a parameter of the OnCheckElementSimpleContent and OnCheckAttributeValue events.

Properties

AttributeName
Contains the name of the attribute that validation of the value is being performed on. Use only in OnCheckAttributeValue or On_Check_Attribute_Value.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
CheckData_object.AttributeName;
```

Usage in VBScript

```
CheckData_object.AttributeName
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put the following code
// in the On_Check_Attribute_Value macro
// This code will only work in XMetaL Author

var cd = Application.CheckData;
var elem = cd.Element;
var name = elem.tagName;
var attrName = cd.AttributeName;
```



```
if((name == "InlineGraphic" || name == "Graphic")
&& attrName == "FileRef") {
var value = new String(cd.Value);
var msg = null;

if(str.indexOf(" ")>=0) {
msg = "URL cannot have whitespace:"
}

else if (str.indexOf("\\")>= 0) {
msg = "URL cannot have backslash:"
}

if(msg) {
msg += cd.AttributeName;
msg += "='";
msg += value;
msg += "'";
cd.ValidationMsg = msg;
}
}
```

Element
Returns the DOMElement object for the content or attribute value that is being validated.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMElement object

Usage in JScript

```
CheckData_object.Element;
```

Usage in VBScript

```
CheckData_object.Element
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put the following code
// in the On_Check_Element_SimpleContent macro
// This code will only work in XMetaL Author

var cd = Application.CheckData;
var elem = cd.Element;
var name = elem.tagName;

// Check value..... if(name == "Email") {
var value = cd.Value;

if (value != null) {
```

```
var re = new RegExp("^[^@ ]+@[^@ ]+$");

if (re.exec(value) == null) {
var msg = "Invalid email address value:"
msg += "'";
msg += value;
msg += "'";
cd.ValidationMsg = msg;
}
}
}
```

ValidationMsg
Contains the error message provided by the system's simple type validation. If the system does not find an error, this property returns an empty string.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
CheckData_object.ValidationMsg;
```

Usage in VBScript

```
CheckData_object.ValidationMsg
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put the following code
// in the On_Check_Element_SimpleContent macro
// This code will only work in XMetaL Author

var cd = Application.CheckData;
var elem = cd.Element;
var name = elem.tagName;

// Check value..... if(name == "Email") {
var value = cd.Value;

if (value != null) {
var re = new RegExp("^[^@ ]+@[^@ ]+$");

if (re.exec(value) == null) {
var msg = "Invalid email address value:"
msg += "'";
msg += value;
msg += "'";
cd.ValidationMsg = msg;
}
```



```
}
}
```

Value
Represents the value being validated.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
CheckData_object.Value;
```

Usage in VBScript

```
CheckData_object.Value
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put the following code
// in the On_Check_Element_SimpleContent macro
// This code is applicable only to XMetaL Author

var cd = Application.CheckData;
var elem = cd.Element;
var name = elem.tagName;

// Check value..... if(name == "Email") {
var value = cd.Value;

if (value != null) {
var re = new RegExp("^[^@ ]+@[^@ ]+$");

if (re.exec(value) == null) {
var msg = "Invalid email address value:"
msg += "'";
msg += value;
msg += "'";
cd.ValidationMsg = msg;
}
}
}
```

Clipboard

You can use the Clipboard interface to check the contents and modify the contents of the standard Windows clipboard. A clipboard object can contain text, files, other objects, or any other item that can be held in the Windows clipboard. This interface is available only when scripting for XMetaL Author.

Properties

DataObject

Expresses the standard IUnknown interface. Same as `Application.DropDataObject`.

Applies to

XMetaL Author

Access

Read only

Returns

IUnknown pointer

Usage in JScript

```
var myClipboard = Application.Clipboard.DataObject;
```

Usage in VBScript

```
dim myClipboard = Application.Clipboard.DataObject
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Get the clipboard object
Set co = Application.Clipboard.DataObject
```

HasFile

Returns True if the content on the clipboard has one or more files.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
var myBoolean = Application.Clipboard.HasFile;
```

Usage in VBScript

```
dim myBoolean = Application.Clipboard.HasFile
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Find out if there are any files on the clipboard
' and notify the user if there are. Dim clipfiles
clipfiles = Application.Clipboard.HasFile
If clipfiles = True Then
MsgBox ("There is at least one file on the clipboard.")
End If
```

HasFormat

Returns True if the format specified is on the clipboard.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
var myBoolean = Application.Clipboard.HasFormat("strFormat");
```

Usage in VBScript

```
dim myBoolean = Application.Clipboard.HasFormat(strFormat)
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Find out if the format strFormat is on the clipboard
Dim f
f = Application.Clipboard.HasFormat(strFormat)

If f = True Then
MsgBox ("This format is on the clipboard.")
End If
```

HasText

Returns True if the content on the clipboard is text.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
var myBoolean = Application.Clipboard.HasText;
```

Usage in VBScript

```
dim myBoolean = Application.Clipboard.HasText
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Find out if the clipboard contains text
' and notify the user if it does. Dim istext istext =
Application.Clipboard.HasText

If istext = True Then
MsgBox ("There is text on the clipboard.")
End If
```

Text

When reading, the text on the clipboard is returned. When setting the text on the clipboard, no value is returned.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var myString = Application.Clipboard.Text;
Application.Clipboard.Text = myString;
```

Usage in VBScript

```
dim myString = Application.Clipboard.Text
Application.Clipboard.Text = myString
```


Example

```
' XMetaL Script Language VBSCRIPT:
' Put some text on the clipboard
Application.Clipboard.Text = "This came from the clipboard."

' Now read it back
MsgBox (Application.Clipboard.Text)
```

Methods

FirstFormatName

Returns the name of the first available format on the clipboard. If there is an object on the clipboard, you can retrieve its type using this method.

Applies To

XMetaL Author

Returns

String

Usage in JScript

```
var myString =
 Application.Clipboard.FirstFormatName();
```

Usage in VBScript

```
dim MyString = Application.Clipboard.FirstFormatName
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Get the name of the data type for the first object
' on the clipboard ... try it with text, graphics and
' other types on the clipboard
name = Application.Clipboard.FirstFormatName
name2 = Application.Clipboard.NextFormatName
MsgBox ("First type = " & name)
MsgBox ("Next type = " & name2)
```

NextFormatName

Returns the name of the next format on the clipboard. If there is an object on the clipboard, you can retrieve its type using this method.

Applies To

XMetaL Author

Returns

String

Usage in JScript

```
var myString = Application.Clipboard.NextFormatName();
```

Usage in VBScript

```
myString = Application.Clipboard.NextFormatName
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Get the name of the data type for the first object
' on the clipboard ... try it with text, graphics and
' other types on the clipboard
name = Application.Clipboard.FirstFormatName
name2 = Application.Clipboard.NextFormatName
MsgBox ("First type = " & name)
MsgBox ("Next type = " & name2)
```

SetEmpty
Empties the contents of the clipboard.

Applies To

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.Clipboard.SetEmpty();
```

Usage in VBScript

```
Application.Clipboard.SetEmpty
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Clear the clipboard if it contains text

If Application.Clipboard.HasText = True then
Application.Clipboard.SetEmpty
MsgBox "Text has been cleared from the clipboard."
End If
```

Customizations

Using the customization file editor, you can configure XMetaL to treat selected elements in the current DTD or schema as paragraphs, toggling elements, image elements, or list elements. The Customizations interface stores information about elements with a modified (Treat As Type) property in the customization file for the DTD or schema.

You can also rank the elements of each type. You can then use the Customizations interface to read these properties programmatically. All properties returned using this interface are read only. There are no methods available.

To get the Customizations interface, use `Document.Customizations`.

Properties

Images

Returns the TreatAsImages object for a document. This object stores information about the elements treated as images in a document's customization.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

TreatAsImages object

Usage in JScript

```
var imagesObject = Document.Customizations.Images;
```

Usage in VBScript

```
set imagesObject = Document.Customizations.Images
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Images
num2 = objs.Count
MsgBox("Number of images is " & cstr(num2))

For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.Item(cnt1)
MsgBox("Name = " & obj1.Name & " source = " & obj1.Source & " height = "
&obj1.Height & " width = " & obj1.Width & " AltText = " & obj1.AltText)
Next
```

Links

Returns the TreatAsLinks object for a document. This object stores information about the elements treated as links in a document's customization.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

TreatAsLinks object

Usage in JScript

```
var linksObject = Document.Customizations.Links;
```

Usage in VBScript

```
set linksObject = Document.Customizations.Links
```

Example

```
var linksCollection = Document.Customizations.Links;
if (linksCollection.Count > 0)
   linkObject = linksCollection.Item(0);
```

Lists

Returns the TreatAsLists object for a document. This object stores information about the elements treated as lists in a document's customization.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

TreatAsLists object

Usage in JScript

```
var listsObject = Document.Customizations.Lists;
```

Usage in VBScript

```
set listsObject = Document.Customizations.Lists
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Lists
num2 = objs.count
MsgBox("Number of lists is " & cstr(num2))

For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)
```



```
if (obj1.type = 2) Then
MsgBox("Type = " & cstr(obj1.type) & " Name = " & obj1.name & " Term = "
 & obj1.Term & " Definition = " & obj1.Definition)
Else
MsgBox("Type = " & cstr(obj1.type) & " Name = " & obj1.name & " ListHeader
 = " & obj1.ListHeader & " ListItem = " & obj1.ListItem)
End If
Next
```

Paragraphs
Returns the TreatAsParagraphs object for a document. This object stores information about the elements treated as paragraphs in a document's customization.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

TreatAsParagraphs object

Usage in JScript

```
var paragraphsObject = Document.Customizations.Paragraphs;
```

Usage in VBScript

```
set paragraphsObject = Document.Customizations.Paragraphs
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Paragraphs

num2 = objs.count
MsgBox("Number of paragraphs is " & cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)
MsgBox("Name = " & obj1.name & " Parent = """ & obj1.Parent & """")
Next
```

TogglingElements
Returns the TogglingElements object for a document. This object stores information about the elements treated as toggling elements in a document's customization.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

TogglingElements object

Usage in JScript

```
obj = Document.Customizations.TogglingElements;
```

Usage in VBScript

```
set obj = Document.Customizations.TogglingElements
```

Example

```
'XMetaL Script Language VBScript:
'displays all of the toggling elements in a document's customization

Set ctm = ActiveDocument.Customizations
Set objs = ctm.TogglingElements

MsgBox("Bold element is " & objs.Bold)
MsgBox("Italic element is " & objs.Italic)
MsgBox("Underline element is " & objs.Underline)

num2 = objs.Count
MsgBox("Number of toggling elements with macros is " & cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.Item(cnt1)
MsgBox("Name = " & obj1.name & " MacroName = " & obj1.MacroName)
Next
```

DBImport

You can import data from a database using the Import Table dialog. This dialog is implemented using a macro. You can update data you have imported also using a macro.

Using the Import Database dialog, you can create a table in your document based on the contents of a database or spreadsheet file. When you create a query, the parameters are saved to a file that can be re-used to refresh the data.

To allow users access to this functionality, you must add macros to your customization. For an example implementation, see the Import Table and Update Table macros in the sample macro file `journalist.mcr`. These macros may be customized for your DTD and local file system and attached to menus or toolbars.

The Import Database dialog box is implemented by the DBImport interface.

In order to use the Import Database dialog, the following components must be installed *before* you install XMetaL:

- Microsoft Data Access Components (MDAC). If you do not have MDAC installed when you install XMetaL, the XMetaL installer offers to install it.
- Windows Scripting Engine. This is installed when you install Internet Explorer.

Methods

NewDBImport

Displays the XMetaL Import Database dialog.

Returns

Long

Usage in JScript

```
DBImport_object.NewDBImport("strParamFile", "strOutputFile", [boolIsSGML]);
```

XMetaL saves the details of the query in `strParamFile`. The output (an HTML, CALS, or generic XML table) is saved in `strOutputFile`. If `boolIsSGML` is True, the table markup is generated in SGML (this affects only the COLSPEC element of CALS tables). Returns 0 if the user clicks **OK** to close the Import Database dialog, and 1 otherwise.

Usage in VBScript

```
DBImport_object.NewDBImport("strParamFile", "strOutputFile", [boolIsSGML])
```

Example

```
// XMetaL Script Language JSCRIPT:
// Show the dialog box
var dbi = new ActiveXObject("SoftQuad.DBImport");
var ok = dbi.NewDBImport("paramfile.txt", "outputfile.txt");
if (ok) {
//...code to insert the table in the document... }
```

UpdateDBImport

Updates the current table using the XMetaL Import Database dialog box.

Returns

Long

Usage in JScript

```
DBImport_object.UpdateDBImport("strParamFile", "strOutputFile");
```

The output (an HTML, CALS, or generic XML table) is saved in `strOutputFile`. Returns 0 if the user clicks **OK** to close the Import Database dialog box, and 1 otherwise.

Usage in VBScript

```
DBImport_object.UpdateDBImport("strParamFile", "strOutputFile")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Show the dialog box
var dbi = new ActiveXObject("SoftQuad.DBImport");
```

```
var ok = dbi.UpdateDBImport(paramFile, tableFile);
if (ok) {
//...code to insert the table in the document... }
```

Document

The Document object represents an open XMetaL document. The special ActiveDocument always represents the active XMetaL document, that is, the document displayed in the active window. This object is globally available to all scripts.

In XMetaL Author, the Documents object represents all open XMetaL documents. A script intended for use with XMetaL Author can use several Document objects, and perform operations on the documents represented by each one. Note that modifying a document in a script does not necessarily make it the active document.

In XMetaL XMAX, only one document is open at a time. A script intended for use with XMetaL XMAX can use only one Document object, ActiveDocument. This object can also be retrieved through the top-level XMetaL XMAX Document interface.

The Document object inherits the DOMNode interface.

The Document interface is identical to the DOMDocument interface. Although the Document interface is defined by the DOM, it is grouped with the general interfaces because XMetaL's implementation includes many extra features.

Properties

The Document interface supports the following properties, in addition to those inherited from the DOMNode interface.

ActiveInPlaceControl

Used in the OnShouldCreate, OnInitialize, OnFocus and OnSynchronize macros to obtain the IDispatch object for the in-place ActiveX control (if there is one) associated with the current element.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

InPlaceControl (IDispatch) object

Usage in JScript

```
ActiveDocument.ActiveInPlaceControl;
```

Usage in VBScript

```
ActiveDocument.ActiveInPlaceControl
```


Example

```
// XMetaL Script Language JSCRIPT:

// Control is the IsoView CGM viewer in this example
var aipc = ActiveDocument.ActiveInPlaceControl;
var isoview = aipc.Control;
// Filename is a control-specific property!
isoview.Filename = "c:\\Images\\default.cgm";
```

AllUsersMacroFile
Returns the full path to the macro file available to all users for the DTD or schema of the current document (that is, the DTD- or schema-specific macro file stored in the XMetaL macro folder).

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Document_object.AllUsersMacroFile;
```

Usage in VBScript

```
Document_object.AllUsersMacroFile
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.Host.Alert(ActiveDocument.AllUsersMacroFile);
```

AlwaysUndoClearAfterSave
Clears the undo stack whenever `Document.Save` is called.

Applies to

XMetaL XMAX and XMetaL Author

Access

Returns

Usage in JScript

Usage in VBScript

Example

AddCompareFunc
This property is reserved for internal use.

AutoScaleImageHeight
This property is reserved for internal use only.

AutoScaleImageWidth
This property is reserved for internal use only.

AutoSetImageWidthHeight
This property is reserved for internal use only.

BackgroundSpellchecking
Returns True if the document is being spell-checked in the background.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/Write

Returns

Boolean

Usage in JScript

```
Document_object.BackgroundSpellchecking = false;
```

Usage in VBScript

```
Document_object.BackgroundSpellchecking
```

Example

```
// XMetaL Script Language JSCRIPT:

if (ActiveDocument.BackgroundSpellchecking) {
ActiveDocument.Host.Alert("This document is being actively spellchecked.");
}
```

BodyAttribute
This property is unsupported.

BrowserApplication

Returns or sets the path of the browser application used to preview the document. This property is used to send command-line parameters to the browser application. If the built-in Page Preview view is used, this property's value is the null string. This property is intended to be used in the On_Before_Document_Preview event macro, and has a null value until that macro has been called for the first time in the current session.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var strBrowserApp = Document_object.BrowserApplication;
Document_object.BrowserApplication = "strPath";
```

Usage in VBScript

```
dim strBrowserApp = Document_object.BrowserApplication
Document_object.BrowserApplication = "strPath";
```

Example

```
// XMetaL Script Language JSCRIPT:
// Place this code in the On_Before_Document_Preview event macro
// If the Browser Application path is not set, set it to a default path

if (ActiveDocument.BrowserApplication == null) {
ActiveDocument.BrowserApplication = "C:\\Program Files\\Internet Explorer";
}
```

BrowserURL

Returns or sets the URL of the document being previewed in the built-in Page Preview view or with an external application. This URL may point to a temporary file. This property is intended to be used in the On_Before_Document_Preview special macro, and has a null value until that macro has been called for the first time in the current session.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var strBrowserURL = Document_object.BrowserURL;
Document_object.BrowserURL = "strURL";
```

Usage in VBScript

```
vbl = Document_object.BrowserURL
Document_object.BrowserURL = "strURL";
```

Example

```
// Copy into On_Before_Document_Preview event macro
// XMetaL Script Language JSCRIPT:
var defaultdoc = "C:\\Data\Default.xml";
ActiveDocument.BrowserURL = defaultdoc;
```

ChangedNodes

When certain types of nodes are changed, one or more of the following flags are set: ContentInserted, ContentDeleted, AttributesChanged, and NewNode. This mechanism applies to the following nodes: DOMElement, DOMComment, DOMCDATASection, DOMProcessingInstruction, DOMEntityReference, DOMCharacterReference. This property returns a DOMNodeList of nodes in the document for which one or more of these flags have been set. ‘Changed’ flags can be turned off using ClearNodeChangedStates and ClearAllChangedStates.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNodeList object

Usage in JScript

```
var changedNodeList = Document_object.ChangedNodes;
```

Usage in VBScript

```
dim changedNodeList = Document_object.ChangedNodes
```

Example

```
// XMetaL Script Language JSCRIPT:

var chgdNodes = ActiveDocument.ChangedNodes;

if (chgdNodes.length==0) {
ActiveDocument.Host.Alert("No changed nodes.");
} else {
ActiveDocument.Host.Alert(chgdNodes.length + " changed nodes.");
}
```


ChangedNodesbyKey

When certain types of nodes are changed, one or more of the following flags are set: ContentInserted, ContentDeleted, AttributesChanged, and NewNode. This mechanism applies to the following nodes: DOMElementDOMComment, DOMCDATASection, DOMProcessingInstruction, DOMEntityReference, DOMCharacterReference. This property returns a DOMNodeList of nodes tracked in all open documents as the user edits them. ‘Changed’ flags can be turned off using ClearNodeChangedStatesbyKey and ClearAllChangedStatesbyKey.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNodeList object

Usage in JScript

```
var changedNodesByKeyList = Document_object.ChangedNodesbyKey;
```

Usage in VBScript

```
dim changedNodesByKeyList = Document_object.ChangedNodesbyKey
```

Example

```
// XMetaL Script Language JSCRIPT:
// if scripting for XMetaL Author,
// put 'var mykey ="abc"' into On_Application_Open,
// and put the following in a local macro

// if scripting for XMetaL XMAX, define var mykey = "abc"
// elsewhere in the program, then run the following code
// in a macro or script

if (ActiveDocument.ViewType==2){
ActiveDocument.ViewType=1;
}
var nds = ActiveDocument.ChangedNodesbyKey(mykey);
var l = nds.length;
var changenode = null;
var name = null;
var f = 0;
ActiveDocument.Host.Alert(l);

if (l==0){
ActiveDocument.Host.Alert("The method works.");
} else {

for (var i=0; i<=l;i++){
f++;
changenode=nds.item(i);
name = changenode.nodeName;
ActiveDocument.Host.Alert(name
+ "\n\nNewNode:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey, "NewNode",
 changenode)
+ "\nContentInserted:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
```

```
"ContentInserted", changenode)
+ "\nContentDeleted:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"ContentDeleted", changenode)
+ "\nAttributesChanged:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"AttributesChanged", changenode));

if (f==nds.length){
break;
}
}
}

//Uncomment these methods one at a time running

//Application.RemoveChangedNodeKey(mykey);
//ActiveDocument.ClearAllChangedStatesbyKey(mykey);
//ActiveDocument.ClearNodeChangedStates(mykey, changenode, true);
//ActiveDocument.ClearNodeChangedStatesbyKey(mykey,changenode,false);
```

CheckData
Returns the CheckData object. It is meant for use in the following events and event macros: On_Check_Element_SimpleContent, On_Check_Attribute_Value, OnCheckElementSimpleContent (XMetaL XMAX only), OnCheckAttributeValue (XMetaL XMAX only). Do not change the DOMNode during the execution of either of the event macros; non-recoverable errors may result.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

CheckData object

Usage in JScript

```
var myCheckData = ActiveDocument.CheckData;
```

Usage in VBScript

```
dim myCheckData = ActiveDocument.CheckData
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put the following code
// in the On_Check_Element_SimpleContent macro

var cd = ActiveDocument.CheckData;
var elem = cd.Element;
var name = elem.tagName;

// Check value.....

if(name == "Email") {
var value = cd.Value;
```



```
if (value != null) {
var re = new RegExp("^[^@ ]+@[^@ ]+$");

if (re.exec(value) == null) {
var msg = "Invalid email address value:"
msg += "'";
msg += value;
msg += "'";
cd.ValidationMsg = msg;
}
}
}


// Put the following code
// in the On_Check_Attribute_Value macro

var cd = ActiveDocument.CheckData;
var elem = cd.Element;
var name = elem.tagName;
var attrName = cd.AttributeName;

if((name == "InlineGraphic" || name == "Graphic")
&& attrName == "FileRef") {
var value = new String(cd.Value);
var msg = null;

if(str.indexOf(" ")>=0) {
msg = "URL cannot have whitespace:"
}

else if (str.indexOf("\\")>= 0) {
msg = "URL cannot have backslash:"
}

if(msg) {
msg += cd.AttributeName;
msg += "='";
msg += value;
msg += "'";
cd.ValidationMsg = msg;
}
}
```

Clipboard
Returns the Clipboard object.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Clipboard object

Usage in JScript

```
var cobj = ActiveDocument.Clipboard;
```

Usage in VBScript

```
set cobj = ActiveDocument.Clipboard
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Get the clipboard object
Set co = ActiveDocument.Clipboard
```

CRLocator

This property is reserved for internal use only.

CurrentCSS

Default CSS path property for all views.

Applies to

XMetaL Author and XMAX

Access

Read only

Usage in JScript

```
ActiveDocument.CurrentCSS;
```

Usage in VBScript

```
ActiveDocument.CurrentCSS
```

Example

```
//XMetaL Script Language JSCRIPT:
//Using the CamerasInFocus sample in XMetaL Author,
//copy and paste this script example to change css style.

var style = ActiveDocument.CurrentCSS;
Application.Alert("Active Document css:\n" + ActiveDocument.CurrentCSS);
var newStyle = Application.Path +
"\\Display\\journalist_5.css";

ActiveDocument.CurrentCSS = newStyle;
Application.Alert("Active Document css:\n" +
ActiveDocument.CurrentCSS);

ActiveDocument.CurrentCSS = style;
Application.Alert("Active Document css:\n" +
ActiveDocument.CurrentCSS);
```

CurrentUserMacroFile

Returns the full path to the current user's macro file for the DTD or schema of the current document (that is, the DTD- or schema-specific macro file stored in the current user's personal settings folder). If you are scripting for XMetaL XMAX, use `Document.AllUsersMacroFile` instead.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Document_object.CurrentUserMacroFile;
```

Usage in VBScript

```
Document_object.CurrentUserMacroFile
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.Host.Alert(ActiveDocument.CurrentUserMacroFile);
```

CustomDocumentProperties

Returns a DocumentProperties object containing the custom properties for the document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DocumentProperties object

Usage in JScript

```
Document_object.CustomDocumentProperties;
```

Usage in VBScript

```
Document_object.CustomDocumentProperties
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the number of custom properties

var docProps;
```

```
docProps = ActiveDocument.CustomDocumentProperties;
ActiveDocument.Host.Alert(docProps.Count);
```

Customizations
Returns a Customizations object containing the various properties set in the customization file editor in XMetaL Developer.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Customizations object

Usage in JScript

```
Document_object.Customizations;
```

Usage in VBScript

```
Document_object.Customizations
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Paragraphs

num2 = objs.count
MsgBox("Number of elements treated as paragraphs is " + cstr(num2))
```

CustomProperties
Returns a NameVariantProperties object containing the custom properties for the document.

Applies to

XMetaL Author

Access

Read only

Returns

NameVariantProperties object

Usage in JScript

```
Document_object.CustomProperties;
```


Usage in VBScript

```
Document_object.CustomProperties
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the number of custom properties
var docProps;
docProps=ActiveDocument.CustomProperties;
ActiveDocument.Host.Alert(docProps.Count);
```

DataObjectAsText

Forces the object on the clipboard to be treated as text.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
var myString = ActiveDocument.DataObjectAsText("strFormatName", objData);
```

`strFormatName` indicates the format to apply to the object when creating the text equivalent. `objData` represents the object to treat as text.

Usage in VBScript

```
dim myString = ActiveDocument.DataObjectAsText("strFormatName", objData)
```

Example

```
// XMetaL Script Language JSCRIPT:
// treat the object on the clipboard as text
var data = ActiveDocument.Clipboard.DataObject;
var txt = ActiveDocument.DataObjectAsText("ASCII", data);
Application.Alert (txt);
```

DisplayStylesFile

Returns the full path to the document's display styles file. This is the styles file that XMetaL uses to format the document in Normal or Tags On view; it is not necessarily the styles file that is linked to a document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Document_object.DisplayStylesFile;
```

Usage in VBScript

```
Document_object.DisplayStylesFile
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.Host.Alert(ActiveDocument.DisplayStylesFile);
```

doctype

Returns a DOMDocumentType object, which contains information about the DTD or schema and document type declaration of the document. Returns a null string if the document has no document type declaration.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMDocumentType object

Usage in JScript

```
Document_object.doctype;
```

Usage in VBScript

```
Document_object.doctype
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the document type name

ActiveDocument.Host.Alert(ActiveDocument.doctype.name);
```

documentElement

Provides access to the node corresponding to the root (top-level) element of the document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMElement object

Usage in JScript

```
Document_object.documentElement;
```

Usage in VBScript

```
Document_object.documentElement
```

Example

```
// XMetaL Script Language JSCRIPT:

var elem;
elem = ActiveDocument.documentElement;
ActiveDocument.Host.Alert(elem.nodeName);
```

DocumentTitle

This property is unsupported.

DOMEnforceValid

By default, rules checking stays on during DOM operations that modify the document. Therefore, operations that would create an invalid document are disallowed. This property returns and sets whether rules checking can be turned off in the document during DOM operations.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = Document_object.DOMEnforceValid;
Document_object.DOMEnforceValid = boolState;
```

Usage in VBScript

```
dim boolState = Document_object.DOMEnforceValid
Document_object.DOMEnforceValid = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in a "Journalist" DTD document
ActiveDocument.DOMEnforceValid = false;
var newTag = ActiveDocument.createElement("Article");
var r = ActiveDocument.Range;
r.MoveToElement("Para");
var curNode = r.ContainerNode;
// With rules checking on, this operation would
// be disallowed
curNode.appendChild(newTag);
```

DropClipboardText
Returns True if the On_Document_Before_DropText event macro is called because text was pasted from the clipboard. Returns False if the macro is called because of a drag-and-drop action. This property should be used only in On_Document_Before_DropText.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
ActiveDocument.DropClipboardText;
```

Usage in VBScript

```
ActiveDocument.DropClipboardText
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Find out if the action was a Paste or a Drag-and-Drop
If ActiveDocument.DropClipboardText = True then
Msgbox ("Was pasted.")
Else
MsgBox ("Was dragged-and-dropped.")
End If
```

DropDataObject
This property is available in the On_Drop_FmtType event macros and in the XMetaL XMAX OnDrop event. It returns an `IUnknown` pointer to the data object that was dropped. The macro can then use this pointer to pass the object to a DLL to be interpreted. In XMetaL Author, returns the same object as the `Application.DropDataObject` property.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

IUnknown pointer

Usage in JScript

```
ActiveDocument.DropDataObject;
```

Usage in VBScript

```
ActiveDocument.DropDataObject
```

Example

```
' XMetaL Script Language VBScript:
' put this code into On_Drop_External_Text event macro
' DLLName.MyObject is not a real DLL
Set obj = CreateObject("DLLName.MyObject")
str = obj.GetText(Document.DropDataObject)
Set rng = Document.DropPoint
rng.PasteString str
rng.Select
```

DropInternalText
Returns True if the text is from this application and False if it is from another application. This property is intended for use in the On_Before_Drop_Text event macro.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
ActiveDocument.DropInternalText;
```

Usage in VBScript

```
ActiveDocument.DropInternalText
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Find out if the text came from this application or another
If ActiveDocument.DropInternalText = True then
MsgBox ("Came from this application.")
```

```
Else
MsgBox ("Came from another application.")
End If
```

DropPoint
Returns a Range object corresponding to the point in the document where the object is dropped. This property is meant to be used in On_Document_Before_DropText and On_Document_After_DropText event macros. This property is similar to `Application.DropText` (which is only available when scripting for XMetaL Author), but the two properties are not identical.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Range object

Usage in JScript

```
ActiveDocument.DropPoint;
```

Usage in VBScript

```
ActiveDocument.DropPoint
```

Example

```
// XMetaL Script Language JSCRIPT:
// Add this code to On_Document_Before_DropText
// Remove tags in the dropped text when the drop target is "Para" element

removeTags();

function removeTags() {
if (!ActiveDocument.DropPoint || (ActiveDocument.DropPoint.ContainerName
 != "Para")) {
return;// drop target is not a "Para" element
}

var dropText = ActiveDocument.DropText;

// delete tags
dropText = dropText.replace(/<[^>]*>/g, "");

// Set ActiveDocument.DropText with new content
ActiveDocument.DropText = dropText;

return true;
}
```


DropRange

Returns the range that contains the contents that were pasted or dropped into the document. This property is intended for use in the On_Document_After_DropText event macro. In XMetaL Author, this property returns the same value as `Application.DropRange`.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Range object

Usage in JScript

```
ActiveDocument.DropRange;
```

Usage in VBScript

```
ActiveDocument.DropRange
```

Example

```
' XMetaL Script Language VBScript:
' Select the pasted or dropped text and
' surround it with paragraph tags
ActiveDocument.DropRange
Selection.Surround("P")
```

DropText

When you want to process text before it is dropped, you can use this property to get or set the dropped text. This property is meant to be used in On_Document_Before_DropText event macro. This property returns the same value as `Application.DropText` (available in XMetaL Author only).

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var myString = ActiveDocument.DropText;
ActiveDocument.DropText = myString;
```

Usage in VBScript

```
dim myString = ActiveDocument.DropText
ActiveDocument.DropText = myString
```

Example

```
// XMetaL Script Language JSCRIPT:
// Add this code to On_Document_Before_DropText
// Remove tags in the dropped text when the drop target is "Para" element

removeTags();

function removeTags() {
if (!ActiveDocument.DropPoint || (ActiveDocument.DropPoint.ContainerName
 != "Para")) {
return;// drop target is not a "Para" element
}

var dropText = ActiveDocument.DropText;

// delete tags
dropText = dropText.replace(/<[^>]*>/g, "");

// Set ActiveDocument.DropText with new content
ActiveDocument.DropText = dropText;

return true;
}
```

DropTextIsMultiCell
Determines whether clipboard data was generated from an XMetaL HTML or CALS multi-cell table cut or copy operation. For use only in the On_Document_Before_DropText macro. If True, also implies that DropText is read only.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Document_object.DropTextIsMultiCell;
```

Usage in VBScript

```
Document_object.DropTextIsMultiCell
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert(ActiveDocument.DropTextIsMultiCell);
```


EditAttributeName

Returns the attribute-type whose custom-edit button was clicked within the grid-control of the Attribute Inspector. This property is valid only during the On_Edit_Attribute_From_AI event macro.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var name = Document_object.EditAttributeName;
```

Usage in VBScript

```
var name = Document_object.EditAttributeName
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert("The attribute selected when the custom-edit
button was clicked is " + ActiveDocument.EditAttributeName);
```

EditAttributeNode

Returns the DOMElement node for the element selected in the dropdown list of the Attribute Inspector when the custom-edit button was clicked. This property is valid only during the On_Edit_Attribute_From_AI event macro.

Applies to

XMetaL Author

Access

Read only

Returns

DOMElement object

Usage in JScript

```
var node = Document_object.EditAttributeNode;
```

Usage in VBScript

```
var node = Document_object.EditAttributeNode
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert("The element selected when the custom-edit
button was clicked is " + ActiveDocument.EditAttributeNode.nodeName);
```

EditAttributeValue
Returns or accepts the attribute value whose custom-edit button was clicked within the grid-control of the Attribute Inspector. The attribute value is for the attribute denoted by the EditAttributeName/EditAttributeNode properties. This property is valid only during the On_Edit_Attribute_From_AI event macro.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var value = Document_object.EditAttributeValue;
```

Usage in VBScript

```
var value = Document_object.EditAttributeValue
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert("The value of the attribute selected when the
custom-edit button was clicked is " + ActiveDocument.EditAttributeValue);
```

FormattingUpdating
Enables or disables the updating of formats in a document. This property is set to True by default. If False, the formatting is not updated, which can improve the performance of some macros. You must experiment to determine for which macros the performance can be improved by using the property. For example, if you are running a macro that makes many calls to SetRenderedContent, performance will improve if this property is set to False.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var myBoolean = ActiveDocument.FormattingUpdating;
ActiveDocument.FormattingUpdating = myBoolean;
```

Usage in VBScript

```
dim myBoolean = ActiveDocument.FormattingUpdating
ActiveDocument.FormattingUpdating = myBoolean
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.FormattingUpdating = false;
// ... many operations requiring formatting updates
ActiveDocument.FormattingUpdating = true;
```

FullName

The full path and file name of the active document. If the document has never been saved, an empty string is returned.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Document_object.FullName;
```

Usage in VBScript

```
Document_object.FullName
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.Host.Alert(ActiveDocument.FullName);
```

Host

Returns the DocumentHost object corresponding to the host application of the document. This property provides a way for a document to communicate with its host application.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DocumentHost object

Usage in JScript

```
var myDocumentHost = ActiveDocument.Host;
```

Usage in VBScript

```
dim myDocumentHost = ActiveDocument.Host
```

Example

```
// XMetaL Script Language JSCRIPT:
// Set the cursor to "not allowed"
// if mouse is over a read-only container,
// and to the default cursor otherwise.

// Get the node the mouse is over
var curNode=ActiveDocument.MouseOverNode;
var rng = ActiveDocument.Range;
// Select the contents of the node
rng.SelectNodeContents(curNode);
// Check the read-only status and
// set the cursor accordingly
if (rng.ReadOnly) {
ActiveDocument.Host.SetCursor(17);
} else {
ActiveDocument.Host.SetCursor(0);
}
```

implementation

The DOMImplementation object that handles this document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMImplementation object

Usage in JScript

```
Document_object.implementation;
```

Usage in VBScript

```
Document_object.implementation
```


Example

```
// XMetaL Script Language JSCRIPT:

var curImpl, xmlFeature;
curImpl = ActiveDocument.implementation;
xmlFeature = curImpl.hasFeature("XML","1.0");
ActiveDocument.Host.Alert(xmlFeature);
xmlFeature = curImpl.hasFeature("XML","2.1");
ActiveDocument.Host.Alert(xmlFeature);
```

InlineMediaVisible

Returns true if the `ActiveDocument` is displaying inline images and/or other media via ActiveX controls.

Access

Read only

Applies to

XMetaL® XMAX™ and XMetaL® Author

Returns

Boolean

Usage in JScript

```
var inlineImagesVisible =
    ActiveDocument.InlineMediaVisible;
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.InlineMediaVisible;
```

internalSubsetEditable

Returns True if the internal subset is editable; otherwise False.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
Document_object.internalSubsetEditable;
```

Usage in VBScript

```
Document_object.internalSubsetEditable
```

Example

```
// XMetaL Script Language JSCRIPT:
// Make internal subset editable if it is not already
if (!ActiveDocument.internalSubsetEditable) {
  ActiveDocument.internalSubsetEditable = true;
}
```

IsEntityDeclared

Returns True if the general entity specified has been declared in the current document or its DTD or schema. This property does not apply to parameter entities.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Document_object.IsEntityDeclared("strName");
```

Usage in VBScript

```
Document_object.IsEntityDeclared("strName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Declare entity if it is not already declared

if (!ActiveDocument.IsEntityDeclared("Product")) {
ActiveDocument.DeclareTextEntity("Product");
}
```

IsHTML

Always returns False. Indicates that the document being edited is not a HTML document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Document_object.IsHTML;
```

Usage in VBScript

```
Document_object.IsHTML
```

Example

```
// XMetaL Script Language JSCRIPT:

if (!ActiveDocument.IsHTML) {
ActiveDocument.Host.Alert("This is not a HTML document.");
}
```

IsInWellFormedEditingMode
Returns True if the document is in well-formed editing mode.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Document_object.IsInWellFormedEditingMode;
```

Usage in VBScript

```
Document_object.IsInWellFormedEditingMode
```

Example

```
// XMetaL Script Language JSCRIPT:
// Tell the user if the document is well formed or not

if (ActiveDocument.IsInWellFormedEditingMode) {
ActiveDocument.Host.Alert("This is a WellFormed document.");
} else {
ActiveDocument.Host.Alert("This is not a WellFormed document.");
}
```

IsNotationDeclared
Returns True if the notation entity specified has been declared in the current document or its DTD or schema. This property does not apply to parameter entities.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Document_object.IsNotationDeclared("cgm");
```

Usage in VBScript

```
Document_object.IsNotationDeclared("cgm")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Declare entity if it is not already declared

if (!ActiveDocument.IsNotationDeclared("cgm")) {
  ActiveDocument.DeclareNotation("cgm", "pubid", "sysid");
}
```

IsSGML

Returns True if the document being edited is an SGML document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Document_object.IsSGML;
```

Usage in VBScript

```
Document_object.IsSGML
```

Example

```
// XMetaL Script Language JSCRIPT:
```



```
if (ActiveDocument.IsSGML) {
ActiveDocument.Host.Alert("This is an SGML document.");
}
```

IsValid
Indicates whether the document is valid according to the DTD or schema of the current document. Displays no messages to the user.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Document_object.IsValid;
```

Usage in VBScript

```
Document_object.IsValid
```

Example

```
// XMetaL Script Language JSCRIPT:

if (ActiveDocument.IsValid) {
ActiveDocument.Host.Alert("Document is valid.");
} else {
ActiveDocument.Host.Alert("Document is not valid.");
}
```

IsWordSpellingCorrect
Indicates if the word under the selection is spelt correctly.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Document_object.IsWordSpellingCorrect;
```

Usage in VBScript

```
Document_object.IsWordSpellingCorrect
```

Example

```
// XMetaL Script Language JSCRIPT:

if (!ActiveDocument.IsWordSpellingCorrect) {
ActiveDocument.Host.Alert("The wprd is not spelt correctly.");
}
```

IsXML
Returns True if the document being edited is an XML document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Document_object.IsXML;
```

Usage in VBScript

```
Document_object.IsXML
```

Example

```
// XMetaL Script Language JSCRIPT:

if (ActiveDocument.IsXML) {
ActiveDocument.Host.Alert("This is a XML document.");
}
```

MarkWalker
This property is reserved for internal use only.

MinimizeTagIcons
You can use this property to minimize tag icons. Users can see the tag icons as small (without the tag name) or large (normal) size.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolTagsAreMinimized = Document_object.MinimizeTagIcons;
Document_object.MinimizeTagIcons = true;
```

Usage in VBScript

```
dim boolTagsAreMinimized = Document_object.MinimizeTagIcons
Document_object.MinimizeTagIcons = true
```

Example

```
// XMetaL Script Language JSCRIPT:
// If tags are not already minimized,
// make them minimized.

if (ActiveDocument.MinimizeTagIcons == false) {
ActiveDocument.MinimizeTagIcons = true;
}
```

MouseOverNode

Returns the node corresponding to the element, comment, entity, or processing instruction that the mouse is currently over. This property should be used only inside the On_Mouse_Over and On_Mouse_Out event macros.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNode object

Usage in JScript

```
var myDOMNode = ActiveDocument.MouseOverNode;
```

Usage in VBScript

```
dim myDOMNode = ActiveDocument.MouseOverNode
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the element name in the status bar
var curNode = ActiveDocument.MouseOverNode;
var curNodeName = curNode.NodeName;

if (curNodeName!="") {
ActiveDocument.Host.SetStatusText = curNodeName;
}
```

Name
The file name (not including the path) of the document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
myFileName=Document_object.Name;
```

Usage in VBScript

```
myFileName=Document_object.Name
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.Host.Alert(ActiveDocument.Name);
```

Path
The path (not including the filename) of the document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
pathToFile=Document_object.Path;
```

Usage in VBScript

```
pathToFile=Document_object.Path
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.Host.Alert(ActiveDocument.Path);
```

PreviousViewType
Returns the XMetaL document view that the document was displayed in before the current view. This property can be used in any macro, but is particularly intended for use in the On_View_Change event macro and the OnViewChange event (XMetaL XMAX only).

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
var intPrevViewType = Document_object.PreviousViewType;
```

Allowed values are:

- sqViewUnspecified (-1): Unspecified
- sqViewNormal (0): Normal view
- sqViewTagsOn (1): Tags On view
- sqViewPlainText (2): Plain Text view (XMetaL Author only)
- sqViewBrowse (3): Page Preview (XMetaL Author only)

Usage in VBScript

```
dim intPrevViewType = Document_object.PreviousViewType
```

Example

```
// XMetaL Script Language JSCRIPT:
if ((ActiveDocument.ViewType == 0 ||
ActiveDocument.ViewType == 1) &&
ActiveDocument.PreviousViewType == 2) {
// Code to restore read-only property
// ... }
```

querySelector
This property is reserved for internal use.

querySelectorAll
This property is reserved for internal use.

Range
Creates and returns a Range object corresponding to the user's selection in the document. It is often more useful to use a Range object rather than a Selection object when moving around the document for the following reasons: there are no restrictions on where Range objects can be placed; the movement of Range objects is invisible to the user; you can have as many Range objects as you need, but only one selection; if the user moves the insertion point/selection, Range objects are not affected; and Range objects stay the same when you switch documents.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Range object

Usage in JScript

```
Document_object.Range;
```

Usage in VBScript

```
Document_object.Range
```

Example

```
// XMetaL Script Language JSCRIPT:
// Find Title element and display its text

var rng = ActiveDocument.Range;
rng.MoveToDocumentStart();
rng.Find.Execute("<Title");
rng.SelectContainerContents();
var title = rng.Text;
ActiveDocument.Host.Alert(title);
```

ReadOnlyHint
This property is reserved for internal use only.

RefreshCssStyleDoc
Returns the XMetaL document instance that is ready for additional CSS rules to be appended to the default style sheets for your customization (for example, journalist.css). Valid only during On_View_RefreshCssStyle or On_StructureView_RefreshCssStyle event macro.

Applies to

XMetaL Author and XMetaL XMAX

Access

Read only

Returns

Document object

Usage in JScript

```
ActiveDocument.RefreshCssStyleDoc();
```

Usage in VBScript

```
ActiveDocument.RefreshCssStyleDoc
```

Example

```
var xmDoc = ActiveDocument.RefreshCssStyleDoc;
```

RegisterNonCopyableAttr
This property is reserved for internal use only.

RemoveCompareFunc
This property is reserved for internal use.

ResourceSet
Returns the ResourceSet object which contains all document assets and user-defined properties. These assets and properties are set when a document customization is loaded.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

ResourceSet object

Usage in JScript

```
var rs = ActiveDocument.ResourceSet;
```

Usage in VBScript

```
set rs = ActiveDocument.ResourceSet
```

Example

```
// XMetaL Script Language JSCRIPT:
// Get the full set of document resources
```

```
var rs = ActiveDocument.ResourceSet;
ActiveDocument.Host.Alert("There are " + rs.Count + " items in the set.")
```

RMPreserveMode
This property is reserved for internal use only.

RulesChecking
Sets and returns the rules checking state of the specified document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Description

In XMetaL Author, you can add the 'Rules checking always off' option to the **General** tab of the **Options** dialog box by setting `rules_checking_always_off_option_shown` to `true` in the XMetaL configuration file. If rules checking is turned off using this option, it cannot be turned back on using `Document.RulesChecking`. In XMetaL Author, you can also turn rules checking off by setting `rules_checking_always_off` to `true` in the XMetaL configuration file. This has the same effect as turning rules checking off from the Options dialog box. In XMetaL Author, a 'Turn rules checking off?' message can displayed when the user attempts to insert invalid markup if you set `show_rules_check_off_dialog` to `true` in the XMetaL configuration file.

Usage in JScript

```
var boolRulesCheckingOn = Document_object.RulesChecking;
Document_object.RulesChecking = true;
```

Usage in VBScript

```
dim boolRulesCheckingOn = Document_object.RulesChecking
Document_object.RulesChecking = True
```

Example

```
//XMetaL Script Language JSCRIPT:

if (ActiveDocument.RulesChecking) {
ActiveDocument.RulesChecking = false;
} else {
ActiveDocument.RulesChecking = true;
}
```

RulesFile
Returns the full path to the document's rules file.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strRulesFile = Document_object.RulesFile;
```

Usage in VBScript

```
dim strRulesFile = Document_object.RulesFile
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.Host.Alert(ActiveDocument.RulesFile);
```

Saved

Indicates whether the active document has been saved since changes were last made.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Description

Setting this property to True makes the document appear to be ‘clean’: the **Save** command and the corresponding toolbar button are grayed out in XMetaL Author, and closing the file does not result in a prompt to save changes. In XMetaL Author and XMetaL XMAX, setting this property to True clears the Undo and Redo stacks. However, this does not actually save the document. In XMetaL Author, if the document has never been saved, the **Save** command and toolbar button are not grayed out. Setting this property to False has no effect.

Usage in JScript

```
Document_object.Saved;
```

Usage in VBScript

```
Document_object.Saved
```

Example

```
// XMetaL Script Language JSCRIPT:

if (!ActiveDocument.Saved) {
ActiveDocument.Save();
}
```

SpellAutoCorrect

Returns True if the document has automatic correction of spelling mistakes while typing enabled.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/Write

Returns

Boolean

Usage in JScript

```
Document_object.SpellAutoCorrect = false;
```

Usage in VBScript

```
Document_object.SpellAutoCorrect
```

Example

```
// XMetaL Script Language JSCRIPT:

if (ActiveDocument.SpellAutoCorrect) {
ActiveDocument.Host.Alert("This document is being actively corrected while
 typing.");
}
```

StructureViewVisible

Gets or sets the display state of the XMetaL structure view.

Applies to

XMetaL Author and XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = Document_object.StructureViewVisible;
Document_object.StructureViewVisible = boolState;
```

Usage in VBScript

```
dim boolState = Document_object.StructureViewVisible
Document_object.StructureViewVisible = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
// Toggle Structure View display state

if (ActiveDocument.StructureViewVisible) {
ActiveDocument.StructureViewVisible = false;
} else {
ActiveDocument.StructureViewVisible = true;
}
```

StyleSheets

Returns a list of stylesheets as a StyleSheetList object.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

StyleSheetList object

Description

The item that the StyleSheetList object returns is a stylesheet as a DOMCSSStylesheet object. To use this property to return a DOMCSSStyleSheet object, you would follow this procedure:

```
var styleSheetList = Document.StyleSheets;

if (styleSheetList.length > 0) {
var styleSheet = styleSheetList.item(index);
}
```

where `length` is a long integer representing the number of stylesheets in the list and `index` is a long integer representing the index number of the stylesheet you want to read.

The following properties of the DOMCSSStyleSheet interface are part of the W3C specifications, but are not implemented:

- .disabled
- .href

- .media
- .ownerNode
- .ownerRule
- .parentStyleSheet
- .Title
- .type

The only fully-implemented property of the DOMCSSStyleSheet interface is `DOMCSSStyleSheet.cssRules`. Use this interface to return a list of CSSRules as a DOMCSSRuleList object, which can then be used to return an individual rule as a CSSStyleRule object. To continue the above code, to return CSSRules:

```
if (styleSheet != null) {
var rulesList = styleSheet.cssRules;

if (rulesList != null && rulesList.length > 0) {
var rule = rulesList.item(index)
}
}
```

where `length` is a long integer representing the number of rules in the list and `index` is a long integer representing the index number of the rule you want to read.

The CSSStyleRule object has these properties:

- cssText - A string value that represents the text of the rule
- parentRule - Returns the parent rule as a CSSStyleRule object
- parentStyleSheet - Returns the parent stylesheet for the current CSSStyleRule as a DOMCSSStyleSheet object
- selectorText - A string value that represents the CSS selector
- Style - Returns the rule's CSS style declaration, an ordered collection, as a CSSStyleDeclaration object
- type - An integer representing the CSSStyleRule type

The CSSStyleDeclaration that is returned by using the `CSSStyleRule.Style` property has these properties:

- cssText - A string value that represents the text of the style declaration block
- parentRule - Returns the parent rule as a CSSStyleRule object. The CSS style rule that contains the declaration.
- length - A long value indicating the number of properties that have been explicitly set

The CSSStyleDeclaration that is returned by using the CSSStyleRule.Style property has these methods:

- getPropertyPriority(name as string) - Use the parameter `name` of type `DOMString`. getPropertyPriority represents a string value that is used to retrieve the priority of a CSS property (e.g., the 'important' qualifier) if the property has been explicitly set in this declaration block.
- getPropertyValue(name as string) - Use the parameter `name` of type `DOMString`. getPropertyValue represents a string value that contains the CSS property value for the property name. Returns an empty string if the property has not been set.
- item(index as long) - This method is used to retrieve the property name that has been explicitly set in this declaration block. The order of the properties retrieved using this method does not have to be the order in which they were set. This method can be used to iterate over all properties in this declaration block.

  Use the parameter `index`, of the type `unsigned long`, which represents the index of the property name to retrieve. The return value is `DOMString`, a string representing the name of the property at this ordinal position. The string is empty if no property exists at this position.

Usage in JScript

```
ss = ActiveDocument.StyleSheetList;
```

Usage in VBScript

```
ss = ActiveDocument.StyleSheetList
```

Example

```
// XMetaL Script Language JSCRIPT:
// Use this code in the Journalist dtd.
// Warning! This code may trigger anti-virus software.

if (ActiveDocument.ViewType == sqViewNormal
|| ActiveDocument.ViewType == sqViewTagsOn) {
var fso, f;
fso = new ActiveXObject("Scripting.FileSystemObject");
f = fso.CreateTextFile("C:\\temp\\CSS-OM Test.txt", true);
var styleSheetList = ActiveDocument.styleSheets;

if (styleSheetList.length > 0) {
var stylesheet = styleSheetList.item(0);

if (stylesheet != null) {
var ruleslist = stylesheet.cssRules;

if (ruleslist != null && ruleslist.length > 0) {
var i;

for (i = 0; i < ruleslist.length; i++) {
var rule = ruleslist.item(i);

if (rule != null && rule.type == 1) {
f.WriteLine(rule.selectorText);
var style = rule.style;

if (style != null) {
var j;

for (j = 0; j < style.length; j++) {
var propname = style.item(j);
var propval = style.getPropertyValue(propname);
f.WriteLine("" + propname + " : " + propval);
}
}
}
}
}
}
}
f.Close();
}
```

TagsOnGraphicalTables

Toggles display of tables in Tag On view mode between graphical or tags.

Applies to

XMetaL Author and XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = Document_object.TagsOnGraphicalTables;
Document_object.TagsOnGraphicalTables = boolState;
```

Usage in VBScript

```
dim boolState = Document_object.TagsOnGraphicalTables
Document_object.TagsOnGraphicalTables = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
// Toggle table display mode

if (ActiveDocument.TagsOnGraphicalTables) {
  ActiveDocument.TagsOnGraphicalTables = false;
} else {
  ActiveDocument.TagsOnGraphicalTables = true;
}
```

Title

Gets or sets the window title for the document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var strTitle = Document_object.Title;
Document_object.Title = strTitle;
```

Usage in VBScript

```
dim strTitle = Document_object.Title
Document_object.Title = strTitle
```


Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.Host.Alert(ActiveDocument.Title);
ActiveDocument.Title="A new title";
ActiveDocument.Host.Alert(ActiveDocument.Title);
```

TrackRevisions
Returns True if changes are being tracked in the current document. Can also be set to False to turn revision tracking off.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
ActiveDocument.TrackRevisions = True;
```

Usage in VBScript

```
ActiveDocument.TrackRevisions = True
```

Example

```
// XMetaL Script Language JSCRIPT:
// Remember the revision tracking original state
// then turn on change tracking
var origState = ActiveDocument.TrackRevisions;
ActiveDocument.TrackRevisions = true;
// Create a new text node and insert it
// at the beginning of the current element
var strText = "Hello world.";
textNode = ActiveDocument.createTextNode(strText);
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
curNode.insertBefore(textNode,curNode.firstChild);
// Set revision tracking back to its original state.
ActiveDocument.TrackRevisions = origState;
```

ValidationErrorList
Returns a ValidationErrorList object containing the list of errors found in the most recent call to `Document.Validate`. If there is no ValidationErrorList object to return, the property will return null. If changes are made to the document after `Document.Validate` is called, the ValidationErrorList returned may contain errors that no longer exist in the document. This can cause unpredictable behavior when calling properties and methods. You should only manipulate the ValidationErrorList object immediately after a call to `Document.Validate`.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

ValidationErrorList object

Usage in JScript

```
ActiveDocument.ValidationErrorList;
```

Usage in VBScript

```
ActiveDocument.ValidationErrorList
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
// web page.

// Validates the active document, and displays a list of validation errors.
// When you click OK, you go to the error numbered gErrNum.
// gErrNum persists (if the script is run in an Internet Explorer web
page),
// and you can run validate() a number of times to visit each
// validation error in the document in sequence.

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myXMControl" HEIGHT="500" WIDTH="500">
</OBJECT>

<SCRIPT>
// define global variable
var gErrnum = 0;

function assert(bool) {
if (bool == false) {
ActiveDocument.Host.Alert("There was a programming error somewhere.");
}
}

function validate() {
var doc = myXMControl.Document;
if (doc) {
doc.Validate();
var vel = doc.ValidationErrorList;

if (vel) {

if (vel.Count == 0) { // valid document
assert(doc.IsValid);
ActiveDocument.Host.Alert("The document is valid.");

} else { // one or more validation errors
assert(!doc.IsValid);
var msg = "Errors:\n\nItem\tLevel\tNumber\tMessage\n";
var index = 1;
var ve = vel.Item(index++);
```



```
while (ve) {
msg += index-1 + "\t" + ve.ErrorLevel
+ "\t" + ve.ErrorType
+ "\t" + ve.ErrorMessage;
ve = vel.Item(index++);
msg += "\n";
}
gErrNum++;

if (gErrNum > vel.Count) {
gErrNum = 1; // go back to the first error
}
ve = vel.Item(gErrNum);
msg += "\nPress OK to go to error " + gErrNum + " (Item " + gErrNum + ")";
ActiveDocument.Host.Alert(msg);

// There are 2 ways to go to the error:
//ValidationErrorList.GoToError(Index)
//ValidationError.Range.Select()
var doItWithGoToError = true;
if (doItWithGoToError) {
vel.GoToError(gErrNum);
} else {
ve.Range.Select();
}
}
}
}
}
</SCRIPT>
```

ViewType
Returns or sets the XMetaL document view that the document is displayed in. This property can be used in the On_Document_Open_View event macro, but not in the On_Document_Open_Complete event macro. It can also be used in the `OnViewChange` event (XMetaL XMAX only).

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Integer

Usage in JScript

```
var intViewType = Document_object.ViewType;
Document_object.ViewType = intViewType;
```

Allowed values are:

- sqViewUnspecified (-1): Unspecified
- sqViewNormal (0): Normal view
- sqViewTagsOn (1): Tags On view
- sqViewPlainText (2): Plain Text view (XMetaL Author only)
- sqViewBrowse (3): Page Preview (XMetaL Author only)

Usage in VBScript

```
dim intViewType = Document_object.ViewType
Document_object.ViewType = intViewType
```

Example

```
// XMetaL Script Language JSCRIPT:
// If in Normal View, switch to Tags On

if (ActiveDocument.ViewType==0) {
ActiveDocument.ViewType=1;
}
```

xml

You can access the entire XML/SGML document (with document type declaration and XML processing instruction) and store the information in a variable using this property.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Description

One example is when XMetaL is the front end of a larger solution with other applications interacting with it. For instance, in a publishing house, there are times when an author wants to preview XML or SGML documents in a third-party application. With this property, XMetaL can hand off the entire document, including the document type declaration, to the third-party application. If you are scripting for XMetaL XMAX, see also XMetaL XMAX `Document.xml`. If you want to return the entire document with change tracking processing instructions intact, use the `Document.xmlWithCT` property.

Usage in JScript

```
var strDoc = ActiveDocument.xml;
```

Usage in VBScript

```
dim strDoc = ActiveDocument.xml
```

Example

```
// XMetaL Script Language JSCRIPT:
// retrieve the entire document into a string

var doc = ActiveDocument.xml;
ActiveDocument.Host.Alert(doc);
```


xmlWithCT

XML source recovered using the `Document.xml` property has all change tracking processing instructions removed. Use this property to recover XML source with change tracking processing instructions preserved.

Applies to

XMetaL Author

Access

Read only

Returns

String value corresponding to the entire document with change tracking processing instructions preserved.

Usage in JScript

```
var strDoc = ActiveDocument.xmlWithCT;
```

Usage in VBScript

```
dim strDoc = ActiveDocument.xmlWithCT
```

Example

```
// XMetaL Script Language JSCRIPT:
// retrieve the entire document and all changes
// made with Change Tracking

var doc = ActiveDocument.xmlWithCT;
ActiveDocument.Host.Alert(doc);
```

Methods

The Document interface supports the following methods, in addition to those inherited from the DOMNode interface.

AcceptAllChanges

Accepts all tracked and applies the changes to the document. The changes are no longer marked as tracked. Normally, this method is used to handle an event in the user interface, such as a click on a toolbar button.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.AcceptAllChanges();
```

Usage in VBScript

```
ActiveDocument.AcceptAllChanges
```

Example

```
' XMetaL Script Language VBScript:
' XMetaL XMAX example -- assumes myXMControl was previously initialized
' Accepts all changes when a button named AcceptAllButton is clicked
' in the user interface

Sub AcceptAllButton_Click Handles AcceptAllButton.Clicked
If Not (myXMControl.Document Is Nothing) Then
myXMControl.Document.AcceptAllChanges
End If
End Sub
```

AcceptChange
Accepts the currently selected tracked change. Applies the change to the document. The change is no longer marked as tracked. Normally, this method is used to handle an event in the user interface, such as a click on a toolbar button.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.AcceptChange();
```

Usage in VBScript

```
ActiveDocument.AcceptChange
```

Example

```
' XMetaL Script Language VBScript:
' XMetaL XMAX example -- assumes myXMControl was previously initialized
' Accepts the currently selected change when a button named
' AcceptButton is clicked in the user interface

Sub AcceptButton_Click Handles AcceptButton.Click
If (TypeName(myXMControl.Document)<>"Nothing") And
myXMControl.Document.CanAcceptOrRejectChange) Then
myXMControl.Document.AcceptChange
End If
End Sub
```

AcceptDropFormat
Enables an event macro called On_Drop_strFmtType to be called when an object with the specified is dropped into this document. A macro specified with this method applies to the current document only. Generally this macro calls a DLL that converts the object into a format which XMetaL can use. If the object is dropped onto the background of the XMetaL work area, the macro On_Drop_Files is called instead. Generally, this method

is called in the On_Document_Open_Complete event macro. If you are scripting for XMetaL Author, you can also use `Application.AcceptDropFormat`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.AcceptDropFormat(strFmtName, strFmtType);
```

`strFmtName` is the name that the application from which the object came gives to that kind of object; `strFmtType` is a string (no spaces) of your choice.

Usage in VBScript

```
Document_object.AcceptDropFormat(strFmtName, strFmtType)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Causes the On_Drop_HTML macro to be called
// when a selection is dragged from MS Word 2000
// into the XMetaL document window
ActiveDocument.AcceptDropFormat("HTML Format", "HTML");
```

Activate

Activates the specified Document object (the corresponding document becomes the active document, and is then also represented by the ActiveDocument object). The document must already be open.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.Activate();
```

Usage in VBScript

```
Document_object.Activate
```

Example

```
// XMetaL Script Language JSCRIPT:
// Store the object representing
// the active document
var doc = Application.ActiveDocument;
```

```
// Add a document; the new document automatically
// becomes the active document
Documents.Add();
// re-activate the original active document
doc.Activate();
```

AddWordToDictionary
Return true if successful adding the word under the cursor into the User Word List with a <skip> flag.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
var succeeded = Document_object.AddWordToDictionary();
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.AddWordToDictionary(); // Red-squiggly line goes away now
```

attachTransclusion
This method is reserved for internal use only.

attachTransclusionEx
This method is reserved for internal use only.

BeginMutations
Tells XMetaL to group subsequent document changes via API into a single undoable operation. The optional nice-name string parameter will appear in the Undo/Redo dropdown list toolbar control. Call `EndMutations` to signal the end of document changes. BeginMutations calls cannot be nested.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.BeginMutations("strNiceName");
```

Usage in VBScript

```
ActiveDocument.BeginMutations "strNiceName"
```


Example

```
// XMetaL XMAX example - assumes that myXMControl was previously
initialized

myXMControl.Document.BeginMutations();
myXMControl.Selection.InsertElement("p");
myXMControl.Selection.InsertElement("b");
myXMControl.Document.EndMutations(true);
```

CanAcceptOrRejectChange

Returns True if the current selection is in a revision mark, which means that the current selection is a tracked change which can be accepted or rejected. Returns False otherwise. Normally, this method is used to enable or disable toolbar buttons and other user interface objects.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
ActiveDocument.CanAcceptOrRejectChange();
```

Usage in VBScript

```
ActiveDocument.CanAcceptOrRejectChange
```

Example

```
'XMetaL Script Language VBScript
' XMetaL XMAX example - assumes that myXMControl was previously initialized
' Enables/disables AcceptButton and RejectButton

Sub myXMControl_OnUpdateUI
If (TypeName(myXMControl.Document)<>"Nothing") And
myXMControl.Document.CanAcceptOrRejectChange) Then
AcceptButton.Enabled = False
RejectButton.Enabled = False
Else
AcceptButton.Enabled = True
RejectButton.Enabled = True
End If
End Sub
```

ClearAllChangedStates

Turns off all ‘changed’ flags, so all nodes in the document appear unchanged. After a call to this method, `Document.ChangedNodes` will return a zero-length list.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.ClearAllChangedStates();
```

Usage in VBScript

```
Document_object.ClearAllChangedStates
```

Example

```
// XMetaL Script Language JSCRIPT:
// Change node should not return null so the length should be "0" not
"null"
// Create a text node from scratch - by walking to the child of
// an existing para containing text that has never been examined before

ActiveDocument.ClearAllChangedStates();
var txt = Selection.ContainerNode.firstChild;
var chgdNodes = ActiveDocument.ChangedNodes;
ActiveDocument.Host.Alert(chgdNodes.length); // should be one, the new
text node

for (i=0;i<chgdNodes.length;i++){

if (chgdNodes(i) == null){
ActiveDocument.Host.Alert("Surprise! A null value."); // unexpected
} else {
ActiveDocument.Host.Alert(chgdNodes(i).nodeType);
}
}
```

ClearAllChangedStatesbyKey

Turns off ‘changed’ flags for the indicated key; ChangedNodesbyKey returns a zero-length list.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.ClearAllChangedStatesbyKey("strKey");
```

`strKey` is a string value that indicates the key to turn off the changed flags for.

Usage in VBScript

```
Document_object.ClearAllChangedStatesbyKey("strKey")
```


Example

```
// XMetaL Script Language JSCRIPT:
// If you are scripting for XMetaL Author,
// put 'var mykey ="abc"' into On_Application_Open,
// and put the following in a local macro.
// If you are scripting for XMetaL XMAX, define mykey locally—
// that is, uncomment the following line:
// var mykey = "abc";

if (ActiveDocument.ViewType==2){
ActiveDocument.ViewType=1;
}
var nds = ActiveDocument.ChangedNodesbyKey(mykey);
var l = nds.length;
var changenode = null;
var name = null;
var f = 0;
ActiveDocument.Host.Alert(l);

if (l == 0){
ActiveDocument.Host.Alert("The method works!");
} else {

for (var i=0; i<=l;i++){
f++;
changenode = nds.item(i);
name = changenode.nodeName;
ActiveDocument.Host.Alert(name
+ "\n\nNewNode:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey, "NewNode",
 changenode)
+ "\nContentInserted:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"ContentInserted", changenode)
+ "\nContentDeleted:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"ContentDeleted", changenode)
+ "\nAttributesChanged:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"AttributesChanged", changenode));

if (f==nds.length) {
break;
}
}
}

//Uncomment these methods one at a time running

//Application.RemoveChangedNodeKey(mykey);
//ActiveDocument.ClearAllChangedStatesbyKey(mykey);
//ActiveDocument.ClearNodeChangedStates(mykey, changenode, true);
//ActiveDocument.ClearNodeChangedStatesbyKey(mykey,changenode,false);
```

ClearNodeChangedStates
Turns off all ‘changed’ flags for nodeNode (a DOMNode).

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.ClearNodeChangedStates(nodeNode, boolDeep);
```

If `boolDeep` is True, this operation is recursively applied to all descendent nodes of nodeNode.

Usage in VBScript

```
Document_object.ClearNodeChangedStates(nodeNode, boolDeep)
```

Example

```
// XMetaL Script Language JSCRIPT:
// If you are scripting for XMetaL Author,
// put 'var mykey ="abc"' into On_Application_Open,
// and put the following in a local macro.
// If you are scripting for XMetaL XMAX, define mykey locally—
// that is, uncomment the following line:
// var mykey = "abc";

if (ActiveDocument.ViewType==2){
ActiveDocument.ViewType=1;
}
var nds = ActiveDocument.ChangedNodesbyKey(mykey);
var l = nds.length;
var changenode = null;
var name = null;
var f = 0;
ActiveDocument.Host.Alert(l);

if (l == 0){
ActiveDocument.Host.Alert("The method works!");
} else {

for (var i=0; i<=l;i++){
f++;
changenode = nds.item(i);
name = changenode.nodeName;
ActiveDocument.Host.Alert(name
+ "\n\nNewNode:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey, "NewNode",
 changenode)
+ "\nContentInserted:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"ContentInserted", changenode)
+ "\nContentDeleted:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"ContentDeleted", changenode)
+ "\nAttributesChanged:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"AttributesChanged", changenode));

if (f==nds.length) {
break;
}
}
}

//Uncomment these methods one at a time running

//Application.RemoveChangedNodeKey(mykey);
//ActiveDocument.ClearAllChangedStatesbyKey(mykey);
//ActiveDocument.ClearNodeChangedStates(mykey, changenode, true);
//ActiveDocument.ClearNodeChangedStatesbyKey(mykey,changenode,false);
```

ClearNodeChangedStatesbyKey
Turns off all ‘changed’ flags for `strKey` and `nodeNode` (a DOMNode). If `boolDeep` is True, this operation is recursively applied to all descendent nodes of `nodeNode`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.ClearNodeChangedStatesbyKey("strKey", nodeNode, boolDeep);
```

Usage in VBScript

```
Document_object.ClearNodeChangedStatesbyKey("strKey", nodeNode, boolDeep)
```

Example

```
// XMetaL Script Language JSCRIPT:
// If you are scripting for XMetaL Author,
// put 'var mykey ="abc"' into On_Application_Open,
// and put the following in a local macro.
// If you are scripting for XMetaL XMAX, define mykey locally—
// that is, uncomment the following line:
// var mykey = "abc";

if (ActiveDocument.ViewType==2){
ActiveDocument.ViewType=1;
}
var nds = ActiveDocument.ChangedNodesbyKey(mykey);
var l = nds.length;
var changenode = null;
var name = null;
var f = 0;
ActiveDocument.Host.Alert(l);

if (l == 0){
ActiveDocument.Host.Alert("The method works!");
} else {

for (var i=0; i<=l;i++){
f++;
changenode = nds.item(i);
name = changenode.nodeName;
ActiveDocument.Host.Alert(name
+ "\n\nNewNode:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey, "NewNode",
 changenode)
+ "\nContentInserted:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"ContentInserted", changenode)
+ "\nContentDeleted:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"ContentDeleted", changenode)
+ "\nAttributesChanged:\t\t" + ActiveDocument.GetNodeStatebyKey(mykey,
"AttributesChanged", changenode));

if (f==nds.length) {
break;
}
}
}

//Uncomment these methods one at a time running

//Application.RemoveChangedNodeKey(mykey);
//ActiveDocument.ClearAllChangedStatesbyKey(mykey);
```

```
//ActiveDocument.ClearNodeChangedStates(mykey, changenode, true);
//ActiveDocument.ClearNodeChangedStatesbyKey(mykey,changenode,false);
```

Close
Closes the document, and may save changes. Scripts using `ActiveDocument.Close` should not be dragged or pasted into a document. `ActiveDocument.Close` can be used in macros, but should be avoided in event macros.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.Close([intSaveChanges]);
```

The allowed values are:

- sqPromptToSaveChanges (0)=prompt the user to save changes (the default)
- sqSaveChanges(1)=save changes without prompting
- sqDoNotSaveChanges(2)=do not save changes

Usage in VBScript

```
Document_object.Close [intSaveChanges]
```

Example

```
// XMetaL Script Language JSCRIPT:
Documents.Add(); // open a new document—this method with only work in
XMetaL Author
ActiveDocument.Close(2); // close the new document, do not save changes
```

createAttribute
Creates a DOMAttr node. To associate this node with an element, use `DOMElement.setAttributeNode`. Attributes can also be inserted with the `Selection.ContainerAttribute` and `Selection.ElementAttribute` methods. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMAttr object

Usage in JScript

```
Document_object.createAttribute("strName");
```


Usage in VBScript

```
Document_object.createAttribute("strName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create the node
IdNode = ActiveDocument.createAttribute("Id");
// Assign the attribute a value
IdNode.value = "xyzzy";
// Get the element node
elemNode = Selection.ContainerNode;
// Associate the attribute node with the element node
elemNode.setAttributeNode(IdNode);
```

createAttributeNS
Creates a DOMAttr node. This method is namespace aware. To associate this node with an element, use `DOMElement.setAttributeNodeNS`. Attributes can also be inserted with the `Selection.ContainerAttributeNS` and `Selection.ElementAttributeNS` methods. You can use these methods when you want to provide a local name (instead of a qualified name) for an attribute.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMAttr object

Usage in JScript

```
Document_object.createAttributeNS("strNamespaceURI", "strQualifiedName");
```

`strQualifiedName` is in the form `prefix:localName`. The attribute is associated with the namespace named `strNamespaceNameURI`.

Usage in VBScript

```
Document_object.createAttributeNS("strNamespaceURI", "strQualifiedName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create the node
IdNode =
ActiveDocument.createAttributeNS("http://www.mydomainname.com/Journalist",
 "myns:Id");
// Assign the attribute a value
IdNode.value = "xyzzy";
// Get the element node
elemNode = Selection.ContainerNode;
// Associate the attribute node with the element node
elemNode.setAttributeNode(IdNode);
```

createAttributeWF
This method is reserved for internal use only.

createCDATASection
Creates a DOMCDATASection node containing the string specified. CDATA sections can also be created with `Selection.InsertCDATASection`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMCDATASection object

Usage in JScript

```
Document_object.createCDATASection("strData");
```

Usage in VBScript

```
Document_object.createCDATASection("strData")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a CDATA section node and insert it
// after the current node
// Create the node
var cdSect = ActiveDocument.createCDATASection("<Enter>");
var rng = ActiveDocument.Range;
// Get the node after the current node
var nextNode = rng.ContainerNode.nextSibling;
// Insert the CDATA Section node before the "next" node
nextNode.parentNode.insertBefore(cdSect,nextNode);
```

createCharacterReference
Creates a character reference node with the value specified.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMCharacterReference object

Usage in JScript

```
Document_object.createCharacterReference(longValue);
```

`longValue` can be a decimal or hexadecimal number. In JScript a hexadecimal value is expressed as '0xNN'; in VBScript it is expressed as '&HNN'; 'NN' is two hexadecimal digits (0-f).

Usage in VBScript

```
Document_object.createCharacterReference(longValue)
```


Example

```
//XMetaL Script Language JSCRIPT:
// Create a character reference node and insert it
// in the current node
// Create the node
var charRef = ActiveDocument.createCharacterReference(300);
var rng = ActiveDocument.Range;

if (rng.CanInsertText) {
var curNode = rng.ContainerNode;
// Insert the CDATA Section node before the "next" node
curNode.appendChild(charRef);
}
```

createComment
Creates a DOMComment node containing the string specified. Comments can also be inserted with `Selection.InsertComment` method.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMComment object

Usage in JScript

```
Document_object.createComment("strText");
```

Usage in VBScript

```
Document_object.createComment("strText")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a comment node and insert it
// at the end of the current node
// Create the comment node
var comment = ActiveDocument.createComment("Correct?");
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
// Insert the comment node
curNode.appendChild(comment);
```

createDocumentFragment
Creates an empty DOMDocumentFragment object.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMDocumentFragment object

Usage in JScript

```
Document_object.createDocumentFragment();
```

Usage in VBScript

```
Document_object.createDocumentFragment
```

Example

```
// XMetaL Script Language JSCRIPT:
// Succeeds if pasted anywhere inside a Sect1
// element in a Journalist DTD document
// Create the document fragment
var docFrag = ActiveDocument.createDocumentFragment();
// Add Sect and Title nodes
var sect1Node = ActiveDocument.createElement("Sect1");
docFrag.appendChild(sect1Node);
var titleNode = ActiveDocument.createElement("Title");
sect1Node.appendChild(titleNode);
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
// If the current node is not a Sect1, try to find one

if (curNode) {

while (curNode.nodeName!="Sect1" && curNode.NodeName!="Article") {
rng.SelectElement();
var curNode = rng.ContainerNode;
}
}

// Insert the document fragment before the Sect1 node
if (curNode.nodeName=="Sect1"){
curNode.parentNode.insertBefore(docFrag,curNode);
}
```

createElement
Creates a DOMElement object with the name specified. Elements can also be inserted with `Selection.InsertElement`, `Selection.InsertElementWithRequired`, and `Selection.InsertWithTemplate`. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMElement object

Usage in JScript

```
Document_object.createElement("strName");
```

Usage in VBScript

```
Document_object.createElement("strName")
```


Example

```
// XMetaL Script Language JSCRIPT:
// Create a Para element node and insert
// it at the end of the document
var paraElem = ActiveDocument.createElement("Para");
var docElem = ActiveDocument.documentElement;
docElem.appendChild(paraElem);
```

createElementNS
Creates a DOMElement object with the name specified. This method is namespace aware. Use this method only with global (top-level) elements. Elements can also be inserted with `Selection.InsertElementNS`, `Selection.InsertElementWithRequiredNS`, and `Selection.InsertWithTemplateNS`. You can use these methods if you want to provide a local name (instead of a qualified name) for an element, or if you do not want to create a global element.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMElement object

Usage in JScript

```
Document_object.createElementNS("strNamespaceNameURI", "strQualifiedName");
```

`strQualifiedName` is in the form `prefix:localName`. The element is associated with the namespace named `strNamespaceNameURI`.

Usage in VBScript

```
Document_object.createElementNS("strNamespaceNameURI", "strQualifiedName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a Para element node and insert
// it at the end of the document
var paraElem =
ActiveDocument.createElementNS("http://www.mydomainname.com/Journalist",
 "myns:Para");
var docElem = ActiveDocument.documentElement;
docElem.appendChild(paraElem);
```

createElementWF
This method is reserved for internal use only.

createEntityReference
Creates a DOMEntityReference object corresponding to a reference to the specified entity. Entity references can also be inserted with `Selection.InsertEntity`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMEntityReference object

Usage in JScript

```
Document_object.createEntityReference("strName");
```

Usage in VBScript

```
Document_object.createEntityReference("strName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create an entity reference node and insert it
// at the end of the current container
var newER = ActiveDocument.createEntityReference("Acirc");
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
curNode.appendChild(newER);
```

CreateMark
This method is reserved for internal use only.

CreatePreviewFile
Saves the current document with a document type declaration that does not contain a system identifier; this format is suitable for previewing (or other processing) by an external processor. Returns the file path of the saved file. If you want to delete this file, you need to use the Windows FileSystemObject.

Applies to

XMetaL Author

Returns

String

Usage in JScript

```
var strPreviewFilePath = Document_object.CreatePreviewFile();
```

Usage in VBScript

```
dim strPreviewFilePath = Document_object.CreatePreviewFile
```

Example

```
// XMetaL Script Language JSCRIPT:
var tmpFile = ActiveDocument.CreatePreviewFile();
```

createProcessingInstruction
Creates a DOMProcessingInstruction node corresponding to a processing instruction with the specified strings. Processing instructions can also be inserted with `Selection.InsertProcessingInstruction`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMProcessingInstruction object

Usage in JScript

```
Document_object.createProcessingInstruction("strTarget", "strData");
```

Usage in VBScript

```
Document_object.createProcessingInstruction("strTarget", "strData")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a PI node and insert it at the end of the
// current container
var newPI = ActiveDocument.createProcessingInstruction(
"my_formatter", "endpage");
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
curNode.appendChild(newPI);
```

createTextNode

Creates a DOMText node containing the specified string.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMText object

Usage in JScript

```
Document_object.createTextNode("strText");
```

Usage in VBScript

```
Document_object.createTextNode("strText")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a new text node and insert it
// at the beginning of the current element
var strText = "Hello world.";
textNode = ActiveDocument.createTextNode(strText);
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
curNode.insertBefore(textNode,curNode.firstChild);
```

createTextNodeWF
This method is reserved for internal use only.

DeclareExternalEntity
Creates an external entity declaration in the document for the entity, public identifier, and system identifier specified. If a declaration for `strName` already exists, it is replaced. This method does not support SUBDOC SGML entities.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value.

Usage in JScript

```
Document_object.DeclareExternalEntity("strName", "strPublicId", "strSystemId");
```

Usage in VBScript

```
Document_object.DeclareExternalEntity("strName", "strPublicId", "strSystemId")
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.DeclareExternalEntity("FrontMatter", "-//local//Book front
 matter//EN", "book/front.xml");
```

DeclareGraphicEntity
Creates a graphic entity declaration in the document for the entity, public identifier, system identifier, notation, and type specified.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Description

The allowed values for `strType` are:

- sqDefaultEntity (0)=NDATA (default)
- sqCDataEntity (1)=CDATA
- sqSDataEntity (2)=SDATA

Usage in JScript

```
Document_object.DeclareGraphicEntity("strName", "strPublicId", "strSystemId", "strNotation",
 ["strType"]);
```


If a declaration for `strName` already exists, it is replaced.

Usage in VBScript

```
Document_object.DeclareGraphicEntity "strName", "strPublicId", "strSystemId", "strNotation",
 ["strType"]
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.DeclareGraphicEntity("Face101",
"-//local//Graphic #101//EN",
"book/graphics/face101.gif", "GIF");
```

DeclareNotation

Creates a notation declaration in the document and returns True if successful.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Document_object.DeclareNotation("strName", "strPublicId", "strSystemId");
```

If a declaration for `strName` already exists, it is replaced.

Usage in VBScript

```
Document_object.DeclareNotation "strName", "strPublicId", "strSystemId"
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.DeclareNotation("GIF",
"-//local//Graphic #101//EN",
"book/graphics/face101.gif");
```

DeclareTextEntity

Creates a text entity declaration in the document for the entity, content, and (optionally) type specified.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.DeclareTextEntity("strName","strContent", ["strType"]);
```

If a declaration for `strName` already exists, it is replaced.

The allowed values for `strType` are:

- sqDefaultEntity (0)=no type (default)
- sqCDataEntity (1)=CDATA
- sqSDataEntity (2)=SDATA

Usage in VBScript

```
Document_object.DeclareTextEntity("strName","strPublicId", ["strType"])
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.DeclareTextEntity("Product", "XMetaL");
```

DeleteEntityDeclaration

Deletes the declaration for entity specified, if it is declared in the document, and the document contains no references to it.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.DeleteEntityDeclaration("strName");
```

Usage in VBScript

```
Document_object.DeleteEntityDeclaration("strName")
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.DeleteEntityDeclaration("Product");
```

DeleteNotationDeclaration

Deletes the declaration for a specified notation, if it is declared in the document and the document contains no references to it. Returns True if deleted.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Document_object.DeleteNotationDeclaration("cgm");
```

Usage in VBScript

```
Document_object.DeleteNotationDeclaration("cgm")
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.DeleteNotationDeclaration("cgm");
```

detachTransclusion
This method is reserved for internal use only.

detachTransclusionEx
This method is reserved for internal use only.

DropNotAllowed
This method can be called only from within the On_Drag_Over_<format> event macro or from within the OnDragOver event (available in XMetaL XMAX only). When set, this changes the cursor to the 'drop not allowed' shape to provide the user with visual feedback about where objects can be dropped. In XMetaL Author, this method is identical to `Application.DropNotAllowed`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.DropNotAllowed();
```

Usage in VBScript

```
ActiveDocument.DropNotAllowed
```

Example

```
' XMetaL Script Language VBScript:

' Put this code in the On_Application_Open:
ActiveDocument.AcceptDropFormat "HTML Format", "HTMLFormat"
... 'Put this code in the On_Drop_HTMLFormat macro:
Set rng = Document.DropPoint
rng.PasteString "dropHTML"
rng.Select
... 'Put this code in the On_Drag_Over_HTMLFormat macro:
Set rng = Document.DropPoint
If rng.IsParentElement("Title") Then
ActiveDocument.Host.SetStatusText "Over the title element"
Document.DropNotAllowed
Else
```

```
ActiveDocument.Host.SetStatusText ""
End If
```

EmptyClipboard
Clears the Windows clipboard.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.EmptyClipboard();
```

Usage in VBScript

```
ActiveDocument.EmptyClipboard
```

Example

```
// XMetaL Script Language JSCRIPT:

if (ActiveDocument.Clipboard.HasText){
ActiveDocument.EmptyClipboard();
ActiveDocument.Alert("Cleared text from the clipboard.");
}
```

EndMutations
Tells XMetaL to end the previous group of document changes via API into a single undoable operation. Must be preceded with a `BeginMutations` call. The optional boolean parameter indicates if the command has caused any changes. EndMutations calls cannot be nested.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.EndMutations(true);
```

Usage in VBScript

```
ActiveDocument.EndMutations True
```


Example

```
// XMetaL XMAX example - assumes that myXMControl was previously
initialized

myXMControl.Document.BeginMutations();
myXMControl.Selection.InsertElement("p");
myXMControl.Selection.InsertElement("b");
myXMControl.Document.EndMutations(true);
```

exportTransclusion
This method is reserved for internal use only.

formatGraphicTable
Displays the table as if it had been created by the user; changes the appearance from tags to the usual view. XMetaL identifies tables in documents at the following times: during a load-time parse operation, upon insert of the table by user action. Only at those times are tables recognized as such and rendered as tables in the editor. Therefore, if you insert a table using the element list or through scripting, XMetaL does not display the table in its usual way; instead, it displays the inserted table as a set of tags. If you want the table to appear as if it had been created by the user, then you must use the formatGraphicTable method to change the appearance from tags to the usual view.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.formatGraphicTable(nodeTbleNode);
```

`nodeTblNode` is the DOM Node of the table node of the inserted table you want to format. This parameter is required and has no default value.

Usage in VBScript

```
ActiveDocument.formatGraphicTable(nodeTbleNode)
```

Example

```
// XMetaL Script Language JSCRIPT:
// First, create a table
var numcols = 2;
var numrows = 3;

Selection.InsertElement("TABLE");
Selection.InsertElement("TBODY");

for (var r=1; r<numrows+1; r++)//loop for rows
{
Selection.InsertElement("TR");

for (var c=1; c<numcols+1; c++)//loop for columns
{
Selection.InsertElement("TD");
```

```
Selection.MoveRight();
} //end loop for columns

Selection.MoveRight();
}//end loop for rows

var nd = Selection.ContainerNode;
nd = nd.parentNode;

ActiveDocument.formatGraphicTable(nd);
```

getElementById
Returns a DOMElement, corresponding to the element that has an ID attribute with the value specified. (ID attributes are those that are declared to have the ID type in the document's DTD or schema.) Returns null if no such element exists in the document. ID value matching is case-sensitive in XML documents; in SGML documents matches are case-insensitive if the NAMECASE parameter is set to YES (the default) in the SGML declaration, and case-sensitive if NAMECASE is set to NO.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMElement object

Usage in JScript

```
Document_object.getElementById("strVal");
```

Usage in VBScript

```
Document_object.getElementById("strVal")
```

Example

```
// Copy into On_Mouse_Over" event macro
// XMetaL Script Language JSCRIPT:
// Get the element whose ID attribute is the
// same as the "Linkend" attribute of the
// element that the mouse is over.

var Linkend = curNode.getAttribute("Linkend");
var targNode = ActiveDocument.getElementById(Linkend);

// If an element is found, and it is a "Title"
// element, display its contents in the status bar
if (targNode) {

if (targNode.nodeName=="Title") {
rng.SelectNodeContents(targNode);
ActiveDocument.Host.SetStatusText(rng.Text);
}
}
```

getElementsByTagName
Returns a DOMNodeList of DOMElement objects, representing all elements in the document with the element name specified. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
Document_object.getElementsByTagName("strTagName");
```

The special `strTagName` "*" matches all elements.

Usage in VBScript

```
Document_object.getElementsByTagName("strTagName")
```

Example

```
// XMetaL Script Language JSCRIPT:

var elemList;
var allElemList;
// Get all "Title" elements
elemList = ActiveDocument.getElementsByTagName("Title");
ActiveDocument.Host.Alert(elemList.length);
// Get all elements in the document
allElemList = ActiveDocument.getElementsByTagName("*");
ActiveDocument.Host.Alert(allElemList.length);
```

getElementsByTagNameEx
Returns either a live, snapshot or forward-iterator-only DOMNodeList containing all the descendant elements of this document that have the name 'tagname'.

The special tagname '*' matches all elements. The "mode" parameter affects the DOMNodeList behavior as follows below:

- mode == 3 returns a "snapshot" list; document mutations will not cause list to change though nodes may become invalid.
- mode == 4 returns a "live" list; document mutations can cause list to change.

mode == 5 returns a "forward-iterator-only" list; special case of "snapshot" list. DOMNodeList::item(0) method gets first in-sequence item and moves iterator to the head of the list, whereby calling DOMNodeList::item() method gets next in-sequence item regardless of index parameter value (must be different from 0) provided.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
Document_object.getElementsByTagNameEx("tagname");
```

Usage in VBScript

```
Document_object.getElementsByTagNameEx("tagname")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Using the CamerasInFocus sample in XMetaL Author,
// copy and paste this script example to get a list of titles.
// The XPath string searches for a list of all "Title" nodes.
// "mode" 2 parameter forces to  return "forward-iterator-only" "titles" list;
// calling DOMNodeList::item(0) method moves "forward-iterator-only" to the first item.
// calling DOMNodeList::item(1) method gets the next item in sequence.

var tableOfContents = "";
var currTitleNumber = 1;

var titles = ActiveDocument.getElementsByTagNameEx("Title",5);
// Select first element from the titles list
var title = titles.item(0);
while(title){
            tableOfContents += currTitleNumber + ". " + title.childNodes.item(0).data +
"\n";
            currTitleNumber++;
            title = titles.item(1); // Select first element from the titles list
}
Application.Alert(tableOfContents);
```

getElementsByTagNameNS

Returns a DOMNodeList of DOMElement objects, representing all elements in the document with the local element name specified in the namespace specified. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
Document_object.getElementsByTagNameNS("strNamespaceNameURI", "strLocalName");
```

The special `strLocalName` "*" matches all elements in the namespace `strNamespaceNameURI`.

Usage in VBScript

```
Document_object.getElementsByTagNameNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:

var elemList;
var allElemList;
// Get all "Title" elements in the http://www.mydomainname.com/Journalist
 namespace
elemList=ActiveDocument.getElementsByTagNameNS("http://www.mydomainname.com/Journalist",
 "Title");
ActiveDocument.Host.Alert(elemList.length);
```



```
// Get all elements in the http://www.mydomainname.com/Journalist namespace
allElemList=ActiveDocument.getElementsByTagNameNS("http://www.mydomainname.com/Journalist",
 "*");
ActiveDocument.Host.Alert(allElemList.length);
```

getNodesByXPathEx
Returns either a live, snapshot or forward-iterator-only DOMNodeList containing all the nodes of this document that match up with the provided XPath expression.

The "mode" parameter affects the DOMNodeList behavior as follows:

- mode == 3 returns a "snapshot" list; document mutations will not cause the list to change though nodes may become invalid.
- mode == 4 returns a "live" list; document mutations can cause the list to change.

mode == 5 returns a "forward-iterator-only" list; special case of "snapshot" list. DOMNodeList::item(0) method gets first in-sequence item and moves the iterator to the head of the list, whereby calling DOMNodeList::item() method gets next in-sequence item regardless of index parameter value (must be different from 0) provided.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
Document_object.getNodesByXPathEx(strXPathExpression);
```

Usage in VBScript

```
Document_object.getNodesByXPathEx(strXPathExpression)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Using the CamerasInFocus sample in XMetaL Author,
// copy and paste this script example to get a list of titles.
// The XPath string searches for a list of "Title" nodes that has
// a "*/Sect1" parent with the DOMDocument node as the context node.
// "mode" 2 parameter forces to  return "forward-iterator-only" "titles" list;
// calling DOMNodeList::item(0) method moves "forward-iterator-only" to the first item.
// calling DOMNodeList::item(1) method gets the next item in the sequence.

var tableOfContents = "";
var currTitleNumber = 1;

var titles = ActiveDocument.getNodesByXPathEx("*/Sect1/Title",5);
// Select first element from the titles list
var title = titles.item(0);
while(title){
            tableOfContents += currTitleNumber + ". " + title.childNodes.item(0).data +
"\n";
            currTitleNumber++;
            title = titles.item(1); // Select first element from the titles list
}
Application.Alert(tableOfContents);
```

GetNodeState
Returns True if the specified flag is set for the DOMNode specified. If certain types of nodes change, one or more of the following flags are set: ContentInserted, ContentDeleted, AttributesChanged, NewNode. This mechanism applies to the following nodes: DOMElement, DOMComment, DOMCDATASection, DOMProcessingInstruction, DOMEntityReference, DOMCharacterReference. `GetNodeState` returns True if the flag specified is set for the node specified. Flags can be turned off using `Document.ClearNodeChangedStates` and `Document.ClearAllChangedStates`

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Document_object.GetNodeState("strFlag", nodeNode)
```

Usage in VBScript

```
Document_object.GetNodeState("strFlag", nodeNode)
```

Example

```
// XMetaL Script Language JSCRIPT:

docElem = ActiveDocument.documentElement;

if (ActiveDocument.GetNodeState("AttributesChanged", docElem)) {
ActiveDocument.Host.Alert("Attributes changed in top-level tag.");
}
```

GetNodeStatebyKey
Returns True if the flag specified is set for the DOMNode specified. If certain types of nodes change, one or more of the following flags are set: ContentInserted, ContentDeleted, AttributesChanged, NewNode. This mechanism applies to the following nodes: DOMElement, DOMComment, DOMCDATASection, DOMProcessingInstruction, DOMEntityReference, DOMCharacterReference. This method returns True if the flag specified is set for the node specified. Flags can be turned off using `Document.ClearNodeChangedStatesbyKey` and `Document.ClearAllChangedStatesbyKey`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Document_object.GetNodeStatebyKey("strKey", "strFlag", nodeNode)
```

`strKey` indicates the key to search on.

Usage in VBScript

```
Document_object.GetNodeStatebyKey("strKey", "strFlag", nodeNode)
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set nds = ActiveDocument.ChangedNodesbyKey("abc")
MsgBox(nds.length)
i = 0
Set changenode = nds.item(i)

While Not changenode Is Nothing
MsgBox(changenode.nodeName)
MsgBox("NewNode: " + cstr(ActiveDocument.GetNodeStatebyKey("abc",
"NewNode", changenode)))
MsgBox("ContentInserted: " + cstr(ActiveDocument.GetNodeStatebyKey("abc",
 "ContentInserted", changenode)))
MsgBox("ContentDeleted: " + cstr(ActiveDocument.GetNodeStatebyKey("abc",
 "ContentDeleted", changenode)))
MsgBox("AttributesChanged: " + cstr(ActiveDocument.GetNodeStatebyKey("abc",
 "AttributesChanged", changenode)))
i = i+1
Set changenode = nds.item(i)
Wend
```

GetRenderedContent

Returns the rendered content (content displayed instead of the actual content) for the specified DOMNode.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
Document_object.GetRenderedContent(nodeNode);
```

Usage in VBScript

```
Document_object.GetRenderedContent(nodeNode)
```

Example

```
// XMetaL Script Language JSCRIPT:
var curNode = Selection.ContainerNode;
var rText = ActiveDocument.GetRenderedContent(curNode);
```

GetSpellingResult

Returns a semicolon-separated list of suggested spellings for the current misspelled word in the selection. The mode parameter must be 0.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
var strSemicolonListOfSuggestedWords = Document_object.GetSpellingResult(0);
```

Example

```
// XMetaL Script Language JSCRIPT:

var strSemicolonListOfSuggestedWords = ActiveDocument.GetSpellingResult(0);
ActiveDocument.Host.Alert(strSemicolonListOfSuggestedWords);
```

getTextEntityReplacementText
This method is reserved for internal use only.

GetUserString
This method is reserved for internal use only.

GotoNextChange
Selects the next tracked revision in the document. If no tracked revision is found, XMetaL XMAX will fire the OnMessage event with a message string asking if the user would like to start searching from the top of the document. XMetaL Author displays this message to the user. Normally, this method is used to handle actions in the user interface, such as a click on a toolbar button. Do not use this method to programmatically traverse the list of revision marks.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.GotoNextChange();
```

Usage in VBScript

```
ActiveDocument.GotoNextChange
```

Example

```
' XMetaL Script Language VBScript:
' Assumes the existence of a NextButton in the user interface
' Goes to the next tracked revision, if possible

Sub NextButton_Click handles NextButton.Click
ActiveDocument.GotoNextChange
End Sub
```

GotoPrevChange
Selects the previous tracked revision in the document. If no tracked revision is found, XMetaL XMAX will fire the OnMessage event with a message string asking if the user would like to start searching from the bottom

of the document. XMetaL Author displays this message to the user. Normally, this method is used to handle actions in the user interface, such as a click on a toolbar button. Do not use this method to programmatically traverse the list of revision marks.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.GotoPrevChange();
```

Usage in VBScript

```
ActiveDocument.GotoPrevChange
```

Example

```
' XMetaL Script Language VBScript:
' Assumes the existence of a PrevButton in the user interface
' Goes to the previous tracked revision, if possible

Sub PrevButton_Click handles PrevButton.Click
ActiveDocument.GotoPrevChange
End Sub
```

HideEditButtonInAttributeInspector

Configures the Attribute Inspector to hide the custom-edit button whenever the named attribute-type has focus within the grid control. By default, XMetaL hides the custom-edit button for all attribute types on a document. This method is safe to call during or after the On_Document_Open_Complete event-macro.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.HideEditButtonInAttributeInspector("strAttrName");
```

`strAttrName` is a qualified name, in the form `namespace prefix:local name`. If you do not want to specify a namespace, omit the namespace prefix and colon.

Usage in VBScript

```
Document_object.HideEditButtonInAttributeInspector("strAttrName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Hide the edit button in the Attribute Inspector for the audience
attribute
ActiveDocument.HideEditButtonInAttributeInspector("audience");
```

HideInAttributeInspector
Prevents the attribute specified from being displayed in the Attribute Inspector.

Applies to

XMetaL Author

Returns

No return value.

Usage in JScript

```
Document_object.HideInAttributeInspector("strAttrName", ["strElemName"]);
```

If `strElemName` is specified, this applies only to the `strAttrName` attribute of that element; if `strElemName` is omitted or null, the `strAttrName` attribute of all elements of which it is an attribute is hidden. Both `strAttrName` and `strElemName` are qualified names, in the form `namespace prefix:local name`. If you do not want to specify a namespace, omit the namespace prefix and colon.

Usage in VBScript

```
Document_object.HideInAttributeInspector("strAttrName", ["strElemName"])
```

Example

```
// XMetaL Script Language JSCRIPT:
// Hide the Id attribute of all elements
ActiveDocument.HideInAttributeInspector("Id");
// Hide the Status attribute of the Para element
ActiveDocument.HideInAttributeInspector("Status", "Para");
```

IgnoreWordSpelling
Returns True if the word under the cursor was successfully flagged to be ignored for the remainder of the document editing session. Equivalent to clicking "Skip All" in the SpellChecker user interface.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
var succeeded = Document_object.IgnoreWordSpelling();
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.IgnoreWordSpelling(); // Red-squiggly line goes away for this edit session
```

importNode
This method is reserved for internal use only.

includeTransclusionEx
This method is reserved for internal use only.

Layout
Applies text layout to the entire document. The text layout is specified in the Text Layout dialog box, which is accessed through the customization file editor in XMetaL Developer. This method can be used only if the document is displayed in Plain Text view. This method applies the text layout and returns True if the document is in Plain Text view, and does nothing and returns False otherwise.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Document_object.Layout();
```

Usage in VBScript

```
Document_object.Layout
```

Example

```
// XMetaL Script Language JSCRIPT:
// Format the entire document

if (ActiveDocument.ViewType == 2) {
ActiveDocument.Layout();
}
```

MoveDropPoint
Moves the DropPoint to the insertion point specified by range. This method performs the same action as `Application.DropPoint` (available in XMetaL Author only).

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.MoveDropPoint(myRange);
```

The `range` passed to this method is an insertion point that specifies where the drop or paste occurs. `Range` must be within the current document. If it is not an insertion point, the range is collapsed to the right, and the collapsed range is used as the new drop point.

Usage in VBScript

```
ActiveDocument.MoveDropPoint myRange
```

Example

```
' XMetaL Script Language VBScript:
' Set the drop point to the right of the
' user-selected text, and insert an image. set sel=Selection
ActiveDocument.MoveDropPoint sel
Set rng = ActiveDocument.DropPoint
' Drop an image named filename
' note that filename must be defined elsewhere
rng.InsertImage(filename)
```

PostSetFocus

This method can be called only with the value of 1 passed to it. When it is called, focus switches to the XMetaL Author editing window after the script containing this method has completed executing.

Applies to

XMetaL XMAX

Returns

N/A

Usage in JScript

```
ActiveDocument.PostSetFocus(1);
```

Usage in VBScript

```
ActiveDocument.PostSetFocus(1)
```

Example

```
// XMetaL Script Language JSCRIPT:
// First, make sure the editing window does not have focus
ResourceManager.Visible=1;
ResourceManager.SelectTab("Desktop");

// Now let Author know that we want focus on the
// editing window AFTER this script finishes running
ActiveDocument.PostSetFocus(1);
ActiveDocument.Alert("The editing window should have focus after this
script is done.");

// ... Some more code ...
```


PreviewInBrowser
Previews the document in the browser specified.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.PreviewInBrowser("strBrws");
```

Usage in VBScript

```
Document_object.PreviewInBrowser "strBrws"
```

Example

```
// XMetaL Script Language JSCRIPT:
var brws = "c:\\Program Files\\Internet Explorer\\iexplore.exe";
ActiveDocument.PreviewInBrowser(brws);
```

QueryAttributes
This method is reserved for internal use only.

Redo
Re-does the specified number of previous undone actions, if possible. The default is to redo the last undone action, if possible. Returns True if the redo succeeds. When this method is called from a macro or script located in an application customization or document customization, it can only redo actions that were undone in that macro or script.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Document_object.Redo([intNumberOfTimes]);
```

Usage in VBScript

```
Document_object.Redo([intNumberOfTimes])
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.Redo(); //redo the last undone action
```

RefreshCssStyle

Refreshes the XMetaL display by reloading the CSS (cascading style sheet) file linked to the document and fires both the On_View_RefreshCssStyle and On_StructureView_RefreshCssStyle event macros.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.RefreshCssStyle();
```

Usage in VBScript

```
Document_object.RefreshCssStyle
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.RefreshCssStyle();
```

RefreshCssStyleToAppend

Refreshes the XMetaL display by loading additional CSS styles. Valid only during On_View_RefreshCssStyle or On_StructureView_RefreshCssStyle event macro.

Applies to

XMetaL Author and XMetaL XMAX

Access

N/A

Returns

N/A

Usage in JScript

```
ActiveDocument.RefreshCssStyleDoc.RefreshCssStyleToAppend('cssStr');
```

Note: `cssStr` must not contain `@import` statements.

Usage in VBScript

```
ActiveDocument.RefreshCssStyleDoc.RefreshCssStyleToAppend 'cssStr'
```


Example

```
ActiveDocument.RefreshCssStyleDoc.RefreshCssStyleToAppend('[class~=\"topic/p\"]
 { color: #3399CC; }');
```

RejectAllChanges
Rejects all tracked changes in the document. The changes are deleted, and the original text is restored. Normally, this method is used to handle events in the user interface, such as a click on a toolbar button.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.RejectAllChanges();
```

Usage in VBScript

```
ActiveDocument.RejectAllChanges
```

Example

```
' XMetaL Script Language VBScript
' XMetaL XMAX example -- assumes myXMControl was previously initialized
' Rejects all changes when a button named RejectAllButton is clicked
' in the user interface

Sub RejectAllButton_Click Handles RejectAllButton.Clicked
If Not (myXMControl.Document Is Nothing) Then
myXMControl.Document.RejectAllChanges
End If
End Sub
```

RejectChange
Rejects the currently selected tracked change. The change is deleted, and the original text is restored. Normally, this method is used to handle user interface events, such as a click on a toolbar button.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.RejectChange();
```

Usage in VBScript

```
ActiveDocument.RejectChange
```

Example

```
' XMetaL Script Language VBScript:
' XMetaL XMAX example -- assumes myXMControl was previously initialized
' Rejects the currently selected change when a button named
' RejectButton is clicked in the user interface

Sub RejectButton_Click Handles RejectButton.Click
If (TypeName(myXMControl.Document)<>"Nothing") And
myXMControl.Document.CanAcceptOrRejectChange) Then
myXMControl.Document.RejectChange
End If
End Sub
```

Reload
Reloads the specified document from the disk. The document must be open. This method does not check if the document has been changed before reloading.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.Reload();
```

Usage in VBScript

```
Document_object.Reload
```

Example

```
// XMetaL Script Language JSCRIPT:
// doc must be initialized as a Document object before using this code

if (Application.IsDocumentOpen(doc)) {
doc.Reload();
}
```

ReplaceWord
Replaces the word under the cursor that is misspelt, with the given word provided.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
var succeeded = Document_object.ReplaceWord("New");
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.ReplaceWord("Correct");
```

Save

Saves the specified document, if possible. In XMetaL XMAX, you can save only a document opened with LoadFromFile using this method. You can use the Saved property to determine if the operation was successful.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.Save();
```

Usage in VBScript

```
ActiveDocument.Save
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Saves the active document

ActiveDocument.Save
```

SaveAs

Saves the document, if possible, using the specified file name.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.SaveAs([strFileName], [boolAddToRecentFiles]);
```

In XMetaL Author, if `strFileName` is not specified, the **Save As** dialog box appears. In XMetaL XMAX, `SaveAs` will fail silently if `strFileName` is not specified, or if the document cannot be saved with the name `strFileName` for any reason. You can check the Saved property to see if the `SaveAs` operation was

successful. `boolAddToRecentFiles`, an optional parameter used only by XMetaL Author, determines whether the file is added to the recent files area of the **File** menu. Examples of strings you can use for `strFileName`: `C:\my.xml`, `\\my_network_computer\my.xml`, `http://www.mydomainname.com/my.xml` (Includes WebDAV), `https://www.mydomainname.com/my.xml` (Includes WebDAV). Loading a XAC using HTTPS only works if the file is on the same HTTPS server.

Usage in VBScript

```
Document_object.SaveAs [strFileName], [boolAddToRecentFiles]
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Saves the active document as "myFile.xml"
' and adds a reference to it in the Recent Files menu.
' This script will work in both XMetaL Author and XMetaL XMAX,
' but XMetaL XMAX does not add any references to any menus.

ActiveDocument.SaveAs "myFile.xml", true
```

ScrollToSelection

Scrolls the screen to the current selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.ScrollToSelection();
```

Usage in VBScript

```
Document_object.ScrollToSelection
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Scroll the screen to the current selection

ActiveDocument.ScrollToSelection
```

SetCTMRulesService

This method is reserved for internal use only.

SetImageService

This method is reserved for internal use only.

SetMarkPresentationService

This method is reserved for internal use only.

SetSpellCheckerService
This method is reserved for internal use only.

SetRenderedContent
Enables you to display the text specified (rendered content) instead of the actual DOMNode content. For example, an element might contain a date in DD-MM-YYYY format, but display the equivalent date in MM/DD/YYYY format.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.SetRenderedContent(nodeNode,"strText")
```

Usage in VBScript

```
Document_object.SetRenderedContent(nodeNode,"strText")
```

Example

```
// XMetaL Script Language JSCRIPT:
var rng = ActiveDocument.Range;
rng.SelectContainerContents();
var Date1 = rng.Text;
var date1Fmt = /\s*([0-9]{2})-([0-9]{2})-([0-9]{4})\s*/;
var result = Date1.match(date1Fmt);

if (result) {
var Date2 = result[2] + "/" + result[1] + "/" + result[3];
var curNode = rng.ContainerNode;
ActiveDocument.SetRenderedContent(curNode,Date2);
}
```

SetUserString
This method is reserved for internal use only.

ShowEditButtonInAttributeInspector
Configures the Attribute Inspector to show the custom-edit button whenever the named attribute-type has focus within the grid control. This method is limited to CDATA attribute types. This method is safe to call during or after the On_Document_Open_Complete event-macro.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.ShowEditButtonInAttributeInspector("strAttrName");
```

`strAttrName` is a qualified name, in the form `namespace prefix:local name`. If you do not want to specify a namespace, omit the namespace prefix and colon.

Usage in VBScript

```
Document_object.ShowEditButtonInAttributeInspector("strAttrName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Show the edit button in the Attribute Inspector for the audience
attribute
ActiveDocument.ShowEditButtonInAttributeInspector("audience");
```

showHideTransclusion
This method is reserved for internal use only.

showHideTransclusionEx
This method is reserved for internal use only.

ShowInAttributeInspector
Undoes the effect of the HideInAttributeInspector method and enables the attribute specified to be displayed in the Attribute Inspector. Since all attributes are displayed in the Attribute Inspector by default, this method is needed only to undo a previous use of HideInAttributeInspector.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.ShowInAttributeInspector("strAttrName", ["strElemName"]);
```

If `strElemName` is specified, this applies only to the `strAttrName` attribute of that element; if `strElemName` is omitted or null, the `strAttrName` attribute of all elements of which it is an attribute is shown.Both `strAttrName` and `strElemName` are qualified names, in the form `namespace prefix:local name`. If you do not want to specify a namespace, omit the namespace prefix and colon.

Usage in VBScript

```
Document_object.ShowInAttributeInspector "strAttrName", ["strElemName"]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Show the Id attribute of all elements
ActiveDocument.ShowInAttributeInspector("Id");
// Show the Status attribute of the Para element
ActiveDocument.ShowInAttributeInspector("Status", "Para");
```

Undo

Undoes the the specified number of previous change(s) to the document. The default is to undo the last change, if possible. Returns True if the undo succeeds. When this method is called from a macro or script located in an application customization or document customization, it can undo only actions performed in that macro or script.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Document_object.Undo([intNumberOfTimes]);
```

Usage in VBScript

```
Document_object.Undo([intNumberOfTimes])
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.Undo(); //undo the last action
```

UndoClear

Clears the document's undo stack, making it appear that the document has not been changed since it was last saved.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.UndoClear();
```

Usage in VBScript

```
Document_object.UndoClear
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.UndoClear();
```

UniqueAttributeValue

Returns a string that is not used as the value of any specified attribute in the active document.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
Document_object.UniqueAttributeValue("strAttrName", "strPrefix", [intNumber]);
```

`strAttributeName` is a qualified name in the form `namespace prefix:local name`. If you do not want to specify a namespace, omit the namespace prefix and colon. If `strPrefix` is such a string, it is returned. If not, the number 1 is appended to `strPrefix` and the test is repeated. The number is incremented until an unused value is found. If `intNumber` is specified then that value is used as the starting number instead of 1. A maximum of 1000 tests are performed. If an unused value is not found, the null string is returned.

Usage in VBScript

```
Document_object.UniqueAttributeValue("strAttrName", "strPrefix", [intNumber])
```

Example

```
// XMetaL Script Language JSCRIPT:
// Check whether a particular value is already used as
// an ID attribute; if not, use it. var Unique;
var attName = "Id";
var attVal = "xyzzy";
Unique=ActiveDocument.UniqueAttributeValue(attName,attVal);
ActiveDocument.Host.Alert(Unique);

if (attVal==Unique){
Selection.ContainerAttribute(attName)=attVal;
}
```

The following code demonstrates how to work around the upper limit of 1000 searches:

```
' XMetaL Script Language VBSCRIPT:

For cnt = 0 To 10000 Step 1000
str = ActiveDocument.UniqueAttributeValue("Id", "a", cnt)

if not str = "" Then
Exit For
End if
Next

if not str = "" Then
MsgBox("Found unique value = """ + str + """")
else
MsgBox("Unable to find unique value")
end if
```

Validate

Validates the document according to its DTD, schema, or rules file. A message is displayed to the user.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Document_object.Validate();
```

Usage in VBScript

```
Document_object.Validate
```

Example

```
// XMetaL Script Language JSCRIPT:
// A script that toggles document rule-checking

if (ActiveDocument.RulesChecking) {
var response = ActiveDocument.Host.MessageBox("Running macros without
rules checking may have unpredictable results.\nDo you want to proceed?",
 32+1, "Toggle Rules Checking");
if (response == 1)
ActiveDocument.RulesChecking = false;
}
else {
ActiveDocument.RulesChecking = true;
if (!ActiveDocument.RulesChecking) {
ActiveDocument.Host.Alert("Could not turn Rules Checking on due to
validation errors.");
ActiveDocument.Validate();
}
}
```

DocumentHost

The DocumentHost interface provides a way to communicate with the host application of the document. When working with XMetaL Author, XMetaL Author will always be the host application. When working with XMetaL XMAX, the host application could be Internet Explorer or a custom application.

This interface allows you to write portable document customizations. Use DocumentHost methods when you want to create a document customization that will work with both XMetaL Author and XMetaL XMAX. The properties and methods are similiar to the properties and methods of the same name in the Application interface.

You can acquire the DocumentHost interface with the `Document.Host` property.

Properties

DisplayAlerts

Enables or disables the displaying of alerts that can be issued by various operations, for example, output from `DocumentHost.Alert`, find string not found (XMetaL Author only), attempt to save an invalid file (XMetaL Author only), `DocumentHost.Beep` method. In XMetaL Author, this property is identical to `Application.DisplayAlerts`.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Integer

Usage in JScript

```
ActiveDocument.Host.DisplayAlerts = -1;
var myInteger = ActiveDocument.Host.DisplayAlerts;
```

Allowed values are:

- sqAlertsNone (0)= Turn off alerts
- sqAlertsAll (-1)= Turn on alerts

Usage in VBScript

```
ActiveDocument.Host.DisplayAlerts = -1;
var myInteger = ActiveDocument.Host.DisplayAlerts;
```

Example

```
// XMetaL Script Language JSCRIPT:
// Turn off alerts
ActiveDocument.Host.DisplayAlerts = sqAlertsNone;
// Next two lines should display nothing
ActiveDocument.Host.Alert("Should not see this message!");
Selection.Find.Execute("xyzzyfvvkkp");
// Turn alerts back on
ActiveDocument.Host.DisplayAlerts = sqAlertsAll;
```

Methods

Alert

Displays an alert dialog box with the specified message and title (if supplied). Output from this method can be suppressed with `DocumentHost.DisplayAlerts`. The OnMessage event fires in XMetaL XMAX after a call to this method. You can use this event to apply a custom appearance to alert boxes. Do not call `DocumentHost.Alert` in the On_Document_Activate or On_Document_Deactivate event macros.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.Host.Alert("strMessage", ["strTitle"]);
```


Usage in VBScript

```
ActiveDocument.Host.Alert "strMessage", ["strTitle"]
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.Host.Alert("Your Message Here", "Your Title");
```

Beep

Makes a 'beep' sound. This can be suppressed using `DocumentHost.DisplayAlerts`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.Host.Beep();
```

Usage in VBScript

```
ActiveDocument.Host.Beep
```

Example

```
// XMetaL Script Language JSCRIPT:
// Add this code to the SaveAsRTF macro
// This will warn you if you try to run the macro while the active document
 is invalid
if (!ActiveDocument.IsValid) {
ActiveDocument.Host.Beep();
ActiveDocument.Host.Alert("You should not try to run this macro while the
 active document is invalid!");
}
```

Confirm

Displays a confirmation dialog box, displaying the message and title specified and containing **OK** and **Cancel** buttons. Clicking these buttons returns True and False, respectively. The OnMessage event fires in XMetaL XMAX after a call to this method. You can use this event to apply a custom appearance to confirmation boxes. Do not call `DocumentHost.Confirm` in the On_Document_Activate or On_Document_Deactivate event macros.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
var myBoolean = ActiveDocument.Host.Confirm("strMessage", ["strTitle"]);
```

Usage in VBScript

```
dim myBoolean = ActiveDocument.Host.Confirm("strMessage", ["strTitle"])
```

Example

```
// XMetaL Script Language JSCRIPT:

var closeFile = true;
var titleTxt = "";
var rng = ActiveDocument.Range;
var titleFound = rng.Find.Execute("<Title>");

if (titleFound) {
rng.SelectContainerContents();
titleTxt = rng.Text;
}
var dlgMsg = "Document has no title. Close anyway?";
var dlgTitle = "XML Editor";
if (!titleTxt) {
closeFile = ActiveDocument.Host.Confirm(dlgMsg,dlgTitle);
}
if (closeFile) {
ActiveDocument.Close();
}
```

CreateFormDlg

Launches the form specified, which contains the full path name and filename. You can access the form's interface through the returned form object. Do not call `Document.CreateFormDlg` in the On_Document_Activate or On_Document_Deactivate event macros. If you are working with XMetaL Author only, you can use the `Application.CreateFormDlg` method.

Applies to

XMetaL Author and XMetaL XMAX

Returns

Boolean

Usage in JScript

```
var myBoolean = Document_object.CreateFormDlg("strFileName", [nodeDOMNode]);
```

The optional parameter `nodeDOMNode` indicates the DOMNode to map form data to when the form is closed.

Usage in VBScript

```
myBoolean = Document_object.CreateFormDlg("strFileName", [nodeDOMNode])
```

Example

```
// XMetaL Script Language JSCRIPT:
// Call an instance of myform.xft
var dlg=ActiveDocument.Host.CreateFormDlg("C:\\myform.xft");
```

```
dlg.DoModal();
dlg=null;
```

DisableMacro
Disables the specified macro by disabling its shortcut key, and graying out its toolbar button and menu item (if any).

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Description

In XMetaL XMAX, you should disable the macro's toolbar button and menu item in the container application for XMetaL XMAX. In XMetaL Author, the macro can still be run from the Macros dialog box and Macros toolbar. This method is intended to be used only in the On_Update_UI event macro and OnUpdateUI event. Whenever On_Update_UI is called, any macros that were disabled the last time On_Update_UI was executed are automatically enabled again.

In XMetaL Author, a macro that can be disabled in this way should check for conditions that may make it undesirable to be run from the Macros dialog box or Macros toolbar while it is disabled. For example, if the macro should not be run in this way, it can issue a 'beep' using `DocumentHost.Beep`. If you never want to run it from the Macros dialog box or Macros toolbar, set the macro's hide attribute to True (you should do this *after* you associate the macro with a toolbar and/or menu item). In XMetaL Author, this method performs the same action as `Application.DisableMacro`.

Usage in JScript

```
ActiveDocument.Host.DisableMacro("strMacroName");
```

Usage in VBScript

```
ActiveDocument.Host.DisableMacro strMacroName
```

Example

```
// XMetaL Script Language JSCRIPT:

if (!ActiveDocument.IsValid) {
ActiveDocument.Host.DisableMacro("SaveAsRTF");
}
```

MessageBox
Displays a message box containing the text specified and, optionally, a title. The OnMessage event fires in XMetaL XMAX after a call to this method. You can use this event to apply a custom appearance to message boxes. Do not call `DocumentHost.MessageBox` in the On_Document_Activate or On_Document_Deactivate event macros.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Long

Usage in JScript

```
var myLong = ActiveDocument.Host.MessageBox(strMessage, longButtons, [strTitle]);
```

The button configuration is determined by the `longButtons` value. The return value depends on which button the user clicks.

longButtons values

The first group of values (0 to 5) describes the number and type of buttons displayed in the dialog box; the second group (16, 32, 48, 64) describes the icon style; the third group (0,256,512) determines which button is the default; and the fourth group (0,4096) determines the modality of the message box. When adding numbers to create a final value for the argument buttons, use only one number from each group.

Value	Description
0	Display OK button only
1	Display OK and Cancel buttons
2	Display Abort, Retry, and Ignore buttons
3	Display Yes, No, and Cancel buttons
4	Display Yes and No buttons
5	Display Retry and Cancel buttons
16	Display Critical Message icon
32	Display Warning Query icon
48	Display Warning Message icon
64	Display Information Message icon
0	First button is default
256	Second button is default
512	Third button is default
0	Application modal; the user must respond to the message box before continuing work in the current application.
4096	System modal; all applications are suspended until the user responds to the message box.

Message box return values

Value	Button Pressed
1	OK
2	Cancel
3	Abort

4	Retry
5	Ignore
6	Yes
7	No

Usage in VBScript

```
dim myLong = ActiveDocument.Host.MessageBox(strMessage, longButtons, [strTitle]);
```

Example

```
// XMetaL Script Language JSCRIPT:

// Make default response "No"
var response = 7;
var rng = ActiveDocument.Range;
var titleFound = rng.Find.Execute("<Title>");

if (!titleFound) {
var dlgMsg = "Document has no title. Insert one?";
var dlgTitle = "XML Editor";
// Display Ok, No, and Cancel buttons (3)
// and Warning Query icon (32)
var dlgConfig = 35;
response = ActiveDocument.Host.MessageBox(dlgMsg,dlgConfig,dlgTitle);
}
if (response==6) { // User clicked on [Yes]
rng.MoveToDocumentStart();
rng.FindInsertLocation("Title");
rng.InsertElement("Title");
ActiveDocument.Close();
} else if (response==7) { // User clicked on [No]
ActiveDocument.Close();
}
// Else response == 2 ([Cancel]): do nothing
```

NoticeBox
Displays a notice box containing the specified text. The OnMessage event fires in XMetaL XMAX after a call to this method. You can use this event to apply a custom appearance to notice boxes. Do not call `DocumentHost.NoticeBox` in the On_Document_Activate or On_Document_Deactivate event macros.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Long

Usage in JScript

```
var myLong = ActiveDocument.Host.NoticeBox("strMessage", "strBut1", ["strBut2"],
["strBut3"], ["strTitle"]);
```

Up to three buttons, with labels `strBut1`, `strBut2`, and `strBut3` can be specified. The return values are 1, 2, or 3, corresponding to the button clicked by the user. The first button is the initial default. If `strBut3` is omitted then a **Cancel** button with return value 3 is displayed. Pressing Esc or Alt+F4 always returns 3.

Usage in VBScript

```
dim myLong = ActiveDocument.Host.NoticeBox("strMessage", "strBut1", ["strBut2"],
["strBut3"], ["strTitle"])
```

Example

```
// XMetaL Script Language JSCRIPT:
var ans = ActiveDocument.Host.NoticeBox("myMessage", "Yes", "No", "Abort",
 "Proceed?");
ActiveDocument.Host.Alert(ans);
```

PathToURL

Returns a URL. In XMetaL Author, this method performs the same action as `Application.PathToURL`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
var myString = ActiveDocument.Host.PathToURL("strPath", ["strBase"]);
```

If only `strPath` is provided, returns a URL corresponding to `strPath`. If `strBase` is specified, the return value is a URL corresponding to `strPath`, relative to `strBase`. `strBase` can be a path, relative URL, or `file:///` URL to a base document.

Usage in VBScript

```
dim myString = ActiveDocument.Host.PathToURL("strPath", ["strBase"])
```

Example

```
// XMetaL Script Language JSCRIPT:
var u1, u2;
u1=ActiveDocument.Host.PathToURL("c:\\dir1\\dir2\\test.htm");
// Displays "file:///c|/dir1/dir2/test.htm"
ActiveDocument.Host.Alert(u1);
u2=ActiveDocument.Host.PathToURL("c:\\dir1\\images\\one.gif",
"c:\\dir1\\dir2\\test.htm");
// Displays "../images/one.gif"
ActiveDocument.Host.Alert(u2);
```

Prompt

Displays a dialog box containing a text box, above which the specified string is displayed. The dialog box also contains **OK** and **Cancel** buttons. The text in the entry box is returned if the user clicks **OK**; a null string is returned if the user clicks **Cancel.** Do not call `DocumentHost.Prompt` in the On_Document_Activate or On_Document_Deactivate event macros.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
var myString = ActiveDocument.Host.Prompt("strMessage", ["strText"], [intLength],
[intMaxLength], ["strTitle"]);
```

The text box contains the default text `strText`, if specified. `intLength` and `intMaxLength` specify the size and maximum length of the text box. `strTitle` specifies a title for the dialog box.

Usage in VBScript

```
dim myString = ActiveDocument.Host.Prompt("strMessage", ["strText"], [intLength],
[intMaxLength], ["strTitle"])
```

Example

```
// XMetaL Script Language JSCRIPT:
// Displays a text entry box with "apple" as the default entry
var myFruit = ActiveDocument.Host.Prompt("Type the name of a fruit",
"apple");
```

PushInMacro

'Pushes in' the macro button specified. This method is intended to be used only in the On_Update_UI event macro. The default is for the toolbar button to not be pushed in. In XMetaL Author, this method performs the same action as `Application.PushInMacro`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.Host.PushInMacro("strMacroName");
```

Usage in VBScript

```
ActiveDocument.Host.PushInMacro "strMacroName"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Push the toolbar button

if (!ActiveDocument.IsValid) {
ActiveDocument.Host.PushInMacro("SaveAsRTF");
}
```

QueryService

This method allows a document customization to tunnel up to the application layer in an XMetaL-agnostic manner (that is, independent of whether the customization is running inside XMetaL XMAX or XMetaL Author).

The return value is an object of type IDispatch or NULL. The IDispatch object is determined by implementors of the XMetaL Author event macro On_Application_Query_Service or the XMAX OnQueryService COM event. If neither of the events are implemented, the return value will be NULL. XMetaL Author has an exception to the rule whereby the serviceName parameter matches and return one of the three global script objects listed: Application, Documents, ResourceManager.

Applies to

XMetaL XMAX and XMetaL Author

Returns

IDispatch serviceImpl; // Corresponding service implementation for name/key, or NULL

Usage in JScript

```
ActiveDocument.Host.QueryService("strServiceName")
```

Usage in VBScript

```
ActiveDocument.Host.QueryService "strServiceName"
```

Example

```
// XMetaL Script Language JSCRIPT:
var xmsMgr = ActiveDocument.Host.QueryService("XMSERVICEMGR");
if (!xmsMgr)
{
return null;
}
return xmsMgr.FindByType("DITAAPP");}
```

Run

Runs the macro specified. In XMetaL Author, this method can be used to run both default XMetaL macros and DTD-specific macros. In XMetaL XMAX, this method can run DTD-specific macros only.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.Host.Run("strMyMacroName");
```

Usage in VBScript

```
ActiveDocument.Host.Run "strMyMacroName"
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.Host.Run("My macro");
```


RunKeyedMacro

Runs the macro that has the specified shortcut key. In XMetaL Author, this method can be used to run both default XMetaL macros and DTD-specific macros. In XMetaL XMAX, this method can run DTD-specific macros only. In XMetaL Author, this method performs the same action as `Application.RunKeyedMacro`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.Host.RunKeyedMacro("strShortcutKey");
```

Usage in VBScript

```
ActiveDocument.Host.RunKeyedMacro "strShortcutKey"
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.Host.RunKeyedMacro("Ctrl+Alt+G");
```

SetCursor

Sets the mouse cursor according to the value specified. To control the cursor shape within a document, use this method in the On_Mouse_Over and On_Mouse_Out event macros. To control the cursor shape when a document is activated and deactivated, use this method in the On_Document_Activate and On_Document_Deactivate event macros (typically you would set the cursor to 0 in these macros). XMetaL sometimes overrides this method, for example, when macro recording is taking place.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.Host.SetCursor(intType);
```

Allowed values are:

- sqDefaultCursor (0): pointer
- sqCursorArrow (1): arrow
- sqCursorBusy (2): “busy”
- sqCursorIBeam (3): I-beam
- sqCursorHand (4): open hand
- sqCursorGrabHand (5): closed hand

- sqCursorParaArrow (6): paragraph arrow
- sqCursorRightArrow (7): right arrow
- sqCursorCrossHair (8): cross hair
- sqCursorLeftRightDrag (9): drag right or left
- sqCursorUpDownDrag (10): drag up or down
- sqCursorResizeNESW (11): resize (northeast to southwest orientation)
- sqCursorResizeNWSE (12): resize (northwest to southeast orientation)
- sqCursorVDouble (13): vertical double arrow
- sqCursorHDouble (14): horizontal double arrow
- sqCursorVArrow (15): vertical single arrow
- sqCursorMacroRecording (16): “macro recording” (cassette)
- sqCursorNotAllowed (17): “not allowed”

In XMetaL Author, you can define up to 10 extra cursors by setting `cursor_file0`, ..., `cursor_file9` in the XMetaL configuration file to the names of `.cur` files. These variables default to `Cursors\cursor0.cur`, ..., `Cursors\cursor9.cur` in the XMetaL folder. User-defined cursors have `intType` values 18 through 27. These correspond to the constants `sqUserDefinedCursor0`, ..., `sqUserDefinedCursor9`.

Usage in VBScript

```
ActiveDocument.Host.SetCursor intType
```

Example

```
// XMetaL Script Language JSCRIPT:
// Set the cursor to "not allowed"
// if mouse is over a read-only container,
// and to the default cursor otherwise. // Get the node the mouse is over
var curNode=ActiveDocument.MouseOverNode;
var rng = ActiveDocument.Range;
// Select the contents of the node
rng.SelectNodeContents(curNode);
// Check the read-only status and
// set the cursor accordingly
if (rng.ReadOnly) {
ActiveDocument.Host.SetCursor(17);
} else {
ActiveDocument.Host.SetCursor(0);
}
```

SetStatusText

Sets the text at the left end of the status bar to the string specified. The OnStatus event fires in XMetaL XMAX after a call to this method. You can override the text specified in the OnStatus event if required. XMetaL Author sometimes overrides this method, for example, when the mouse is over a toolbar button.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.Host.SetStatusText("My status text");
```

Usage in VBScript

```
ActiveDocument.Host.SetStatusText "My status text"
```

Example

```
// XMetaL Script Language JSCRIPT:
// If the element that the mouse is
// over has a LINKEND attribute,
// display its value in the status bar. // Get the node the mouse is over
var curNode = ActiveDocument.MouseOverNode;
var Linkend = curNode.getAttribute("LINKEND");
// Check if the attribute value is non-null
// and set the status text accordingly

if (Linkend) {
ActiveDocument.Host.SetStatusText(Linkend);
} else {
ActiveDocument.Host.SetStatusText("");
}
```

URLToPath

Returns a URL. In XMetaL Author, this method performs the same action as `Application.URLToPath`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
var myString = ActiveDocument.Host.URLToPath("strURL", ["strBase"]);
```

If only `strURL` is specified, returns a path corresponding to the `strURL`. `strBase` can be a path pointing to a document. If `strURL` is a relative URL, and `strBase` is specified, returns a full path corresponding to `strURL`, using `strBase` as the base path.

Usage in VBScript

```
dim myString = ActiveDocument.Host.URLToPath("strURL", ["strBase"])
```

Example

```
// XMetaL Script Language JSCRIPT:
var p1, p2;
p1=ActiveDocument.Host.URLToPath("file:///c|/data/index.html");
// Displays "c:\data\index.html"
ActiveDocument.Host.Alert(p1);
p2=ActiveDocument.Host.URLToPath("../images/one.gif",
"d:\\dir1\\dir2\\test.htm");
// Displays "d:\dir1\images\one.gif"
ActiveDocument.Host.Alert(p2);
```

DocumentProperties

Enables a collection of user-defined properties to be associated with a document (that is, a Document object).

Each DocumentProperties object consists of one or more DocumentProperty objects, each of which represents a property. To obtain the DocumentProperties object for a particular document, use the `Document.CustomDocumentProperties` property.

Properties

count

Returns the number of properties (DocumentProperty objects) in the collection represented by the DocumentProperties object.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
DocumentProperties_object.count;
```

Usage in VBScript

```
DocumentProperties_object.count
```

Example

```
// XMetaL Script Language JSCRIPT:
var curDoc, curDocProps;
curDoc=ActiveDocument;
curDocProps=curDoc.CustomDocumentProperties;
if (curDocProps.count==0) {
ActiveDocument.Host.Alert("No custom properties.");
}
```

Methods

Add

Creates a new property with name and value specified and adds it to the collection represented by the DocumentProperties object.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DocumentProperties_object.Add("strName","strValue");
```

Usage in VBScript

```
DocumentProperties_object.Add "strName","strValue"
```

Example

```
// XMetaL Script Language JSCRIPT:
var curDoc, curDocProps;
curDoc=Application.ActiveDocument;
curDocProps=curDoc.CustomDocumentProperties;
curDocProps.Add("PreviewTempFile","foo.xml");
```

item

Returns an indexed DocumentProperty object.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DocumentProperty object

Usage in JScript

```
DocumentProperties_object.item(variantIndex);
```

`variantIndex` can be an integer starting from 1 to the value represented by `DocumentProperties.count`, or a string (the property name).

Usage in VBScript

```
DocumentProperties_object.item(variantIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Get the value of the "PreviewTempFile" property
var curDoc, curDocProps, ptf;
curDoc=Application.ActiveDocument;
curDocProps=curDoc.CustomDocumentProperties;
ptf = curDocProps.item("PreviewTempFile");
ActiveDocument.SaveAs(ptf.Value,false);
```

DocumentProperty

Enables a user-defined property to be created and added to a collection (a DocumentProperties object), which is associated with a document (that is, a Document object).

A DocumentProperty object is created with the `DocumentProperties.Add` method. Each property consists of a name and a value.

Properties

Name

Returns the property name.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DocumentProperty_object.Name;
```

Usage in VBScript

```
DocumentProperty_object.Name
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display names of all custom properties
// of the active document
var curDoc, curDocProps, i;
curDoc=ActiveDocument;
curDocProps=curDoc.CustomDocumentProperties;
i = 1;
while (i<=curDocProps.count) {
ActiveDocument.Host.Alert(curDocProps.item(i).Name);
i++;
}
```

value

Sets or returns the property value.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var strValue = DocumentProperty_object.value;
DocumentProperty_object.value = strValue;
```

Usage in VBScript

```
dim strValue = DocumentProperty_object.value
DocumentProperty_object.value = strValue
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display names and values of all custom properties
// of the active document
var curDoc, curDocProps, i;
curDoc=ActiveDocument;
curDocProps=curDoc.CustomDocumentProperties;
i = 1;
while (i<=curDocProps.count) {
ActiveDocument.Host.Alert(curDocProps.item(i).name + "=" +
curDocProps.item(i).value);
i++;
}
```

Methods

Delete

Deletes the property (the DocumentProperty object) from the collection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DocumentProperty_object.Delete();
```

Usage in VBScript

```
DocumentProperty_object.Delete
```

Example

```
// XMetaL Script Language JSCRIPT:
// Change the value of an existing property
var curDoc, curDocProps;
curDoc=Application.ActiveDocument;
curDocProps=curDoc.CustomDocumentProperties;
ptf = curDocProps.item("PreviewTempFile");
if (ptf.Value=="tmp.xml")
ptf.Delete();
```

Documents

This object is a collection of Document objects, representing all of the currently open XMetaL documents. There is only one Documents object, addressed as `Documents`. It is globally available only when scripting for XMetaL Author.

Properties

Count

The number of open documents.

Applies to

XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
Documents.Count;
```

Usage in VBScript

```
Documents.Count
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert(Documents.Count);
```

Methods

Add

Opens a new document and returns the corresponding Document object.

Applies to

XMetaL Author

Returns

Document object

Usage in JScript

```
var myDocument = Documents.Add([boolUseDefaultTemplate=true]);
```

If `boolUseDefaultTemplate` is True, the document is created from the default template. If `boolUseDefaultTemplate` is False then the **Choose DTD or Rules File** dialog box is displayed.

Usage in VBScript

```
dim myDocument = Documents.Add([boolUseDefaultTemplate=true])
```

Example

```
// XMetaL Script Language JSCRIPT:
var doc = Documents.Add();
```

ChooseTemplate

Shows the Choose Template dialog and returns the full-path filename to the template chosen or an empty string.

Applies to

XMetaL Author

Access

N/A

Returns

String

Usage in JScript

```
var templateChosen = Documents.ChooseTemplate();
```

Usage in VBScript

```
dim templateChosen = Documents.ChooseTemplate
```

Example

```
//XMetaL Script Language JSCRIPT:
var templateChosen = Documents.ChooseTemplate();
Application.Alert("The template chosen is " + templateChosen);
```

Close
Closes all open documents.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Documents.Close();
```

Usage in VBScript

```
Documents.Close
```

Example

```
// XMetaL Script Language JSCRIPT:
Documents.Close();
```

item
Returns the Document object representing the open document specified.

Applies to

XMetaL Author

Returns

Document object

Usage in JScript

```
Documents.item(variantIndex);
```

Valid numerical indexes range from 1 to the value represented by `Documents.Count`; string indexes should consist of the FullName property of the desired Document object.

Usage in VBScript

```
Documents.item(variantIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:
// return the first document
var doc = Documents.item(1);
```

Open
Opens the file specified and returns the corresponding Document object.

Applies to

XMetaL Author

Returns

Document object

Usage in JScript

```
Documents.Open("strFileName");
```

`strFileName` can be an absolute or relative path. If relative, it is interpreted as relative to the last folder from which a file was opened from the XMetaL user interface (that is, not from a script). If `intViewType` is specified, the document is opened in the corresponding view. The list of allowed values is the same as for `Document.ViewType`. When specifying the optional integer `intViewType`, use the following values:

- -1 (Unspecified)
- 0 (Normal view)
- 1 (Tags On view)
- 2 (Plain Text view)

You cannot open a document into the Page Preview (HTML view) view type.

Usage in VBScript

```
Documents.Open("strFileName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// substitute a path and file name for "yourFile"
var doc = Documents.Open("yourFile");
```

OpenEx

This method is reserved for internal use only.

OpenString

Creates a new file containing the contents specified. This method returns the corresponding Document object if the file was opened successfully.

Applies to

XMetaL Author

Returns

Document object

Usage in JScript

```
Documents.OpenString(strString, [intViewType], [strFileName], [boolTemplate], [boolIsXML],
 [boolWFEdit]);
```

If the file requires a document type declaration and/or XML declaration, these should be part of the string. If `intViewType` is specified, the document is opened in the corresponding view. The list of allowed values is the same as for `Document.ViewType`.

If `boolTemplate` is True, the file is opened as if it were a template: that is, the filename is `UntitledN` (N is a number), and choosing **Save** displays the **Save As** dialog box. In this case `strFileName` is used a pseudo-filename for the new document; the path is used to resolve relative paths in the file. If `boolTemplate` is False, the file is created with the file name `strFileName`. `boolIsXML` is True when the document being opened is XML. If `boolWFEdit` is True, then the document is edited as a well-formed-only document.

Usage in VBScript

```
Documents.OpenString(strString, [intViewType], [strFileName], [boolTemplate], [boolIsXML],
 [boolWFEdit])
```

Example

```
// XMetaL Script Language JSCRIPT:
var xmlDec = '<?xml version="1.0"?>';
var docType = '<!DOCTYPE Article SYSTEM "journalist.dtd">';
var contents1 = "<Article><Title>New Document</Title>";
var contents2 = "<Para>Contents</Para></Article>";
var doc = xmlDec + docType + contents1 + contents2;
// Open the document in Tags On view
Documents.OpenString(doc,1,"test.xml",false);
```

OpenTemplate

Creates a new document by opening the file specified as a template, and returns the corresponding Document object. When a file is opened as a template, the document title is “UntitledN”, and choosing **Save** opens the **Save As** dialog box rather than saving the original template file. If a template filename (which must be a full path) is not supplied, the tabbed **New** dialog box is opened.

Applies to

XMetaL Author

Returns

Document object

Usage in JScript

```
Documents.OpenTemplate(["strFileName"]);
```

Usage in VBScript

```
Documents.OpenTemplate(["strFileName"])
```

Example

```
// XMetaL Script Language JSCRIPT:
var doc=Documents.OpenTemplate("C:\\Data\\sample1.xml");
```

Save

Saves all open documents.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Documents.Save();
```

Usage in VBScript

```
Documents.Save
```

Example

```
// XMetaL Script Language JSCRIPT:
Documents.Save();
```

ElementList

Provides access to XMetaL Author's element list.

Using this interface you can show or hide the element list, create a new element list tab, and read and write the contents of the element list. If some elements are not displayed in the element list, but you want to expose them to the user, you can add the elements to the list. You can also specify which elements are populated in the element list by removing items.

If an element has been programmatically set to be read-only, and the user clicks in that element and can see all of the possible children in the element list, and then tries to insert one of these elements, the user does not get expected behavior from XMetaL Author because the parent is read-only. You can use this interface to override this behavior. You can also customize XMetaL Author so that certain users can only insert certain elements.

To get the ElementList interface, use `Application.ElementList`:

```
elmList = Application.ElementList
```

`elmList` is now the `ElementList` interface.

Some ElementList properties and methods can be used only in certain event macros. The count, item, and NameToSelect properties, as well as the InsertItem, InsertSortedItem, RemoveAllItems, and RemoveItem methods, can be used only in the On_Update_ElementList event macro.

The SelectedName property can be used only in the following event macros:

- On_ElementList_Insert
- On_ElementList_Change
- On_ElementList_Surround
- On_ElementList_Insert_NoRequired

The SelectTab method must not be used in the On_Update_ElementList event macro.

The remainder of the ElementList properties and methods can be used at any time from any macro in XMetaL Author.

If you are scripting for the XMetaL XMAX environment, use the CanElementList interface. Use the ElementList interface only when you are scripting for the XMetaL Author environment.

Properties

Count

This property is valid only during the On_Update_ElementList event macro. It returns the number of ElementListItem objects.

Applies to

XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
var intCount = Application.ElementList.Count;
```

Usage in VBScript

```
dim intCount =Application.ElementList.Count
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Put this code into On_Update_ElementList
set elmList = Application.ElementList
c = elmList.Count
Application.Alert "There are " & c & " items in the element list."
```

IsInsert

This property returns True if the element list has the ‘Insert’ radio button selected, and False if the ‘Change’ radio button is selected.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
Application.ElementList.IsInsert;
```

Usage in VBScript

```
Application.ElementList.IsInsert
```

Example

```
' XMetaL Script Language VBSCRIPT:
If Application.ElementList.IsInsert = True Then
MsgBox "The 'Insert' radio button is selected."
Else
MsgBox "The 'Change' radio button is selected."
End If
```

Item
Returns the ElementListItem object. This property is valid only during the On_Update_ElementList event macro.

Applies to

XMetaL Author

Access

Read only

Returns

ElementListItem object

Description

The ElementListItem object supports these properties:

- description - A read-write string that contains the description of the list item.
- name - A read-only string that contains the name of the element list item. This is the actual name of the element, and it can be different than `strDisplayName`. This is the qualified name of the element, in the form namespace prefix:local name. If the element has no namespace, the namespace prefix and colon are omitted.
- Required - A read-write boolean value. If this is true, the list item is displayed in bold font in the item list.
- UsageCount - A read-only long value indicating the number of times the element is used in all currently-open documents.

Usage in JScript

```
var it = Application.ElementList.Item(intIndex);
```

The range for the parameter `intIndex` is 0 to the value represented by `ElementList.Count-1`, and indicates which object to return. If you use `strDisplayName` as the parameter instead, this property returns the object with the name contained in the `strDisplayName` string.

Usage in VBScript

```
set it = Application.ElementList.Item(intIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put this code into On_Update_ElementList
var index = 1;
var elmList = Application.ElementList;

for (var i=0; i<=elmList.Count; i++) {
var elementListItem = eList.Item(i);
var d = elementListItem.description;
var n = elementListItem.name;
var r = elementListItem.Required;
var u = elementListItem.UsageCount;
Application.Alert ("Name: " + n + ". Description: " + d +
". Required: " + r + ", and used " + u + " times.");
}
```

NameToSelect

Returns the name to select in the element list (if it is in the list) and is set to the previously selected name when On_Update_ElementList is called. This property is valid only during the On_Update_ElementList event macro. It returns the actual name of the element to be selected, and not the display name of the element. The actual name of the element is a qualified name, in the form namespace prefix:local name. If the element has no namespace, the namespace prefix and colon are omitted.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var n = Application.ElementList.NameToSelect;
```

Usage in VBScript

```
set n = Application.ElementList.NameToSelect
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Put this code into On_Update_ElementList
set elmList = Application.ElementList
MsgBox elmList.NameToSelect & " was the last selected name."
```

SelectedName

Returns the name of the element selected in the element list. This property is used only in the following event macros: On_ElementList_Insert, On_ElementList_Change, On_ElementList_Surround,

On_ElementList_Insert_NoRequired. It returns the actual name of the element to be selected, and not the display name of the element. The actual name of the element is a qualified name, in the form namespace prefix:local name. If the element has no namespace, the namespace prefix and colon are omitted

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var sn = Application.ElementList.SelectedName;
```

Usage in VBScript

```
set sn = Application.ElementList.SelectedName
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put this code into one of the On_ElementList_ macros
var elmList = Application.ElementList;
var curelem = elmList.SelectedName;
Selection.InsertElement(curelem);
Application.Alert(curelem+"\n"+"On ElementList Insert radio button
selected");
```

TabName

Returns the name of the currently selected tab. If setting this property to a string value, the name of the currently selected tab is changed to the string.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var elmList = Application.ElementList.tabName;
```

Usage in VBScript

```
set elmList = Application.ElementList.tabName
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Change the tab name under certain circumstances
set elmList = Application.ElementList
If elmList.tabName = "MyTab" Then
elmList.tabName = "MyNewlyNamedTab"
End If
```

Visible

This property returns True if the element list is visible and False if the element list is hidden. You can also set the element list to be visible by making this property True, and hidden by making this property False.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var v = Application.ElementList.Visible;
```

Usage in VBScript

```
v = Application.ElementList.Visible
```

Example

```
' XMetaL Script Language VBSCRIPT:
If Application.ElementList.Visible = False Then
MsgBox "You cannot see the Element List."
Application.ElementList.Visible = True
MsgBox "Now you can."
Else
MsgBox "You can see the Element List."
Application.ElementList.Visible = False
MsgBox "Now you can't."
End If
```

Methods

AddTab

Adds a new custom tab with the name specified. Tabs created from scripting do not persist between XMetaL sessions. This means that you must put the code for creating your custom tabs in an event macro that is run each time XMetaL starts, or at an appropriate time. The most likely place is in the On_Application_Open

macro, but it could be in the On_Document_Activate with a call to RemoveTab in the On_Document_Deactivate. Using the Document-level macros ensures that your custom tab is seen only for a particular DTD.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ElementList.AddTab ("MyTab");
```

Usage in VBScript

```
Application.ElementList.AddTab "MyTab"
```

Example

```
' XMetaL Script Language VBSCRIPT:
set elmList = Application.ElementList
elmList.AddTab "MyTab"
```

InsertItem

It inserts a new item with the display name and description indicated at the position specified. This method is valid only during the On_Update_ElementList event macro.

Applies to

XMetaL Author

Returns

ElementListItem object

Usage in JScript

```
Application.ElementList.InsertItem(DisplayName, ItemDes, index);
```

`strDisplayName` is usually more descriptive than the actual name of the element being inserted. If the index is set to less than zero, or greater than `ElementList.Count`, this method inserts the new element at the end of the list.

Usage in VBScript

```
Application.ElementList.InsertItem(DisplayName, ItemDes, index)
```

Example

```
// XMetaL Script Language JSCRIPT:
var elmList = Application.ElementList;
var myDisplayName = "My Item";
var myItemDes = "My very own item";
```

```
var index = 1;
elmList.InsertItem(myDisplayName, myItemDes, index);
```

InsertSortedItem
Inserts a new item with the name and description specified into the element list using a case-insensitive alphabetic sort. This method is valid only during the On_Update_ElementList event macro. `strDisplayName` is usually more descriptive than the actual name of the element being inserted.

Applies to

XMetaL Author

Returns

ElementListItem object

Usage in JScript

```
Application.ElementList.InsertSortedItem(DisplayName, ItemDes);
```

Usage in VBScript

```
Application.ElementList.InsertSortedItem(DisplayName, ItemDes)
```

Example

```
// XMetaL Script Language JSCRIPT:
var elmList = Application.ElementList;
var myDisplayName = "My Item";
var myItemDes = "My very own item";
elmList.InsertSortedItem(myDisplayName,myItemDes)
```

RemoveAllItems
Removes all items from the element list. This method is useful in the case when the user's selection is inside of read-only content. It is valid only during the On_Update_ElementList event macro.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
 ElementList.RemoveAllItems;
```

Usage in VBScript

```
ElementList.RemoveAllItems
```


Example

```
' XMetaL Script Language VBSCRIPT:
' Remove all items from the list
If Selection.ReadOnly then
Application.ElementList.RemoveAllItems
End If
```

RemoveItem
Removes an item from the element list. This method is valid only during the On_Update_ElementList event macro. You can specify which item to remove by passing it the index number or display name of the item.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ElementList.RemoveItem("element display name")
Application.ElementList.RemoveItem(num)
```

Usage in VBScript

```
Application.ElementList.RemoveItem("element display name")
Application.ElementList.RemoveItem(num)
```

Example

```
// XMetaL Script Language JSCRIPT:
// If this is a journalist.dtd article, remove
// a couple of items from the element list
var a = ActiveDocument.doctype.name;
var elemList = Application.ElementList;
if(a=="Article")
{
elemList.RemoveItem("Insertion");
elemList.RemoveItem("Deletion");
}
```

RemoveTab
Removes the custom tab with the name specified. You cannot remove tabs that are part of the XMetaL system; you can remove only tabs that you have added yourself.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ElementList.RemoveTab ("MyTab");
```

Usage in VBScript

```
Application.ElementList.RemoveTab "MyTab"
```

Example

```
' XMetaL Script Language VBSCRIPT:
set elmList = Application.ElementList
elmList.RemoveTab "MyTab"
```

SelectTab
This method selects the tab with the name specified. Do not call this method from the On_Update_ElementList event macro.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ElementList.SelectTab("TabName");
```

Usage in VBScript

```
Application.ElementList.SelectTab "TabName"
```

Example

```
// XMetaL Script Language JSCRIPT:
var elemList = Application.ElementList;
elemList.SelectTab("Used");
```

Find

Provides access to XMetaL's find and replace functionality. There is a single Find object, addressed as `Find`. This interface can be accessed through the Find property of the Selection or Range objects.

Methods

Execute
Indicates whether the search was successful, and moves the selection to the found text. Execute cannot find invisible elements, such as table elements that are not displayed by the table editor, and elements that have been marked as 'Hide in Normal View'. Use `Selection.MoveToElement` to find such elements. Alerts

issued by Find can be suppressed with `Application.DisplayAlerts` or `DocumentHost.DisplayAlerts`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Selection_object.Find.Execute(["strFind"], ["strReplace"], ["strConstraintString"],
[boolMatchCase=false], [boolMatchWholeWords=false], [boolMatchPatterns=false],
[boolForward=true], [boolWrap=true], [intFindReplaceAction=1],
[boolSetFindDialogOptions=false]);
```

The parameters are:

- `strFind` = The string or pattern to locate
- `strReplace` = The string to replace `strFind` with
- `strFindIn` = Restrict the operation to `strFindIn` elements; can be an element name with or without tag delimiters
- `boolMatchCase` = match the case when searching
- `boolMatchWholeWords` = match whole words
- `boolMatchPatterns` = interpret pattern matching characters
- `boolForward` = search forward
- `boolWrap` = search wraps past the end of the document
- `intFindReplaceAction` = Type of find/replace action to perform.The possible values are: sqFind (0): Find (default), sqReplace (1): Replace, sqReplaceFind (2): Replace, then Find, sqFindReplaceAll (3): Replace All.
- `boolSetFindDialogOptions` = in addition to performing the find/replace, also initialize the XMetaL Find dialog box fields with the values specified with the other arguments

Usage in VBScript

```
Selection_object.Find.Execute(["strFind"], ["strReplace"], ["strConstraintString"],
[boolMatchCase=false], [boolMatchWholeWords=false], [boolMatchPatterns=false],
[boolForward=true], [boolWrap=true], [intFindReplaceAction=1],
[boolSetFindDialogOptions=false])
```

Example

```
// XMetaL Script Language JSCRIPT:
// find the next occurrence of "hello" in a
// "Para" element
Selection.Find.Execute("hello","","Para");
```

Formatting Object

The Formatting Object interface can be used to convert XML documents into HTML or PDF format for previewing and printing.

This interface is implemented with a number of macros, located in `..\XMetaL 5.0\Author\Startup\multipleOutput.mcr`.

You must install the following applications before using this interface:

- *Adobe Acrobat(TM) Reader*, with **Web Browser Integration** enabled
- *Sun(TM) Java Runtime Engine 2.0*
- *Formatting Object Processor*

To enable these capabilities, you may encounter problems with the versions of the JRE and FOP. Please contact XMetaL Support for more information on the currently supported versions of this software. Make a note of the installation location of FOP: you will need it when using the XMLToPDFSetup method.

Consult the XMetaL Customization Guide for more information about enabling PDF and HTML previewing.

Methods

previewHTML
Converts the current XML document to HTML, and displays the HTML in Page Preview view. You must call `XMLToHTMLSetup` before using this method.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
previewHTML();
```

Usage in VBScript

```
previewHTML
```

Example

```
// XMetaL Script Language JSCRIPT:
previewHTML();
```

previewPDF
Converts the current XML document to PDF format, and displays the PDF in Page Preview view. You must call `XMLToPDFSetup` before using this method.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
previewPDF();
```

Usage in VBScript

```
previewPDF
```

Example

```
// XMetaL Script Language JSCRIPT:
previewPDF();
```

saveAsHTML
Saves the current XML document as HTML. Displays the **Save As** dialog box before saving. You must call `XMLToHTMLSetup` before using this method.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
saveAsHTML();
```

Usage in VBScript

```
saveAsHTML
```

Example

```
// XMetaL Script Language JSCRIPT:
saveAsHTML();
```

saveAsPDF
Saves the current XML document in PDF format. Displays the Save As dialog box before saving. You must call `XMLToPDFSetup` at least once before using this method.

Applies to

XMetaL Author

Returns

No return value.

Usage in JScript

```
saveAsPDF();
```

Usage in VBScript

```
saveAsPDF
```

Example

```
// XMetaL Script Language JSCRIPT:
saveAsPDF();
```

XMLToHTMLSetup
Creates an XSLT file used to generate HTML output from an XML document. This method must be run before executing `previewHTML` or `saveAsHTML`. This method needs to be called only once per installation of XMetaL Author, unless you need to make changes to the output configuration.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
XMLToHTMLSetup();
```

Usage in VBScript

```
XMLToHTMLSetup
```

Example

```
// XMetaL Script Language JSCRIPT:
XMLToHTMLSetup();
```

XMLToPDFSetup
Displays the XML To PDF Setup dialog box. This method must be run before executing `previewPDF` or `saveAsPDF`. This method needs to be called only once per installation of XMetaL Author, unless you need to make changes to the output configuration.

Applies to

XMetaL Author

Returns

No return value

Description

The user must enter the following information into the XML To PDF Setup dialog box:

- Check the Create XSL Style Sheet check box
- Check the Create Batch File check box
- Type the Folder path of FOP (an absolute path to the foldercontaining the Formatting Object Processor)

After the user clicks **OK**, previewPDF or saveAsPDF. To modify the page setup and margins of the PDF output, check the Page Setup check box in the XML To PDF Setup dialog box. Run `XMLToPDFSetup` whenever the page setup should be changed.

Usage in JScript

```
XMLToPDFSetup();
```

Usage in VBScript

```
XMLToPDFSetup
```

Example

```
// XMetaL Script Language JSCRIPT:
XMLToPDFSetup();
```

Mark

Reserved for internal use only.

Methods

Hide
This method is reserved for internal use only.

HideFromRevisionTracking
This method is reserved for internal use only.

Redraw
This method is reserved for internal use only.

Remove
This method is reserved for internal use only.

Show
This method is reserved for internal use only.

ShowToRevisionTracking
This method is reserved for internal use only.

Properties

Id
This property is reserved for internal use only.

Flags
This property is reserved for internal use only.

IsEqual
This property is reserved for internal use only.

IsGreaterThan
This property is reserved for internal use only.

IsLessThan
This property is reserved for internal use only.

Range
This property is reserved for internal use only.

MarkWalker

Reserved for internal use only.

Methods

Properties

First
This property is reserved for internal use only.

Last
This property is reserved for internal use only.

Next
This property is reserved for internal use only.

Previous
This property is reserved for internal use only.

NameVariantProperties

This interface enables a collection of user-defined properties to be associated with the application (that is, the Application object). This interface is available only when you are scripting for the XMetaL Author environment.

These properties can be used to store global information, so that it is accessible to any scripting language (variables set in a specific scripting language are available only to scripts written in the same language). This

mechanism can be used to store strings, numbers, arrays, and any other object that can be represented by a variable.

Each NameVariantProperties object consists of one or more `NameVariantProperty` objects (each of which represents a property).

To obtain the NameVariantProperties object, use the `Application.CustomProperties` property.

Properties

count

Returns the number of properties in the collection represented by the NameVariantProperties object.

Applies to

XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
NameVariantProperties_object.count;
```

Usage in VBScript

```
NameVariantProperties_object.count
```

Example

```
// XMetaL Script Language JSCRIPT:
curAppProps=Application.CustomProperties;
if (curAppProps.count==0) {
Application.Alert("No custom properties.");
}
```

Methods

Add

Creates a new property with name and value specified and adds it to the collection represented by the NameVariantProperties object.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
NameVariantProperties_object.Add("strName",variantValue);
```

`variantValue` can be a number, string, object, or anything else that can be represented by a variable in the scripting language

Usage in VBScript

```
NameVariantProperties_object.Add "strName", variantValue
```

Example

```
// XMetaL Script Language JSCRIPT:
var curAppProps;
curAppProps=Application.CustomProperties;
curAppProps.Add("PreviewTempFile","foo.xml");
```

item
Returns an indexed NameVariantProperty object.

Applies to

XMetaL Author

Returns

NameVariantProperty

Usage in JScript

```
NameVariantProperties_object.item(variantIndex);
```

`variantIndex` can be an integer starting at 1, or a string (the property name).

Usage in VBScript

```
NameVariantProperties_object.item(variantIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Get the value of the "PreviewTempFile" property
var curAppProps, ptf;
curAppProps=Application.CustomProperties;
ptf = curAppProps.item("PreviewTempFile");
ActiveDocument.SaveAs(ptf.Value,false);
```

NameVariantProperty

This interface enables a user-defined property to be created and added to a collection (a NameVariantProperties object), which is associated with an application (an Application object). This interface is only available when you are scripting for the XMetaL Author environment.

These properties can be used to store global information, so that it is accessible to any scripting language (variables set in a specific scripting language are available only to scripts written in the same language). This mechanism can be used to store strings, numbers, array, and any other object that can be represented by a variable.

A NameVariantProperty object is created with the `NameVariantProperties.Add` method. Each property consists of a name and a value.

Properties

Name

Returns the property name.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
NameVariantProperty_object.Name;
```

Usage in VBScript

```
NameVariantProperty_object.Name
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display names of all custom properties
// of the application

var curAppProps, i;
curAppProps=Application.CustomProperties;
i = 1;
while (i<=curAppProps.count) {
Application.Alert(curAppProps.item(i).Name);
i++;
}
```

value

Returns the property value. The value can be a number, string, object, or anything else that can be represented by a variable in the scripting language. To change the value of a property, delete it and then add a new property with the same name and a new value.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
vbl = NameVariantProperty_object.value;
```

Usage in VBScript

```
vbl = NameVariantProperty_object.value
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display names and values of all custom properties
// of the application

var curAppProps, i;
curAppProps=Application.CustomProperties;
i = 1;
while (i<=curAppProps.count) {
Application.Alert(curAppProps.item(i).name+" has a value of
"+curAppProps.item(i).value);
i++;
}
```

Methods

Delete

Deletes the property (the NameVariantProperty object) from the collection.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
NameVariantProperty_object.Delete();
```

Usage in VBScript

```
NameVariantProperty_object.Delete
```

Example

```
// XMetaL Script Language JSCRIPT:
// Change the value of an existing property
var curAppProps;
```



```
curAppProps=Application.CustomProperties;
ptf = curAppProps.item("PreviewTempFile");
if (ptf.Value=="tmp.xml")
ptf.Delete();
```

Range

A Range object represents a range of selected content in any open document.

By contrast, the Selection object always represents the user's current selection. This object inherits all of its properties and methods from the Selection interface. Range objects can be located anywhere in a document, whereas Selection objects can be located only where the insertion point can be located in Normal view.

Note: Unless explicitly stated otherwise, do not use Selection or Range objects in Plain Text view.

It is often more useful to use a Range object when moving around the document for the following reasons:

- There are no restrictions on where range objects can be placed
- The movement of range objects is invisible to the user
- You can have as many range objects as you need, but only one selection
- If the user moves the insertion point/selection, range objects are not affected
- Range objects stay the same when you switch documents.

The `Document.Range` property returns a Range object corresponding to the document's selection. You can have multiple Range objects at one time, but only one Selection object. Range objects are 'invisible' until selected. If you move a Range object to a location and want to see it, you have to select it using the `Selection.Select` method.

ResolveEntityInfo

This interface enables events and event macros to resolve entity references. The `Application.ResolveEntityInfo` property, available only inXMetaL Author, returns a ResolveEntityInfo object. Its properties are intended for use in the On_Application_Resolve_Entity event macro. In XMetaL XMAX, the ResolveEntityInfo object is provided as a parameter of the OnResolveEntity event.

Properties

BasePath

Contains the file system path and filename of the XML document that contains the entity reference.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var myBasePath = ResolveEntityInfo_object.BasePath;
ResolveEntityInfo_object.BasePath = myBasePath;
```

Usage in VBScript

```
dim myBasePath = ResolveEntityInfo_object.BasePath
ResolveEntityInfo_object.BasePath = myBasePath
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put this code into On_Application_Resolve_Entity
// This code is applicable to XMetaL Author only

function ResolveEntityURL(){
var url = Application.ResolveEntityInfo.systemID;
var re = /^iwts:.*/i;// test for interwoven URL prefix

if( !re.test(url) ){
return 0;
}
url=url.substr(5);// remove prefix
try {
var docPath=Application.ResolveEntityInfo.BasePath;
docPath=Application.URLToPath(docPath);
var newURL="c:\\entfiles" + url;
Application.ResolveEntityInfo.Filename=newURL.replace(/\//g,"\\");
} catch(e) {
Application.Alert("Unexpected exception=" + e.description);
return -1;
}
}
ResolveEntityURL();
```

FileName

Contains the file system path and filename of the external DTD or entity reference.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var myFileName = ResolveEntityInfo_object.FileName;
ResolveEntityInfo_object.FileName = myFileName;
```


`FileName` is used by XMetaL Author and XMetaL XMAX to look for external references. If you set `FileName` to a TeamXML path name (that is, "iwts:/..."), neither XMetaL Author nor XMetaL XMAX will find the external reference. You must set this parameter to something like the following:

- `"C:\Program Files\XMetaL 5.0\Rules\Journalist.dtd"`
- `"http://www.someurl.com/somedtd.dtd"` (URL)
- `"../../Rules/journalist.dtd"` (Relative local path)

Usage in VBScript

```
dim myFileName = ResolveEntityInfo_object.FileName
ResolveEntityInfo_object.FileName = myFileName
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put this code into On_Application_Resolve_Entity
// This code is applicable to XMetaL Author only

function ResolveEntityURL(){
var url = Application.ResolveEntityInfo.systemID;
var re = /^iwts:.*/i;// test for interwoven URL prefix

if( !re.test(url) ){
return 0;
}
url=url.substr(5);// remove prefix
try {
var docPath=Application.ResolveEntityInfo.BasePath;
docPath=Application.URLToPath(docPath);
var newURL="c:\\entfiles" + url;
Application.ResolveEntityInfo.Filename=newURL.replace(/\//g,"\\");
} catch(e) {
Application.Alert("Unexpected exception=" + e.description);
return -1;
}
}
ResolveEntityURL();
```

name
The name of the entity or document type name.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var myName = ResolveEntityInfo_object.name;
ResolveEntityInfo_object.name = myName;
```

Usage in VBScript

```
dim myName = ResolveEntityInfo_object.name
ResolveEntityInfo_object.name = myName
```

Example

```
//XMetaL Script Language JSCRIPT:
// This code is applicable to XMetaL Author only

Application.Alert("Name of entity or document type name: " +
Application.ResolveEntityInfo.name);
```

publicID

The public identifier component of the external identifier.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var myPublicID = ResolveEntityInfo_object.publicID;
ResolveEntityInfo_object.publicID = myPublicID;
```

Usage in VBScript

```
dim myPublicID = ResolveEntityInfo_object.publicID
ResolveEntityInfo_object.publicID = myPublicID
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is applicable to XMetaL Author only

Application.Alert("Public ID: " + Application.ResolveEntityInfo.publicID);
```

systemID

The system identifier component of the external identifier. Examples of `systemID`: `"http://www.someurl.com/somedtd.dtd"`; `"../../Rules/journalist.dtd"`; `"iwts:/rules/publication.dtd"`.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var mySystemID = ResolveEntityInfo_object.systemID;
ResolveEntityInfo_object.systemID = mySystemID;
```

Usage in VBScript

```
dim mySystemID = ResolveEntityInfo_object.systemID
ResolveEntityInfo_object.systemID = mySystemID
```

Example

```
// XMetaL Script Language JSCRIPT:
// Put this code into On_Application_Resolve_Entity
// This code is applicable to XMetaL Author only

function ResolveEntityURL(){
var url = Application.ResolveEntityInfo.systemID;
var re = /^iwts:.*/i;// test for interwoven URL prefix

if( !re.test(url) ){
return 0;
}
url=url.substr(5);// remove prefix
try {
var docPath=Application.ResolveEntityInfo.BasePath;
docPath=Application.URLToPath(docPath);
var newURL="c:\\entfiles" + url;
Application.ResolveEntityInfo.Filename=newURL.replace(/\//g,"\\");
} catch(e) {
Application.Alert("Unexpected exception=" + e.description);
return -1;
}
}
ResolveEntityURL();
```

Methods

IsDocType

Returns True if the reference has come from the document type declaration, and False if it does not.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
var myBoolean = ResolveEntityInfo_object.IsDocType();
```

Usage in VBScript

```
dim myBoolean = ResolveEntityInfo_object.IsDocType
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use in XMetaL Author

if (Application.ResolveEntityInfo.IsDocType()) {
Application.Alert("Reference came from DOCTYPE declaration.")
} else {
Application.Alert("Reference did not come from DOCTYPE declaration."
}
```

ResourceManager

The ResourceManager interface enables you customize the XMetaL Resource Manager by adding new tabs, which can contain ActiveX controls of your choice. This interface is available only when scripting for XMetaL Author.

ResourceManager also gives you access to the Assets interface, which enables some customizations of the Resource Manager Assets tab. ResourceManager is a global interface that is available from any script.

Properties

Assets (unsupported)

Returns a COM object that provides access to the Resource Manager Assets tab.

Applies to

XMetaL Author

Access

Read only

Returns

Assets

Usage in JScript

```
ResourceManager.Assets;
```

Usage in VBScript

```
ResourceManager.Assets
```

Example

```
// XMetaL Script Language JSCRIPT:
var rmAssets = ResourceManager.Assets;
var browser = rmAssets.WebBrowser;
```



```
// Use a browser-specific method
browser.Navigate("www.xmetal.com");
```

ControlInTab
Returns the IDispatch object corresponding to the control that is displayed in the specified Resource Manager tab. This object gives the script access to the control's properties and methods.

Applies to

XMetaL Author

Access

Read only

Returns

IDispatch object

Usage in JScript

```
ResourceManager.ControlInTab("strTabname");
```

Usage in VBScript

```
ResourceManager.ControlInTab("strTabname")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Uses Microsoft Calendar control
ResourceManager.AddTab("Date","MSCAL.Calendar");
var calCtrl = ResourceManager.ControlInTab("Date");
// Set control-specific properties
calCtrl.Day = 16;
calCtrl.Month = 4;
calCtrl.Year = 2002;
```

FrontmostTabName
Returns the name of the tab that is in the frontmost position in the Resource Manager.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
ResourceManager.FrontmostTabName;
```

Usage in VBScript

```
ResourceManager.FrontmostTabName
```

Example

```
Application.Alert("The frontmost tab in the Resource Manager is " +
ResourceManager.FrontmostTabName);
```

TabNames
Returns the VB safearray containing the custom tab names currently inside the Resource Manager panel.

Applies to

XMetaL Author

Access

Read only

Returns

VB safearray

Usage in JScript

```
var jscriptTabNames = ResourceManager.TabNames.toArray();
```

Usage in VBScript

```
ResourceManager.TabNames
```

Example

```
// XMetaL Script Language JSCRIPT:
for (var i = ResourceManager.TabNames.lbound(); i <=
ResourceManager.TabNames.ubound(); i++) {
  Application.Alert("TabNames(" + i + ") is named " +
ResourceManager.TabNames.getItem(i));
}
```

Visible
Opens or closes the Resource Manager.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
vbl = ResourceManager.Visible;
ResourceManager.Visible = boolState;
```

Usage in VBScript

```
vbl = ResourceManager.Visible
ResourceManager.Visible = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
// If the Resource Manager is not open, open it

if (!ResourceManager.Visible) {
ResourceManager.Visible = true;
}
```

Methods

AddTab

Adds a tab with the name specified to the Resource Manager, and inserts the ActiveX control with the ProgId specified into the tab. This method can also add previously removed built-in tabs.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
ResourceManager.AddTab("strTabname","strProgId");
```

Usage in VBScript

```
ResourceManager.AddTab "strTabname", "strProgId"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a tab containing the Microsoft Calendar control, and
// re-create the built-in Assets tab which has been removed
ResourceManager.AddTab("Date","MSCAL.Calendar");
ResourceManager.AddTab("Assets","");
```

HideTab

Hides a tab with the name specified to the Resource Manager, and if last tab visible, hides the entire docking panel. This method can also hide built-in tabs.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
ResourceManager.HideTab("strTabname");
```

Usage in VBScript

```
ResourceManager.HideTab "strTabname"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a tab containing the Microsoft Calendar control, and
// hide tab just added
ResourceManager.AddTab("Date","MSCAL.Calendar");
ResourceManager.HideTab("Date");
```

RemoveAllTabs
Removes all custom tabs from the Resource Manager.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
ResourceManager.RemoveAllTabs();
```

Usage in VBScript

```
ResourceManager.RemoveAllTabs
```

Example

```
// XMetaL Script Language JSCRIPT:
ResourceManager.RemoveAllTabs();
```

RemoveTab
Removes the tab with the name specified from the Resource Manager. This method can also remove built-in tabs.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
ResourceManager.RemoveTab("strTabname");
```

Usage in VBScript

```
ResourceManager.RemoveTab "strTabname"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a tab
var visible = ResourceManager.Visible;
ResourceManager.Visible = true;
ResourceManager.AddTab("Date","MSCAL.Calendar");
ResourceManager.SelectTab("Date");
Application.Alert("Tab added; now will be removed.");
// Remove the tab
ResourceManager.RemoveTab("Date");
Application.Alert("Date tab has been removed.");
// Remove the built-in Assets tab
ResourceManager.RemoveTab("Assets");
Application.Alert("Assets tab has been removed.");
ResourceManager.Visible = visible;
```

RenameTab

Renames the specified tab in the Resource Manager to the specified new name. You cannot rename the built-in tabs (Assets, Desktop).

Applies to

XMetaL Author

Access

N/A

Returns

No return value

Usage in JScript

```
ResourceManager.RenameTab ("strOldTabName", "strNewTabName");
```

Usage in VBScript

```
ResourceManager.RenameTab ("strOldTabName", "strNewTabName")
```

Example

```
// XMetaL Script Language JSCRIPT:
ResourceManager.RenameTab ("My Assets", "New Assets");
```

SelectTab

Makes the tab with the name specified the currently selected tab.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
ResourceManager.SelectTab("strTabname");
```

Usage in VBScript

```
ResourceManager.SelectTab "strTabname"
```

Example

```
// XMetaL Script Language JSCRIPT:
ResourceManager.SelectTab("Assets");
```

ShowTab

Shows a tab with the specified name to the Resource Manager, and makes this tab the frontmost tab. This method can also show previously hidden built-in tabs.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
ResourceManager.ShowTab("strTabname");
```

Usage in VBScript

```
ResourceManager.ShowTab "strTabname"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a tab containing the Microsoft Calendar control, and
// re-create the built-in Assets tab which has been removed
ResourceManager.AddTab("Date","MSCAL.Calendar");
ResourceManager.ShowTab("Date");
```


ResultsManager

The ResultsManager interface enables you customize the XMetaL Results panel by adding new tabs, which can contain ActiveX controls of your choice. Typically, new tabs are also added to the View > Results submenu via the `CommandBars.AddToResults` method. This interface is available only when scripting for XMetaL Author.

ResultsManager also gives you access to the Validation and XInclude results. ResultsManager is a property on the global Application interface that is available from any script.

Properties

ControlInTab

Returns the IDispatch object corresponding to the control that is displayed in the specified Results Manager tab. This object gives the script access to the control's properties and methods.

Applies to

XMetaL Author

Access

Read only

Returns

IDispatch object

Usage in JScript

```
var axCtlDisp = Application.ResultsManager.ControlInTab("strTabname");
```

Usage in VBScript

```
Dim axCtlDisp = Application.ResultsManager.ControlInTab "strTabname"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Uses Microsoft Calendar control
Application.ResultsManager.AddTab("Date","MSCAL.Calendar");
var calCtrl = Application.ResultsManager.ControlInTab("Date");
// Set control-specific properties
calCtrl.Day = 16;
calCtrl.Month = 4;
calCtrl.Year = 2002;
```

FrontmostTabName

Returns the name of the tab that is in the frontmost position in the Results Manager.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Application.ResultsManager.FrontmostTabName;
```

Usage in VBScript

```
Application.ResultsManager.FrontmostTabName
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert("The frontmost tab in the Results Manager is " +
Application.ResultsManager.FrontmostTabName);
```

IsLinkLogTabVisible

Opens or closes the Results Manager.

Applies to

XMetaL Author

Access

Read-only

Returns

Boolean

Usage in JScript

```
var vis = Application.ResultsManager.IsLinkLogTabVisible;
```

Usage in VBScript

```
Dim vis = Application.ResultsManager.IsLinkLogTabVisible
```

Example

```
// XMetaL Script Language JSCRIPT:
// If the Link Log tab is not visible, show it
if (!Application.ResultsManager.IsLinkLogTabVisible) {
  Application.ResultsManager.ShowLinkLogTab();
}
```

TabNames

Returns the VB safearray containing the custom tab names currently inside the Results Manager panel.

Applies to

XMetaL Author

Access

Read only

Returns

VB safearray

Usage in JScript

```
var jscriptTabNames = Application.ResultsManager.TabNames.toArray();
```

Usage in VBScript

```
Dim vbArray = Application.ResultsManager.TabNames
```

Example

```
// XMetaL Script Language JSCRIPT:
for (var i = Application.ResultsManager.TabNames.lbound(); i <=
Application.ResultsManager.TabNames.ubound(); i++) {
  Application.Alert("TabNames(" + i + ") is named " +
Application.ResultsManager.TabNames.getItem(i));
}
```

Visible

Opens or closes the Results Manager.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var vis = Application.ResultsManager.Visible;
Application.ResultsManager.Visible = boolState;
```

Usage in VBScript

```
Dim vis = Application.ResultsManager.Visible
Application.ResultsManager.Visible = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
// If the Results Manager is not open, open it

if (!Application.ResultsManager.Visible) {
  Application.ResultsManager.Visible = true;
}
```

Methods

AddTab

Adds a tab with the specified name to the Results Manager, and inserts the ActiveX control, with the ProgId specified, into the tab. This method can also add previously removed built-in tabs.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ResultsManager.AddTab("strTabname","strProgId");
```

Usage in VBScript

```
Application.ResultsManager.AddTab "strTabname", "strProgId"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a tab containing the Microsoft Calendar control, and
// re-create the built-in Assets tab which has been removed
Application.ResultsManager.AddTab("Date","MSCAL.Calendar");
```

AddWebTab

Adds a tab with the specified name to the Results Manager, and inserts the Windows Internet Explorer Web Browser ActiveX control onto the tab, and additional object dispatch interface to return whenever the hosted web page calls `document.external` , and an optional registry subkey path value to override the default Windows Internet Explorer options used by the hosted web browser control.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ResultsManager.AddWebTab("strTabname",externalObjectDisp,
"strOptionalRegSubKeyPath");
```

Usage in VBScript

```
Application.ResultsManager.AddWebTab "strTabname", externalObjectDisp,
"strOptionalRegSubKeyPath"
```

Example

```
// XMetaL Script Language JSCRIPT:
var someResults = new Array(1,2,3);
Application.ResultsManager.AddWebTab("My Results", someResults);
```

Related Links

IDocHostUIHandler::GetOptionKeyPath method

HideLinkLogTab (ResultsManager)

This event closes the Link Log tab in the Results pane.

Applies to

XMetaL® Author

Returns

No return value

Usage in JScript

```
Application.ResultsManager.CloseLinkLogTab()
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.ResultsManager.CloseLinkLogTab();
```

HideTab

Hides a tab with the specified name to the Results Manager, and if the last tab is visible, hides the entire docking panel. This method can also hide built-in tabs.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
Application.ResultsManager.HideTab("strTabname");
```

Usage in VBScript

```
Application.ResultsManager.HideTab "strTabname"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a tab containing the Microsoft Calendar control, and
// hide tab just added
Application.ResultsManager.AddTab("Date","MSCAL.Calendar");
Application.ResultsManager.HideTab("Date");
```

NavigateInWebTab

Loads a URL into the web tab with the specified name within the Results Manager.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ResultsManager.NavigateInWebTab("strTabname","C:\\someWebPage.html");
```

Usage in VBScript

```
Application.ResultsManager.NavigateInWebTab "strTabname", "C:\someWebPage.html"
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.ResultsManager.AddWebTab("My Results");
Application.ResultsManager.NavigateInWebTab("My Results",
"http://www.google.com");
Application.ResultsManager.ShowTab("My Results");
```

RemoveTab

Removes the tab with the specified name from the Results Manager. This method can also remove built-in tabs.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ResultsManager.RemoveTab("strTabname");
```

Usage in VBScript

```
Application.ResultsManager.RemoveTab "strTabname"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a tab
var visible = Application.ResultsManager.Visible;
Application.ResultsManager.Visible = true;
Application.ResultsManager.AddTab("Date","MSCAL.Calendar");
Application.ResultsManager.SelectTab("Date");
Application.Alert("Tab added; now will be removed.");
// Remove the tab
Application.ResultsManager.RemoveTab("Date");
Application.Alert("Date tab has been removed.");
// Remove the built-in Assets tab
Application.ResultsManager.RemoveTab("Assets");
Application.Alert("Assets tab has been removed.");
Application.ResultsManager.Visible = visible;
```

RenameTab
Renames the specified tab in the Results Manager to use the specified new name. You cannot rename the built-in tabs (Assets, Desktop).

Applies to

XMetaL Author

Access

N/A

Returns

No return value

Usage in JScript

```
Application.ResultsManager.RenameTab ("strOldTabName", "strNewTabName");
```

Usage in VBScript

```
Application.ResultsManager.RenameTab ("strOldTabName", "strNewTabName")
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.ResultsManager.RenameTab ("My Results", "New Results");
```

SelectTab
Makes the tab with the specified name the currently selected tab.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Application.ResultsManager.SelectTab("strTabname");
```

Usage in VBScript

```
Application.ResultsManager.SelectTab "strTabname"
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.ResultsManager.SelectTab("Assets");
```

ShowLinkLogTab (ResultsManager)

This method opens the Link Log tab in the Results pane.

Applies to

XMetaL® Author

Returns

No return value

Usage in JScript

```
Application.ResultsManager.ShowLinkLogTab()
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.ResultsManager.ShowLinkLogTab()
```

ShowTab

Shows a tab with the specified name to the Results Manager, and makes this tab the frontmost tab. This method can also show previously hidden built-in tabs.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
Application.ResultsManager.ShowTab("strTabname");
```

Usage in VBScript

```
Application.ResultsManager.ShowTab "strTabname"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a tab containing the Microsoft Calendar control, and
// re-create the built-in Assets tab which has been removed
Application.ResultsManager.AddTab("Date","MSCAL.Calendar");
Application.ResultsManager.ShowTab("Date");
```

Selection

A Selection object represents a document's selection (which could be an insertion point).

The Selection object always represents the user's selection in the active document. It is evaluated dynamically and always takes its value from the current active document. This is true even when it is represented by a variable. For example, when scripting for XMetaL Author, if you define:

```
var sel=Application.Selection;
```

and a different document becomes active, a reference to `sel` returns the user's selection in the new active document.

By contrast, a Range object can represent a selection in any open document. (When scripting for XMetaL XMAX, note that only one document is open at a time.) The Range interface inherits the Selection interface, so we regard Range objects as Selection objects also. However, the special Selection object can be considered to be the only "pure" instance of the Selection interface: that is, it is the only Selection object that is not also a Range object.

In the Usage sections, the expression `Selection_object` refers equally to Selection and Range objects.

Some Selection properties and methods apply to the selection's container. This refers to the markup construct that surrounds the selection: usually this is an element, but can also be a CDATA section, processing instruction, or comment. Other properties and methods apply specifically to the current element.

The Selection interface is used for manipulating selections, but it also provides an interface to many other document-manipulation operations. You can also use DOM methods to modify the document.

It is generally better to apply Selection methods to Range objects when modifying the document's structure (for example, inserting elements) or moving around in the document. This is because Selection and Range work differently in the XMetaL Normal view (they work the same in Tags On view). In Normal view, a Selection object can be placed only where text is allowed. This means that it cannot be placed between tags unless text is also allowed there. By contrast, Range objects can be placed anywhere in Normal view.

Macros that are expected to work the same in the Normal and Tags On views should therefore use Range objects when moving around, and should place the user selection appropriately before the macro terminates. You may want to code all of your macros in this way, even if you intend them to work only in Tags On view, in case they are later re-purposed for Normal view operations.

Note: Unless explicitly stated otherwise, do not use Selection or Range objects in Plain Text view.

Some other reasons for using Range objects are:

- The movement of Range objects is invisible to the user
- You can have as many Range objects as you need, but only one selection
- If the user moves the insertion point/selection, Range objects are not affected
- Range objects stay the same when you switch documents

Properties

CanChange

Indicates whether the selection's container can be changed to the specified element. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanChange("strElementName");
```

Usage in VBScript

```
Selection_object.CanChange("strElementName")
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanChange("NOTE")) {
Selection.ContainerName="NOTE";
} else {
ActiveDocument.Host.Alert("Cannot change to NOTE.");
}
```

CanChangeList

Returns a CanElementList. You can change the current selection into any element listed.

Applies to

XMetaL XMAX

Access

Read only

Returns

CanElementList object

Usage in JScript

```
Selection.CanChangeList;
```

Usage in VBScript

```
Selection.CanChangeList
```

Example

```
// XMetaL Script Language JSCRIPT:
// display a list of all items that you can
// change the current selection into

// pre: myXMControl has previously been defined as an XMetaL XMAX object

function showCanChange() {
var sel = myXMControl.Selection;
var list = sel.CanChangeList;
var num = list.Count;
var str = "";

for (i=0; i<num; i++) {
str+= list.Item(i).Name;
str += "\n";
}
ActiveDocument.Host.Alert(str);
}
```

CanChangeNS

Indicates whether the selection's container can be changed to the specified element, which represents an element local to the namespace specified. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanChangeNS("strNamespacenameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.CanChangeNS("strNamespacenameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanChangeNS("http://www.mydomainname.com", "NOTE")) {
Selection.ContainerNameNS="{http://www.mydomainname.com}NOTE";
} else {
ActiveDocument.Host.Alert("Cannot change to NOTE.");
}
```

CanDelete

Indicates whether the selection can validly be deleted.

Applies To

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanDelete;
```

Usage in VBScript

```
Selection_object.CanDelete
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanDelete) {
Selection.Cut();
}
```

CanInsert

Indicates whether the element specified can replace the selection. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanInsert("strElementName");
```

Usage in VBScript

```
Selection_object.CanInsert("strElementName")
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanInsert("P")) {
Selection.InsertElement("P");
} else {
ActiveDocument.Host.Alert("Cannot insert P here.");
}
```

CanInsertList
Returns a CanElementList object. You can insert any element listed into the current selection.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

CanElementList object

Usage in JScript

```
Selection.CanInsertList;
```

Usage in VBScript

```
Selection.CanInsertList
```

Example

```
// XMetaL Script Language JSCRIPT:
// Use this code with XMetaL XMAX
// display a list of all items that you can
// insert into the current selection

// pre: myXMControl has previously been defined as an XMetaL XMAX object

function showCanInsert() {
var sel = myXMControl.Selection;
var list = sel.CanInsertList;
var num = list.Count;
var str = "";

for (i=0; i<num; i++) {
str+= list.Item(i).Name;
str += "\n";
```

```
}
ActiveDocument.Host.Alert(str);
}
```

Example

```
// XMetaL Script Language JSCRIPT:
// Use this code with XMetaL Author
// display a list of all items that you can
// insert into the current selection

function showCanInsert() {
var sel = Selection;
var list = sel.CanInsertList;
var num = list.Count;
var str = "";

for (i=0; i<num; i++) {
str+= list.Item(i).Name;
str += "\n";
}
ActiveDocument.Host.Alert(str);
}
showCanInsert();
```

CanInsertNS

Indicates whether the specified element, which is local to the namespace specified, can replace the selection. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanInsert("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.CanInsert("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanInsertNS("http://www.mydomainname.com", "P")) {
Selection.InsertElementNS("http://www.mydomainname.com", "P");
} else {
```

```
ActiveDocument.Host.Alert("Cannot insert P here.");
}
```

CanInsertText
Indicates whether text can be inserted at the selection.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanInsertText;
```

Usage in VBScript

```
Selection_object.CanInsertText
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanInsertText) {
Selection.Text="OK to insert text here!";
} else {
ActiveDocument.Host.Alert("Cannot insert text here!");
}
```

CanJoinElementToPreceding
Returns True if the current element can be joined to the preceding element, and False otherwise.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanJoinElementToPreceding;
```

Usage in VBScript

```
Selection_object.CanJoinElementToPreceding
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanJoinElementToPreceding()) {
ActiveDocument.Host.Alert("Can be joined without invalidating.");
}
```

CanMergeTable

Property that merges two adjacent tables into one table.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Description

Returns true if two adjacent tables can be merged into one table.

If withTableBelow is true, it indicates that the table with the selection will be merged with the table after it.

If withTableBelow is false, it indicates that the table with the selection will be merged with the table before it.

Usage in JScript

```
Selection.CanMergeTable(withTableBelow);
```

Usage in VBScript

```
Selection.CanMergeTable(withTableBelow)
```

Example

```
// XMetaL Script Language JSCRIPT:

var withTableBelow = true;
var ok = Selection.CanMergeTable(withTableBelow);
Application.Alert("CanMergeTable returns " + ok);
```

CanPaste

Indicates whether the specified text can be pasted, using either `Selection.PasteStringWithInterpret` or `Selection.PasteString`.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanPaste("String", [boolInterpret]);
```

The string can contain markup, which is parsed in order to determine whether it can be pasted.

If `boolInterpret` is True, this property tests whether the specified text can be pasted by `Selection.PasteStringWithInterpret`. (This method pastes text as an HTML or CALS table if it is tab-formatted as a table.)

Note: This property returns True in some instances when the resulting pasted markup is not valid. Only the top-level, or containing, elements are tested for validity. If you want to perform a more thorough check, use `Selection.CanPasteStrict`.

If `boolInterpret` is False, this property indicates whether the specified text can be pasted over the selection using a basic paste, as done with `Selection.PasteString`. That is, if the paste would not be valid, XMetaL does not attempt to rearrange the markup in order to make it valid; the paste is simply disallowed.

Usage in VBScript

```
Selection_object.CanPaste("String", [boolInterpret])
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanPaste("<P></P>")) {
Selection.PasteString("<P></P>");
} else {
ActiveDocument.Host.Alert("Cannot paste \"<P></P>\"");
}
```

CanPasteStrict

Indicates whether the specified text can be pasted, using either `Selection.PasteStringWithInterpret` or `Selection.PasteString`. When performing the check, XMetaL tests all levels of the markup on the clipboard, not just the parent (container) elements. If you want to perform a less strict validation, use `Selection.CanPaste`.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanPasteStrict("String", [boolInterpret]);
```

The string can contain markup, which is parsed in order to determine whether it can be pasted.

If `boolInterpret` is True, this property tests whether the specified text can be pasted by `Selection.PasteStringWithInterpret`. (This method pastes text as an HTML or CALS table if it is tab-formatted as a table.)

If `boolInterpret` is False, this property indicates whether the specified text can be pasted over the selection using a basic paste, as done with `Selection.PasteString`. That is, if the paste would not be valid, XMetaL does not attempt to rearrange the markup in order to make it valid; the paste is simply disallowed.

Usage in VBScript

```
Selection_object.CanPasteStrict("String", [boolInterpret])
```

Example

```
// XMetaL Script Language JSCRIPT:
if (Selection.CanPasteStrict("<P><P></P></P>")) {
Selection.PasteString("<P><P></P></P>");
} else {
ActiveDocument.Host.Alert("Cannot paste \"<P><P></P></P>\"");
}
```

CanRemoveContainerTags

Indicates whether the selection's container (but not necessarily its contents) can validly be deleted.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanRemoveContainerTags;
```


Usage in VBScript

```
Selection_object.CanRemoveContainerTags
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanRemoveContainerTags) {
Selection.RemoveContainerTags();
}
```

CanSplitContainer
Indicates whether the selection's container can be split into the element specified. Returns True if the split results in a valid document, and False otherwise. If an element is not specified, this property checks if the selection's container can be split into the same element. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanSplitContainer(["strElementName"]);
```

Usage in VBScript

```
Selection_object.CanSplitContainer(["strElementName"])
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanSplitContainer("Para")) {
ActiveDocument.Host.Alert("Can be split without invalidating.");
}
```

CanSplitContainerNS
Indicates whether the selection's container can be split into the element specified, which is local to the namespace specified. Returns True if the split results in a valid document, and False otherwise. If an element and namespace are not specified, this property checks if the selection's container can be split into the same element. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanSplitContainerNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.CanSplitContainerNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanSplitContainer("http://www.mydomainname.com", "Para"))
{
ActiveDocument.Host.Alert("Can be split without invalidating.");
}
```

CanSplitTable

Boolean property that returns true if the table under the selection can be split into two for the given direction.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read-only

Returns

Boolean

Description

Returns true if the table under the selection can be split into two. If belowThisRow is true, it indicates that the table row contains the current selection will be in the 1nd table after splitting. If belowThisRow is false, it indicates that the table row contains the current selection will be in the 2nd table after splitting.

Usage in JScript

```
Selection.CanSplitTable(belowThisRow);
```

Usage in VBScript

```
Selection.CanSplitTable(belowThisRow)
```


Example

```
// XMetaL Script Language JSCRIPT:

var belowThisRow = true;
var ok = Selection.CanSplitTable(belowThisRow);
Application.Alert("CanSplitTable returns " + ok);
```

CanSplitTableGroup

Indicates whether a table group under the selection can be split into two table groups.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Description

Returns true if the table group under the selection can be split into two table groups.

If belowThisRow is true, it indicates that the table row contains the current selection will be in the 1st table group after splitting.

If belowThisRow is false, it indicates that the table row contains the current selection will be in the 2nd table group after splitting.

For HTML table, this API is as same as *SplitTable* on page 425.

Usage in JScript

```
Selection.CanSplitTableGroup(belowThisRow);
```

Usage in VBScript

```
Selection.CanSplitTableGroup(belowThisRow)
```

Example

```
// XMetaL Script Language JSCRIPT:

var belowThisRow = true;
var ok = Selection.CanSplitTableGroup(belowThisRow);
Application.Alert("CanSplitTableGroup returns " + ok);
```

CanSurround

Indicates whether the selection can be surrounded by the element specified. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanSurround("strElementName");
```

Usage in VBScript

```
Selection_object.CanSurround("strElementName")
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanSurround("P")) {
Selection.Surround("P");
} else {
ActiveDocument.Host.Alert("Cannot surround with P.");
}
```

CanSurroundNS

Indicates whether the selection can be surrounded by the element specified, which is local to the namespace specified. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.CanSurroundNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.CanSurroundNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.CanSurroundNS("http://www.mydomainname.com", "P")) {
Selection.SurroundNS("http://www.mydomainname.com","P");
} else {
```



```
ActiveDocument.Host.Alert("Cannot surround with P.");
}
```

Cells

Returns a DOMNodeList of DOMElement objects, representing all cells in the current mult-cell table selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
Selection_object.Cells;
```

Usage in VBScript

```
Selection_object.Cells
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.IsMultiCellSelection) {
  var cells = Selection.Cells;
  ActiveDocument.Host.Alert(cells.length);
}
```

CollapsedContainerTags

Collapses or expands the tags of the selection's immediate container; indicates whether the tags of the selection's container are collapsed. Setting this property to True moves a Selection (but not a Range) object to a point just after the collapsed tag. This property applies to the Tags On and Normal views, and the structure view.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = Selection_object.CollapsedContainerTags;
Selection_object.CollapsedContainerTags = boolState;
```

Usage in VBScript

```
dim boolState = Selection_object.CollapsedContainerTags
Selection_object.CollapsedContainerTags = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
// Change state

if (Selection.CollapsedContainerTags) {
// Expand tags
Selection.CollapsedContainerTags = false;
} else {
// Collapse tags
Selection.CollapsedContainerTags = true;
}
```

CollapsedTags

Indicates whether the selection is anywhere inside a container whose tags are collapsed (not necessarily the selection's immediate container).

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
var boolTagsAreCollapsed = Selection_object.CollapsedTags;
```

Usage in VBScript

```
dim boolTagsAreCollapsed = Selection_object.CollapsedTags
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.Host.Alert(Selection.CollapsedTags);
```

ContainerAttribute

Sets or returns the value of the attribute specified for the selection's container (if the container is an element). This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
Selection_object.ContainerAttribute("strAttrName");
```

Usage in VBScript

```
Selection_object.ContainerAttribute("strAttrName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// This returns the value of the "Id" attribute for the
// selection's container.
ActiveDocument.Host.Alert(Selection.ContainerAttribute("Id"));
// This sets the value of the "Id" attribute for the
// selection's container
Selection.ContainerAttribute("Id")="Newvalue";
ActiveDocument.Host.Alert(Selection.ContainerAttribute("Id"));
```

ContainerAttributeNS
Sets or returns the value of the attribute specified that is local to the namespace specified, for the selection's container (if the container is an element). This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
Selection_object.ContainerAttributeNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.ContainerAttributeNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// This returns the value of the "Id" attribute for the
// selection's container.
ActiveDocument.Host.Alert(Selection.ContainerAttributeNS("http://www.mydomainname.com",
```

```
 "Id"));
// This sets the value of the "Id" attribute for the
// selection's container
Selection.ContainerAttributeNS("http://www.mydomainname.com",
"Id")="Newvalue";
ActiveDocument.Host.Alert(Selection.ContainerAttributeNS("http://www.mydomainname.com",
 "Id"));
```

ContainerName
Sets or returns the name of the container that contains the selection. If the container is an element, the element name is returned. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Description

Other container names are:

- .PROCINS = processing instruction
- .MARKSEC = CDATA section or marked section
- .COMMENT = comment

Setting a value changes the container's element type (if the change is valid).

Usage in JScript

```
var strName = Selection_object.ContainerName;
Selection_object.ContainerName = strName;
```

Usage in VBScript

```
dim strName = Selection_object.ContainerName
Selection_object.ContainerName = strName
```

Example

```
// XMetaL Script Language JSCRIPT:

// display container name
ActiveDocument.Host.Alert(Selection.ContainerName);
// change the container to Note
Selection.ContainerName="Note";
```

ContainerNameNS
Sets or returns the universal name of the container that contains the selection. If the container is an element, the universal element name is returned. An example of a valid universal element name is

`{http://www.mydomainname.com}P`. Universal element names always contain the namespace name URI in curly braces followed by the local name of the element. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Description

Other container names are:

- .PROCINS = processing instruction
- .MARKSEC = CDATA section or marked section
- .COMMENT = comment

Setting a value changes the container's element type (if the change is valid).

Usage in JScript

```
Selection_object.ContainerName = "{http://www.mydomainname.com}P";
var myString = Selection_object.ContainerName;
```

Usage in VBScript

```
Selection_object.ContainerName = "{http://www.mydomainname.com}P"
dim myString = Selection_object.ContainerName
```

Example

```
// XMetaL Script Language JSCRIPT:

// display container name
ActiveDocument.Host.Alert(Selection.ContainerNameNS);
// change the container to Note
Selection.ContainerNameNS="{http://www.mydomainname.com}Note";
```

ContainerNode

Returns the DOMNode corresponding to the selection's container.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNode object

Usage in JScript

```
Selection_object.ContainerNode;
```

Usage in VBScript

```
Selection_object.ContainerNode
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.Host.Alert(Selection.ContainerNode.nodeName);
```

ContainerStyle

Sets or gets the CSS (cascading) style for the selection's container. This formatting persists as long as the document is open in Normal or Tags On view. If the document is closed, or the view is switched to Plain Text, the formatting is lost. This property does not depend on or modify style attributes (such as xhtml:style).

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var strStyle = Selection_object.ContainerStyle;
Selection_object.ContainerStyle = "strStyle";
```

Usage in VBScript

```
dim strStyle = Selection_object.ContainerStyle
Selection_object.ContainerStyle = "strStyle"
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.ContainerStyle = "color:red; display:block";
```

Contains

Indicates whether the selection completely contains the specified Range.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.Contains(Range_object, [boolIncludesAsEdge]);
```

If `boolIncludesAsEdge` is True, then this property returns True if the Range is an insertion point at the left or right edge of the selection. The default value for `boolIncludesAsEdge` is False.

Usage in VBScript

```
Selection_object.Contains(Range_object, [boolIncludesAsEdge])
```

Example

```
// XMetaL Script Language JSCRIPT:
// True if the selection is as follows:
// |abcdefgh<XREF>ijk|
// Use a range to represent the selection

var rng1 = ActiveDocument.Range;
var rng2 = rng1.Duplicate;
rng2.Collapse();
moved = rng2.MoveToElement("XREF");

if (moved && rng1.Contains(rng2)) {
ActiveDocument.Host.Alert("Selection contains XREF");
}
```

Document

Returns the Document object that contains the selection.

Applies To

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Document object

Usage in JScript

```
Selection_object.Document;
```

Usage in VBScript

```
Selection_object.Document
```

Example

```
// XMetaL Script Language JSCRIPT:

var curDoc = Selection.Document;
ActiveDocument.Host.Alert(curDoc.FullName);
```

Duplicate

Returns a Range object that is a copy of the original Selection or Range object. Any changes to the original selection or range do not affect the copy.

Applies To

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Range object

Usage in JScript

```
Selection_object.Duplicate;
```

Usage in VBScript

```
Selection_object.Duplicate
```

Example

```
// XMetaL Script Language JSCRIPT:

var rng = ActiveDocument.Range;
var rng1 = rng;
var rng2 = rng.Duplicate;
//The following statement changes rng *and* rng1,
//but does not change rng2
rng.SelectAfterContainer();
```

ElementAttribute

Sets or returns value of the specified attribute from the specified element (if it is the selection's container or other ancestor). This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var strAttVal = Selection_object.ElementAttribute("strAttrName", "strElementName",
[longSkipNum]);
Selection_object.ElementAttribute("strAttrName", "strElementName", [longSkipNum]) =
strAttVal;
```

If `longSkipNum` is specified, that number of elements of the same name are skipped (useful for nested tables).

Usage in VBScript

```
dim strAttVal = Selection_object.ElementAttribute("strAttrname", "strElementName",
[longSkipNum])
Selection_object.ElementAttribute("strAttrname", "strElementName", [longSkipNum]) =
strAttVal
```

Example

```
// XMetaL Script Language JSCRIPT:
// Return the value of the WIDTH attribute
// from the TABLE element, then convert to pixels
var attr = Selection.ElementAttribute("WIDTH", "TABLE");
Selection.ElementAttribute("WIDTH", "TABLE") = attr*100;
```

ElementAttributeNS

Sets or returns value of the specified attribute in a specified namespace from the specified element (if it is the selection's container or other ancestor) in a specified namespace. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var myAttribute = Selection_object.elementAttribute("strAttrNamespace", "strAttrName",
"steElementNamespace", "strElementName", [longSkipNum=0]);
Selection_object.elementAttribute("strAttrNamespace", "strAttrName", "strElementNamespace",
 "strElementName", [longSkipNum=0]) = myAttribute;
```

If `longSkipNum` is specified, that number of elements of the same name are skipped (useful for nested tables).

Usage in VBScript

```
dim myAttribute = Selection_object.elementAttribute("strAttrNamespace", "strAttrName",
"steElementNamespace", "strElementName", [longSkipNum=0])
Selection_object.elementAttribute("strAttrNamespace", "strAttrName", "strElementNamespace",
 "strElementName", [longSkipNum=0]) = myAttribute
```

Example

```
// XMetaL Script Language JSCRIPT:
// Return the value of the WIDTH attribute
// from the TABLE element, then convert to pixels
var attr =
Selection.elementAttributeNS("http://www.mydomainname.com/width/", "WIDTH",
 "http://www.mydomainname.com/table/","TABLE");
Selection.elementAttributeNS("http://www.mydomainname.com/width/","WIDTH",
 "http://www.mydomainname.com/table/","TABLE") = attr*100;
```

ElementName
Returns the name of the ancestor element of the current selection, a specified number of levels up. A value of 0 returns the name of selection's container. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Selection_object.ElementName(longLevels);
```

Usage in VBScript

```
Selection_object.ElementName(longLevels)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Return the name of the selection container's
// parent element.

ActiveDocument.Host.Alert(Selection.ElementName(1));
```

ElementNameNS
Returns the universal name of the ancestor element of the current selection, a specified number of levels up. A value of 0 returns the universal name of selection's container. An example of a valid universal name is `{http://www.mydomainname.com}P`. Universal names always contain the namespace name URI in curly braces followed by the local name of the element. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var myString = Selection_object.ElementNameNS(longLevels);
```

Usage in VBScript

```
dim myString = Selection_object.ElementNameNS(longLevels)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Return the universal name of the selection container's
// parent element.

ActiveDocument.Host.Alert(Selection.ElementNameNS(1));
```

Find

Provides access to the Find object. This enables you to perform find and replace operations in the document. Alerts issued can be suppressed with `Application.DisplayAlerts` or `DocumentHost.DisplayAlerts`.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Find object

Usage in JScript

```
Selection_object.Find;
```

Usage in VBScript

```
Selection_object.Find
```

Example

```
// XMetaL Script Language JSCRIPT:
// locates and selects "some text"
Selection.Find.Execute("some text");
```

hasAttribute
Indicates whether an attribute specified has been declared in the DTD for the selection's container (the attribute may or may not have a value). This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.hasAttribute("strAttrName");
```

Usage in VBScript

```
Selection_object.hasAttribute("strAttrName")
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.hasAttribute("ID")) {
ActiveDocument.Host.Alert("Identifier element.");
}
```

hasAttributeNS
Indicates whether an attribute specified has been declared in the schema for the selection's container (the attribute may or may not have a value). The attribute specified is local to the namespace specified. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.hasAttributeNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.hasAttributeNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.hasAttributeNS("http://www.mydomainname.com", "ID")) {
ActiveDocument.Host.Alert("Identifier element.");
}
```

Hidden

Returns True if any ancestor of the selection is hidden.

Applies To

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
var boolIsSelectionObjectHidden = Selection_object.Hidden;
```

Usage in VBScript

```
dim boolIsSelectionObjectHidden = Selection_object.Hidden
```

Example

```
// XMetaL Script Language JSCRIPT:
// If the selection is hidden, set an attribute
// of its container
var r = ActiveDocument.Range;

if (r.Hidden) {
r.ContainerAttribute("hidden") = "y";
}
```

HiddenContainer

Hides or displays the selection's container, and returns the container's 'hidden' state.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = Selection_object.HiddenContainer;
Selection_object.HiddenContainer = boolState;
```

Usage in VBScript

```
dim boolState = Selection_object.HiddenContainer
Selection_object.HiddenContainer = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
// If the container is hidden, make it visible
var r = ActiveDocument.Range;

if (r.HiddenContainer) {
r.HiddenContainer = false;
}
```

HorizontalAlignment

This property is unsupported.

InContextOfType

Allows you to find the context of the current selection. For example, if you want to replace the default behavior of the Tab key, you need to know if the current selection is in a context where the Tab key has any special behaviors (such as in a table or list). The determination of the context depends on information that is in the customization file. The settings are determined by examining the (Treat As Type) property of the customization file.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection.InContextOfType ("ContextName");
```

The parameter `strContextName` is the name of the context that you are testing to determine whether or not you are in it. Only the following names can be used in `strContextName`:

- `TableCell` - True if the selection is anywhere inside a table cell

- `Table` - True if the selection is anywhere inside a table
- `List` - True if the selection is anywhere inside a list
- `ListItem` - True if the selection is anywhere inside a list item
- `ListTabSpot` - True if the selection is where a Tab would demote or shift+Tab would promote a list item(s)
- `ListDemoteTabSpot` - True if the selection is where a Tab would demote a list item(s)
- `Graphic` - True if the selection is in an element that has a image representation (Image, InPlaceControl, XFT, HTML input element, or empty element icon)
- `Image` - True if the selection is in an element with type "Treat As Image" in CTM (customization dialog)
- `InPlaceControl` - True if the selection is in a InPlaceControl
- `XFT` - True if the selection is in a XFT forms control
- `ContainerIsBlock` - True if the selection's container has a block style, False if the container of the selection has an inline style

Usage in VBScript

```
Selection.InContextOfType ("ContextName")
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Capture the Tab key and give it back if it
' is appropriate. tblcell = Selection.InContextOfType("TableCell")
lstspot = Selection.InContextOfType("ListTabSpot")
lstspot = lstspot Or Selection.InContextOfType("ListDemoteTabSpot")

If tblcell Or lstspot Then
Selection.TypeTab(False)
Else
MsgBox("Do new behaviour")
End If
```

IsAdjacent

Indicates whether the selection is adjacent to the specified Range.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.IsAdjacent(Range_object);
```

Usage in VBScript

```
Selection_object.IsAdjacent(Range_object)
```

Example

```
// XMetaL Script Language JSCRIPT:
// True if the selection is as follows:
// |abcdefgh|<XREF>
// Use a range to represent the selection

var rng1 = ActiveDocument.Range;
var rng2 = rng1.Duplicate;
rng2.Collapse();
moved = rng2.MoveToElement("XREF");

if (moved) {
rng2.SelectElement();

if (rng1.IsAdjacent(rng2)) {
ActiveDocument.Host.Alert("Selection is adjacent to XREF");
}
}
```

IsEqual

Indicates whether the selection is equal to the specified Range.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.IsEqual(Range_object);
```

Usage in VBScript

```
Selection_object.IsEqual(Range_object)
```

Example

```
// XMetaL Script Language JSCRIPT:
// True if the selection is as follows:
// |MondoBongo|
// Use a range to represent the selection

var rng1 = ActiveDocument.Range;
var rng2 = rng1.Duplicate;
rng2.Collapse();
moved = rng2.Find.Execute("MondoBongo");

if (moved && rng1.IsEqual(rng2)) {
ActiveDocument.Host.Alert("Selection is equal to "MondoBongo"");
}
```


IsGreaterThan

Indicates whether the left end of the selection is greater than (that is, is to the right of) the left end of the specified Range.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.IsGreaterThan(Range_object, [boolOrEqual]);
```

If `boolOrEqual` is True, then this property returns True if the left end of the selection is equal to the left edge of the Range.

Usage in VBScript

```
Selection_object.IsGreaterThan(Range_object, [boolOrEqual])
```

Example

```
//XMetaL Script Language JSCRIPT:
// True if the selection is as follows:
// |abcdefg|hijkl
// Use a range to represent the selection

var rng1 = ActiveDocument.Range;
var rng2 = rng1.Duplicate;
rng2.Collapse();
moved = rng2.Find.Execute("fghijkl");

if (moved && rng2.IsGreaterThan(rng1)) {
ActiveDocument.Host.Alert("Rng2 is greater than the selection");
}
```

IsInsertionPoint

True if the selection is an insertion point (that is, it has no content).

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.IsInsertionPoint;
```

Usage in VBScript

```
Selection_object.IsInsertionPoint
```

Example

```
// XMetaL Script Language JSCRIPT:
var rng = ActiveDocument.Range;

if (!rng.IsInsertionPoint) {
rng.Collapse(0);
}
rng.PasteString("This text should not overwrite any existing content
because the insertion point has been collapsed.");
```

IsLessThan

Indicates whether the left end of the selection is less than (that is, to the left of) the left end of the specified Range.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.IsLessThan(Range_object, [boolOrEqual]);
```

If `boolOrEqual` is True, then this property returns True if the left end of the selection is equal to the left edge of the Range.

Usage in VBScript

```
Selection_object.IsLessThan(Range_object, [boolOrEqual])
```

Example

```
// XMetaL Script Language JSCRIPT:
// True if the selection is as follows:
// |abcdefg|hijkl
// Use a range to represent the selection


var rng1 = ActiveDocument.Range;
var rng2 = rng1.Duplicate;
rng2.Collapse();
moved = rng2.Find.Execute("fghijkl");
```



```
if (moved && rng1.IsLessThan(rng2)) {
ActiveDocument.Host.Alert("Selection is less than Rng2");
}
```

IsMultiCellSelection
True if the selection is a multi-cell table selection.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.IsMultiCellSelection;
```

Usage in VBScript

```
Selection_object.IsMultiCellSelection
```

Example

```
// XMetaL Script Language JSCRIPT:
var rng = ActiveDocument.Range;

if (rng.IsMultiCellSelection) {
  var cells = rng.Cells;
}
```

IsParentElement
Indicates whether the element specified is an ancestor of the selection, at any level. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.IsParentElement("strElementName");
```

Usage in VBScript

```
Selection_object.IsParentElement("strElementName")
```

Example

```
// XMetaL Script Language JSCRIPT:
var isparent;
isparent = Selection.IsParentElement("Appendix");
```

IsParentElementNS
Indicates whether the element **strLocalName**, an element local to the namespace named **strNamespaceNameURI**, is an ancestor of the selection at any level. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.IsParentElementNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.IsParentElementNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:
var isparent;
isparent = Selection.IsParentElementNS("http://www.mydomainname.com",
"Appendix");
```

IsValid
Indicates whether the selection is valid according to the DTD or schema of the current document. Displays no messages to the user. Returns True if the selection is an insertion point.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.IsValid;
```

Usage in VBScript

```
Selection_object.IsValid
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.IsValid) {
ActiveDocument.Host.Alert("Selection is valid.");
} else {
ActiveDocument.Host.Alert("Selection is not valid.");
}
```

NonRemovableContainer

Returns or sets the selection's container's to be removable or non-removable. Setting a container to be non-removable prevents the container from being deleted. Non-removable status persists only as long as the document is displayed in Normal or Tags On view. If you close the document or switch to Plain Text view, the entire document becomes writable again. You may want to create a macro that sets certain containers as non-removable automatically when the file is opened or when the user switches from Plain Text view.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
Selection_object.NonRemovableContainer;
```

Usage in VBScript

```
Selection_object.NonRemovableContainer
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Check to see if the selection's container
' is removable or not. If it is removable, make
' it non-removable.

If Selection.NonRemovableContainer = False then
```

```
Selection.NonRemovableContainer = True
End If
```

Overlaps
Indicates whether the selection overlaps the specified Range.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.Overlaps(Range_object);
```

Usage in VBScript

```
Selection_object.Overlaps(Range_object)
```

Example

```
// XMetaL Script Language JSCRIPT:
// True if the selection is as follows:
// |abcdefg|hijkl
// Use a range to represent the selection

var rng1 = ActiveDocument.Range;
var rng2 = rng1.Duplicate;
rng2.Collapse();
moved = rng2.Find.Execute("fghijkl");

if (moved && rng1.Overlaps(rng2)) {
ActiveDocument.Host.Alert("Selection overlaps Rng2");
}
```

ReadOnly
Returns True if the selection's container, or one of its ancestors, is read-only.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
Selection_object.ReadOnly;
```

Usage in VBScript

```
Selection_object.ReadOnly
```

Example

```
// XMetaL Script Language JSCRIPT:
if (Selection.ReadOnly) {
ActiveDocument.Host.Alert("The section is read-only");
} else {
ActiveDocument.Host.Alert("The section is writable");
}
```

ReadOnlyContainer

Returns or sets the selection's container's read-only flag. Setting a container to be read-only prevents the container's contents and attributes from being modified. Read-only status persists only as long as the document is displayed in Normal or Tags On view. If you close the document or switch to Plain Text view, the entire document becomes writable again. You may want to create a macro that sets the read-only flag for certain containers automatically when the file is opened or when the user switches from Plain Text view.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
Selection_object.ReadOnlyContainer;
```

Usage in VBScript

```
Selection_object.ReadOnlyContainer
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.ContainerAttribute("Status")=="RO") {
Selection.ReadOnlyContainer = true;
}
```

Style

Returns or sets the style element for the selection. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
Selection_object.Style;
```

Usage in VBScript

```
Selection_object.Style
```

Example

```
// XMetaL Script Language JSCRIPT:
// display the string of the style element of the selection
ActiveDocument.Host.Alert(Selection.Style);
```

StyleElement

Returns the element's description (that is, It returns what is shown in the style element drop-down). This API is designed to work with: `Application.StyleElements`, `Application.StyleElementName`, `Application.StyleElementType`, `Application.StyleElementTypeNS`. By calling these APIs, a developer can create customizations to change the tag of one element into another element. Making this customization available in the XMetaL interface ensures that users can change the style of a current element even, in some cases, when the default behavior of XMetaL does not or cannot change the style. This should be used only in the On_Style_Element macro. Do not set this property during the On_Style_Element macro.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
Selection_object.StyleElement;
```

Usage in VBScript

```
Selection_object.StyleElement
```


Example

```
// XMetaL Script Language JSCRIPT:
// display the style of the selection
ActiveDocument.Host.Alert(Selection.StyleElement);
```

StyleNS

Returns or sets the universal name for the style element for the selection. An example of a valid universal element name is `{http://www.mydomainname.com}P`. Universal element names always contain the namespace name URI in curly braces followed by the local name of the element. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
Selection_object.StyleNS = "{http://www.mydomainname.com}P";
var myString = Selection_object.StyleNS;
```

Usage in VBScript

```
Selection_object.StyleNS = "{http://www.mydomainname.com}P"
dim myString = Selection_object.StyleNS
```

Example

```
// XMetaL Script Language JSCRIPT:
// display the universal name of the style element of the selection
ActiveDocument.Host.Alert(Selection.StyleNS);
```

SVCollapsedContainer

Returns or sets the ‘collapsed’ state in the structure view of the selection’s container. The collapsed state of an ancestor of the current container does not affect the value returned: for example, if the current container is not collapsed, but an ancestor is collapsed, this property returns False. It always returns False for a container that is not collapsible (because it has no sub-elements).

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = Selection_object.SVCollapsedContainer;
Selection_object.SVCollapsedContainer = boolState;
```

Usage in VBScript

```
dim boolState = Selection_object.SVCollapsedContainer
Selection_object.SVCollapsedContainer = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
// If the container is collapsed in the structure
// view, expand it
var r = ActiveDocument.Range;

if (r.SVCollapsedContainer) {
r.SVCollapsedContainer = false;
}
```

Text

Provides access to the text in the document's selection. If the selection contains any tag icons, they are returned as text. Assigning a string to `Selection_object.Text` replaces the selection, if the DTD or Schema allows it. If you want to preserve Revision Marks, do not use this property; instead, use `Selection.TextWithRM`.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var strSelectionText = Selection_object.Text;
Selection_object.Text = strSelectionText;
```

Usage in VBScript

```
dim strSelectionText = Selection_object.Text
Selection_object.Text = strSelectionText
```

Example

```
// XMetaL Script Language JSCRIPT:
// Store entire document (text and tags) in a string
```



```
// Note: does not include DOCTYPE
var rng = ActiveDocument.Range;
rng.SelectAll();
var docString = rng.Text;
// Store current element (text and tags) in a string
rng = ActiveDocument.Range;
rng.SelectElement();
var elemString = rng.Text;
```

TextWithRM
Provides access to the text, including Revision Marks, in the document's selection. This property is similar to `Selection.Text`, except that it preserves the Revision Marking Processing Instructions. Assigning a string to `Selection_object.TextWithRM` replaces the selection, if the DTD or schema allows it.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var strSelectionTextWithRM = Selection_object.TextWithRM;
Selection_object.TextWithRM = strSelectionTextWithRM;
```

Usage in VBScript

```
dim strSelectionTextWithRM = Selection_object.TextWithRM
Selection_object.TextWithRM = strSelectionTextWithRM
```

Example

```
// XMetaL Script Language JSCRIPT:
// Store entire document (text, tags and revision marks) in a string
// Note: does not include DOCTYPE
var rng = ActiveDocument.Range;
rng.SelectAll();
var docString = rng.TextWithRM;
// Store current element (text, tags and revision marks) in a string
rng = ActiveDocument.Range;
rng.SelectElement();
var elemString = rng.TextWithRM;
```

ValidationErrorList
Returns a ValidationErrorList containing the list of errors found in the most recent call to `Selection.Validate`. If there is no ValidationErrorList to return, this property returns null. If changes are made to the selection after `Selection.Validate` is called, the ValidationErrorList returned may contain errors that no longer exist in the document. This can cause unpredictable behavior when calling properties and methods on the ValidationErrorList. You should manipulate this property only immediately after a call to `Selection.Validate`.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

ValidationErrorList object

Usage in JScript

```
Selection.ValidationErrorList;
```

Usage in VBScript

```
Selection.ValidationErrorList
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
// web page.
// Validates the selection, and displays a list of validation errors.
// When you click OK, you go to the error numbered gErrNum.
// gErrNum persists (if the script is run in an Internet Explorer web
page),
// and you can run validate() a number of times to visit each
// validation error in the document in sequence.
// The script also provides additional information about some errors.

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myXMControl" HEIGHT="500" WIDTH="500">
</OBJECT>

<SCRIPT>
// define global variable
var gErrnum = 0;

function assert(bool) {
if (bool == false) {
ActiveDocument.Host.Alert("A programming error occured.");
}
}

function validate() {
var doc = myXMControl.Document;
if (doc) {
var sel = Selection;
sel.Validate();
var vel = sel.ValidationErrorList;
if (vel) {
if (vel.Count == 0) { // valid document
assert(sel.IsValid);
ActiveDocument.Host.Alert("The document is valid.");
} else { // one or more validation errors
assert(!sel.IsValid);
var msg = "Errors:\n\nItem\tLevel\tType\tMessage\n";
var index = 1;
var ve = vel.Item(index++);
while (ve) {
msg += index-1
+ "\t" + ve.ErrorLevel
```



```
+ "\t" + ve.ErrorType
+ "\t" + ve.ErrorMessage;
// if (ve.ErrorType == vetEntUndefined) {
if (ve.ErrorType == 21) {
ActiveDocument.Host.Alert("You have an undefined entity.");
}
// if (ve.ErrorLevel == velFatalError) {
if (ve.ErrorLevel == 3) {
ActiveDocument.Host.Alert("A fatal error occurred.");
}
ve = vel.Item(index++);
msg += "\n";
}
gErrNum++;
if (gErrNum > vel.Count) {
gErrNum = 1; // go back to the first error
}
ve = vel.Item(gErrNum);
msg += "\nPress OK to go to error " + gErrNum + " (Item " + gErrNum + ")";
ActiveDocument.Host.Alert(msg);

// There are 2 ways to go to the error:
//ValidationErrorList.GoToError(Index)
//ValidationError.Range.Select()
var doItWithGoToError = true;
if (doItWithGoToError) {
vel.GoToError(gErrNum);
} else {
ve.Range.Select();
}
}
}
}
}
</SCRIPT>
```

WritePermittedContainer
Overrides the behavior of a read-only container. This is useful for creating read-write islands in the middle of read-only content. Although you may set the node to be non-removable (using `Selection.NonRemovableContainer`) users can still edit content inside the containers, even though they cannot remove the containers. Setting `Selection.WritePermittedContainer` to True resets `Selection.ReadOnlyContainer`, if set, on the same node. If the document is closed, or the view is switched to Plain Text, the node information is lost, including the read-write information set up using this method.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolIsWritePermittedContainer = Selection_object.WritePermittedContainer;
Selection_object.WritePermittedContainer = true;
```

Usage in VBScript

```
dim boolIsWritePermittedContainer = Selection_object.WritePermittedContainer
Selection_object.WritePermittedContainer = true
```

Example

```
// XMetaL Script Language JSCRIPT:
// Assume that the user has created a Para
// with an Emphasis element in it. Both contain
// content, and the Para element is read-only.

// The Emphasis element has "inherited" the
// read-only property. The cursor is in the
// Emphasis element:
Selection.WritePermittedContainer = true;

// Now you can edit the content inside the
// Emphasis element, although the Para element
// remains read-only.
```

Methods

AcceptAllChanges

Accepts all changes in the indicated selection. There may be times when you want to accept all changes made (using change tracking) within a selected block. By first setting the Selection to the text marked as changed, you can then use the AcceptAllChanges method to accept all changes in the indicated selection. If the selection is only an insertion point (as is allowed as a valid selection) then there is no effect. You can also use the RejectAllChanges method. It is recommended that you use the Range interface with this method instead of the Selection interface: `Document.Range.AcceptAllChanges`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.AcceptAllChanges();
```

Usage in VBScript

```
Selection_object.AcceptAllChanges()
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a selection and accept all changes made within it

var sel=Application.Selection; //Selected text must contain change-tracking
 changes
sel.AcceptAllChanges();

// Now observe the selection to see all changes were accepted
```


BasicPaste

Performs a simple paste of the contents of the clipboard over the selection, if this operation is valid according to the DTD or schema.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.BasicPaste();
```

Usage in VBScript

```
Selection_object.BasicPaste
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste the clipboard if the current element
// is a P.

if (Selection.ElementName(0)=="P") {
Selection.BasicPaste();
}
```

Collapse

Collapses the selection to an insertion point, located at its start or end point, depending on the value specified.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.Collapse(intCollapseDirection);
```

Collapsing to the start point is the default. Allowed values are:

- `sqCollapseEnd` (0) = collapse to end point
- `sqCollapseStart` (1) = collapse to start point

Usage in VBScript

```
Selection_object.Collapse intCollapseDirection
```

Example

```
// XMetaL Script Language JSCRIPT:
// move insertion point to the end of the
// current selection
Selection.Collapse(0);
```

ContractCell
Contracts the table cell containing the selection, from the specified direction.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection.ContractCell(intDirection);
```

Allowed values are:

- `sqContractFromLeft` (0)= contract from the left
- `sqContractFromTop` (1)= contract from the top
- `sqContractFromRight` (2)= contract from the right
- `sqContractFromBottom` (3)= contract from the bottom

Usage in VBScript

```
Selection.ContractCell intDirection
```

Example

```
//XMetaL Script Language JSCRIPT:

// contract the selected table cell from the right if possible
if (Selection.InContextOfType("TableCell") {
Selection.ContractCell(2);
}
```

Copy
Copies the selection to the clipboard.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.Copy();
```

Usage in VBScript

```
Selection_object.Copy
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL Author
// duplicates selected text (if the selection is text only)

Selection.Copy();
if (Application.Clipboard.HasText) {
Selection.Paste();
}
```

Cut

Cuts the selection, and places it in the clipboard.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.Cut();
```

Usage in VBScript

```
Selection_object.Cut
```

Example

```
//XMetaL Script Lanugage JSCRIPT:
if (Selection.CanDelete) {
Selection.Cut();
}
```

Delete

Deletes the selection; does not change the clipboard.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.Delete();
```

Usage in VBScript

```
Selection_object.Delete
```

Example

```
//XMetaL Script Lanugage JSCRIPT:
if (Selection.CanDelete) {
Selection.Delete();
}
```

DeleteColumn

Deletes the table column containing the selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.DeleteColumn();
```

Usage in VBScript

```
Selection_object.DeleteColumn
```

Example

```
//XMetaL Script Language JSCRIPT:
if (Selection.InContextOfType("TableCell") && Selection.CanDelete){
Selection.DeleteColumn();
}
```

DeleteRow

Deletes the table row containing the selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.DeleteRow();
```


Usage in VBScript

```
Selection_object.DeleteRow
```

Example

```
//XMetaL Script Language JSCRIPT:
if (Selection.InContextOfType("TableCell") && Selection.CanDelete){
Selection.DeleteColumn();
}
```

DropFile

Drops the specified file at the selection, triggering XMetaL's default on-drop action for that file. This method does not trigger the On_Drop_Files event macro; it can, however, be used in that macro.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.DropFile("strFileName");
```

Usage in VBScript

```
Selection_object.DropFile "strFileName"
```

Example

```
' XMetaL Script Language VBSCRIPT:
' Add this code to On_Drop_Files event macro
' This code will work with XMetaL Author only

Set rng = ActiveDocument.DropPoint
ActiveDocument.Host.Alert "Dropped in element: " + rng.ContainerName

For cnt = 1 To Application.DropFileCount
ActiveDocument.Host.Alert "Dropped file: " + _
Application.DropFileName(cnt)
' Do the default drop action:
rng.DropFile(Application.DropFileName(cnt))
Next
```

EndKey

Moves the selection as if the End key were pressed, in various key combinations.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.EndKey([intLocation], [intSelectionType]);
```

The placement of the selection depends on the arguments provided. The allowed values for `intLocation` are:

- `sqLine` (0)= move the selection to end of the current line (default)
- `sqDocument` (1)= move the selection to the bottom right of the document

The allowed values for `intSelectionType` are:

- `sqMove` (0)= collapse the selection to an insertion point (default)
- `sqExtend` (1)= extend the selection

Usage in VBScript

```
Selection_object.EndKey [intLocation], [intSelectionType]
```

Example

```
// XMetaL Script Language JSCRIPT:
// same as pressing the End key.
Selection.EndKey();

// same as pressing Ctrl+End.
Selection.EndKey(1);

// same as pressing Shift+End.
Selection.EndKey(0, 1);

// same as pressing Ctrl+Shift+End.
// (Selects the entire document)
Selection.EndKey(1, 1);
```

ExtendTo

Extends the selection to include the specified Range, if they both have the same container. Returns a boolean indicating whether the selection was extended.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Selection_object.ExtendTo(Range_object);
```

Usage in VBScript

```
Selection_object.ExtendTo(Range_object)
```


Example

```
//XMetaL Script Language JSCRIPT:
// try to extend the selection to the entire document
if (Selection.ExtendTo(ActiveDocument.Range)) {
ActiveDocument.Host.Alert("Selected entire document");
} else {
ActiveDocument.Host.Alert("Could not select entire document");
}
```

FindInsertLocation

Moves the selection to the next or previous valid location that the element specified can be inserted. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Selection_object.FindInsertLocation(strElemName, [boolSearchForward]);
```

This method searches forward if `boolSearchForward` is True (the default) and backward if `boolSearchForward` is False. Returns True if a valid location was found, and False otherwise.

Usage in VBScript

```
Selection_object.FindInsertLocation(strElemName, [boolSearchForward])
```

Example

```
// XMetaL Script Language JSCRIPT:
var rng = ActiveDocument.Range;

if (rng.FindInsertLocation("Para")) {
rng.InsertElement("Para");
} else {
ActiveDocument.Host.Alert("Insert location not found.");
}
```

FindInsertLocationNS

Moves the selection to the next or previous valid location that the element specified can be inserted. The element is local to the namespace specified. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Selection_object.FindInsertLocationNS("strNamespaceNameURI", "strLocalName",
[boolSearchForward]);
```

This method searches forward if `boolSearchForward` is True (the default) and backward if `boolSearchForward` is False. Returns True if a valid location was found, and False otherwise.

Usage in VBScript

```
Selection_object.FindInsertLocationNS("strNamespaceNameURI", "strLocalName",
[boolSearchForward])
```

Example

```
// XMetaL Script Language JSCRIPT:
var rng = ActiveDocument.Range;

if (rng.FindInsertLocationNS("http://www.mydomainname.com", "Para")) {
rng.InsertElementNS("http://www.mydomainname.com", "Para");
} else {
ActiveDocument.Host.Alert("Insert location not found.");
}
```

FindPasteLocation

Indicates whether a paste can take place. The selection may or may not need to be moved.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Selection_object.FindPasteLocation(strPaste, [boolDirection]);
```

The parameter `strPaste` contains the string to paste (can contain markup). The optional argument `boolSearchForward` indicates direction. If `boolSearchForward` is True, this method searches forward; if `boolSearchForward` is False, it searches backwards.

Usage in VBScript

```
Selection_object.FindPasteLocation(strPaste, [boolDirection])
```

Example

```
' XMetaL Script Language VBScript:
' Use this example in a Journalist.dtd Article
MsgBox(Selection.FindPasteLocation("<Emphasis>abc</Emphasis>", true))
Selection.PasteString "<Emphasis>abc</Emphasis>"
```

GotoNext

Collapses the selection to an insertion point and moves forward it to the next specified item.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.GotoNext(intGotoItem);
```

The allowed values are:

- sqElement (0)= moves the selection to the next element, comment, processing instruction, entity reference, or character reference
- sqTableCell (1)= moves the selection to the next table cell (the current selection must be in a table)
- sqWordStart (2)= moves the selection to the beginning of the next word
- sqWordEnd (3)= moves the selection to the next word end (in the current word or next word)

Usage in VBScript

```
Selection_object.GotoNext intGotoItem
```

Example

```
// XMetaL Script Language JSCRIPT:
// move cursor to the beginning of the next word.

Selection.GotoNext(2);
```

GotoPrevious
The same as GotoNext, but moves backward.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.GotoPrevious(intGotoItem);
```

Usage in VBScript

```
Selection_object.GotoPrevious intGotoItem
```

Example

```
// XMetaL Script Language JSCRIPT:
// move cursor to the end of the previous word.

Selection.GotoPrevious(3);
```

HomeKey

Moves the selection as if the Home key were pressed, in various key combinations. The placement of the selection depends on the arguments provided.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.HomeKey([intLocation], [intSelectionType]);
```

The allowed values for `intLocation` are:

- sqLine (0)= move the selection to start of the current line (default)
- sqDocument (1)= move the selection to the upper left of the document

The allowed values for `intSelectionType` are:

- sqMove (0)= collapse the selection to an insertion point (default)
- sqExtend (1)= extend the selection

Usage in VBScript

```
Selection_object.HomeKey [intLocation], [intSelectionType]
```

Example

```
// XMetaL Script Language JSCRIPT:
// same as pressing the Home key.
Selection.HomeKey();

// same as pressing Ctrl+Home.
Selection.HomeKey(1);

// same as pressing Shift+Home.
Selection.HomeKey(0, 1);

// same as pressing Ctrl+Shift+Home.
Selection.HomeKey(1, 1);
```

Indent

This method is unsupported.

InsertBreak

This method is unsupported.

InsertCALSTable

Inserts a CALS table with the specified top-level element, number of rows, and columns. Your DTD or schema must conform to the CALS table model for this method to work correctly. If your DTD or schema does not conform to the table model, your table may not be created. If you omit required CALS table elements, your table may not be created correctly. If you include elements that are not part of the CALS table model, your

table may not be created correctly. If you include attributes that are not a part of the table model, your attributes may not be supported. To insert a HTML table, use the `Selection.InsertTable` method .

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertCALSTable(intNumRows, intNumCols, ["strTableElement"],
[boolWantTHEAD], [boolWantTFoot]);
```

`strTableElement` is a qualified element name in the form `namespace prefix:local name`. If you do not want to specify a namespace, omit the namespace prefix and colon.

If `boolWantTHead` and/or `boolWantTFoot` have the value True, a table header and/or table footer are inserted. If the selection is not in a table cell (an `ENTRY` element) `strTableElement` is *required*.

By contrast, if the selection is in a table cell, `strTableElement` must be null or unspecified, and `boolWantTFoot` must be False.

CALS table can be inserted with a maximum of 150 rows and 150 columns. If `intNumRows` or `intNumCols` is set to more than 150, only 150 rows or columns are created. If more than 150 rows or columns are required, they can be inserted within the table once it has been created.

Usage in VBScript

```
Selection_object.InsertCALSTable intNumRows, intNumCols, ["strTableElement"],
[boolWantTHEAD], [boolWantTFoot]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Insert a 3 x 3 table that has a
// footer but no header. Selection.InsertCALSTable(3,3,"TABLE",false,true);

// Check the context before inserting
// XMetaL Script Language JSCRIPT:

if (Selection.ContainerName != "ENTRY") {
Selection.InsertCALSTable(3,3,"TABLE",true);
} else {
Selection.InsertCALSTable(3,3,"",true);
}
```

InsertCALSTableEx
Inserts a custom CALS table with the specified top-level element, number of rows, and columns. Your DTD or schema must conform to the CALS table model for this method to work correctly. If your DTD or schema does not conform to the table model, your table may not be created. If you omit required CALS table elements, your table may not be created correctly. If you include elements that are not part of the CALS table model, your table may not be created correctly. If you include attributes that are not a part of the table model, your attributes may not be supported. To insert a HTML table, use the `Selection.InsertTable` or `Selection.InsertTableEx` method.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertCALSTableEx(intNumRows, intNumCols, ["strTableElement"],
["strTgroupElement"], [boolWantTHead], [boolWantTFoot]);
```

`strTableElement` is a qualified element name in the form `namespace prefix:local name`. If you do not want to specify a namespace, omit the namespace prefix and colon.

`strTgroupElement` is a qualified element name in the form `namespace prefix:local name`. If you do not want to specify a namespace, omit the namespace prefix and colon.

If `boolWantTHead` and/or `boolWantTFoot` have the value True, a table header and/or table footer are inserted. If the selection is not in a table cell (an `ENTRY` element), `strTableElement` is *required*.

By contrast, if the selection is in a table cell, `strTableElement` must be null or unspecified, and `boolWantTFoot` must be False.

CALS table can be inserted with a maximum of 150 rows and 150 columns. If `intNumRows` or `intNumCols` is set to more than 150, only 150 rows or columns are created. If more than 150 rows or columns are required, they can be inserted within the table once it has been created.

Usage in VBScript

```
Selection_object.InsertCALSTableEx intNumRows, intNumCols, ["strTableElement"],
["strTgroupElement"], [boolWantTHEAD], [boolWantTFoot]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Insert a 3 x 3 table that has a
// footer but no header.
Selection.InsertCALSTableEx(3,3,"calstbl2","tgroup2",false,true);

// Check the context before inserting

// XMetaL Script Language JSCRIPT:
if (Selection.ContainerName != "ENTRY") {
  Selection.InsertCALSTableEx(3,3,"calstbl2","tgroup2", true);
} else {
  Selection.InsertCALSTableEx(3,3,"","",true);
}
```

InsertCaption

Inserts a table caption. This method will work if the selection is in a table that conforms to the HTML 4.0 table model.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertCaption();
```

Usage in VBScript

```
Selection_object.InsertCaption
```

Example

```
// XMetaL Script Language JSCRIPT:
if (Selection.InContextOfType("Table") {
Selection.InsertCaption();
}
```

InsertCDATASection

Inserts a CDATA section surrounding the selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertCDATASection();
```

Usage in VBScript

```
Selection_object.InsertCDATASection
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.InsertCDATASection();
```

InsertColumnsLeft

Inserts the specified number of table columns to the left of the column containing the selection. The default is one column.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertColumnsLeft([longNumber=1]);
```

Usage in VBScript

```
Selection_object.InsertColumnsLeft [longNumber=1]
```

Example

```
// XMetaL Script Language JSCRIPT:
if (Selection.InContextOfType("Table")) {
Selection.InsertColumnsLeft(); // insert 1 column
}
```

InsertColumnsRight

Inserts the specified number of table columns to the right of the column containing the selection. The default is one column.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertColumnsRight([longNumber]);
```

Usage in VBScript

```
Selection_object.InsertColumnsRight [longNumber]
```

Example

```
// XMetaL Script Language JSCRIPT:
if (Selection.InContextOfType("Table")) {
Selection.InsertColumnsRight(2); // insert 2 columns
}
```

InsertComment

Inserts a comment surrounding the selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertComment();
```


Usage in VBScript

```
Selection_object.InsertComment
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.InsertComment();
```

InsertElement
Inserts the specified element at the selection. Does not insert default content (if specified by the customization). Leaves the selection inside the new element. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertElement("strElementName");
```

Usage in VBScript

```
Selection_object.InsertElement "strElementName"
```

Example

```
// XMetaL Script Language JSCRIPT:
// inserts a <B> element at the selection.

Selection.InsertElement("B");
```

InsertElementNS
Inserts the specified element in the specified namespace at the selection. Does not insert default content (if specified by the customization). Leaves the selection inside the new element. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertElementNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.InsertElementNS "strNamespaceNameURI", "strLocalName"
```

Example

```
// XMetaL Script Language JSCRIPT:
// inserts a <B> element (local to http://www.mydomainname.com)
// at the selection.

Selection.InsertElementNS("http://www.mydomainname.com", "B");
```

InsertElementWithRequired
Inserts the specified element, and its first required child element (if there is one). This insertion process is recursive, so that if the required child element itself has a first required child element, it is inserted, and so forth. Note that this does not mean that *all* required child elements of the specified element are inserted, just the recursive sequence of first required child elements. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertElementWithRequired ("strElementName");
```

Usage in VBScript

```
Selection_object.InsertElementWithRequired "strElementName"
```

Example

```
// XMetaL Script Language JSCRIPT:
// inserts an <ADDRESS> element at
// the document's selection.

Selection.InsertElementWithRequired("ADDRESS");
```

InsertElementWithRequiredNS
Inserts the specified element (that is local to the specified namespace), and its first required child element (if there is one). This insertion process is recursive, so that if the required child element itself has a first required child element, it is inserted, and so forth. Note that this does not mean that *all* required child elements of the specified element are inserted, just the recursive sequence of first required child elements. Inserts the specified element and its first required child element (if there is one). This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertElementWithRequiredNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.InsertElementWithRequiredNS "strNamespaceNameURI", "strLocalName"
```

Example

```
// XMetaL Script Language JSCRIPT:
// inserts an <ADDRESS> element (local to http://www.mydomainname.com) at
// the document's selection.

Selection.InsertElementWithRequired("http://www.mydomainname.com",
"ADDRESS");
```

InsertEntity
Inserts a reference to the specified entity.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertEntity("strEntityName");
```

Usage in VBScript

```
Selection_object.InsertEntity "strEntityName"
```

Example

```
// XMetaL Script Language JSCRIPT:
// inserts an   entity reference.

Selection.InsertEntity("nbsp");
```

InsertForm
This method is unsupported.

InsertHorizontalRule
This method is unsupported.

InsertImage
Inserts the appropriate image element at the selection, with the specified attribute values.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertImage("strSrc", ["strAlt"], ["strAlign"], ["strWidth"],
["strHeight"], ["strBorder"]);
```

The element inserted is the first valid image element, as declared with the customization file editor in XMetaL Developer. The following parameters are not currently supported and reserved for future use: strAlign, strBorder. To avoid problems trying to use these unsupported parameters, it is recommended that you use this method with the ElementAttribute property in the following way:

```
Selection.InsertElement("imageElem")
Selection.ElementAttribute("strSrc", "imageElem") = value
Selection.ElementAttribute("strAlt", "imageElem") = value
Selection.ElementAttribute("strWidth", "imageElem") = value
Selection.ElementAttribute("strHeight", "imageElem") = value
```

Usage in VBScript

```
Selection_object.InsertImage "strSrc", ["strAlt"], ["strAlign"], ["strWidth"],
["strHeight"], ["strBorder"]
```

Example

```
// XMetaL Script Language JSCRIPT:
//Insert a picture "yourgif.gif" with 20x20 dimension,
//aligned to bottom with a border
Selection.InsertImage("c:/yourgif.gif", "your picture",
"", "20", "20", "");
// Insert a picture "santiago.jpg";
// dimensions are inserted automatically
Selection.InsertImage("santiago.jpg");
```

InsertLink

This method is unsupported.

InsertNBSP

Inserts a non-breaking space entity (` `) into a document where the entity is declared.

Applies To

XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertNBSP();
```

Usage in VBScript

```
Selection_object.InsertNBSP
```


Example

```
// XMetaL Script Language JSCRIPT:
Selection.InsertNBSP();
```

InsertProcessingInstruction

Inserts an empty processing instruction to the right of the selection. The selection becomes an insertion point inside the processing instruction.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertProcessingInstruction();
```

Usage in VBScript

```
Selection_object.InsertProcessingInstruction
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.InsertProcessingInstruction();
```

InsertReplaceableText

Inserts an `xm-replace_text` processing instruction, which represents replaceable text in the Normal and Tags On views. The target component of the processing instruction is the string ‘xm-replace_text’, and the data component is `strData`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection.InsertReplaceableText("strData");
```

Usage in VBScript

```
Selection.InsertReplaceableText "strData"
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.InsertReplaceableText("Section Title");
```

InsertRowsAbove
Insert the specified number of table rows above the row containing the selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertRowsAbove([longNumber]);
```

Usage in VBScript

```
Selection_object.InsertRowsAbove [longNumber]
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.InsertRowsAbove(5);
```

InsertRowsBelow
Insert the specified number of table rows below the row containing the selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertRowsBelow([longNumber]);
```

Usage in VBScript

```
Selection_object.InsertRowsBelow [longNumber]
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.InsertRowsBelow(1);
```


InsertTable

Inserts an HTML table at the selection. Your DTD or schema must conform to the HTML table model for this method to work correctly. If your DTD or schema does not conform to the table model, your table may not be created. If you omit required HTML table elements, your table may not be created correctly. If you include elements that are not part of the HTML table model, your table may not be created correctly. If you include attributes that are not a part of the table model, your attributes may not be supported. To insert a CALS table, use the `Selection.InsertCALSTable` method.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertTable(intRows, intCols, ["strBGColor"], ["strTableWidth"],
["strTableWidthUnit"], ["strTableBorder"], ["strCellPadding"], ["strCellSpacing"]);
```

The parameters are:

- `intRows` = number of rows
- `intCols` = number of columns
- `strBGColor` = background color
- `strTableWidth` = table width
- `strTableWidthUnit` = unit of measure for table width
- `strTableBorder` = border width
- `strCellPadding` = cell padding, in pixels
- `strCellSpacing` = cell spacing, in pixels

Usage in VBScript

```
Selection_object.InsertTable intRows, intCols, ["strBGColor"], ["strTableWidth"],
["strTableWidthUnit"], ["strTableBorder"], ["strCellPadding"], ["strCellSpacing"]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Insert a 2x2 table with a white background,
// 100% width,a border size of 1
// cell padding of 10 pixels, and
// cell spacing of 5 pixels
Selection.InsertTable(2, 2, "#ffffff", "100", "%", "1", "10", "5");
```

InsertTableEx

Inserts an HTML table at the selection. Your DTD or schema must conform to the HTML table model for this method to work correctly. If your DTD or schema does not conform to the table model, your table may not be created. If you omit required HTML table elements, your table may not be created correctly. If you include elements that are not part of the HTML table model, your table may not be created correctly. If you include attributes that are not a part of the table model, your attributes may not be supported. To insert a CALS table, use the `Selection.InsertCALSTable` method.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertTableEx(intNumRows, intNumCols, strTableElement, strBGColor,
strTableWidth, strTableWidthUnit, strTableBorder, strCellPadding, strCellSpacing);
```

The parameters are:

- `intRows` = number of rows
- `intCols` = number of columns
- `strBGColor` = background color
- `strTableElement` = table element type name
- `strTableWidth` = table width
- `strTableWidthUnit` = unit of measure for table width
- `strTableBorder` = border width
- `strCellPadding` = cell padding, in pixels
- `strCellSpacing` = cell spacing, in pixels

Usage in VBScript

```
Selection_object.InsertTable intRows, intCols, ["strTableElement"], ["strBGColor"],
["strTableWidth"], ["strTableWidthUnit"], ["strTableBorder"], ["strCellPadding"],
["strCellSpacing"]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Insert a 2x2 table with a white background,
// 100% width,a border size of 1
// cell padding of 10 pixels, and
// cell spacing of 5 pixels
Selection.InsertTable(2, 2, "TABLE2", "#ffffff", "100", "%", "1", "10",
"5");
```

InsertWithTemplate
Inserts the specified element. If default content has been specified in the customization (CTM file), this content is inserted with the element; if a script was specified (CTM file), it is executed; if neither are specified then just the element is inserted (as with the insertElement method). This method can be used in 'default content' scripts to insert other elements and their default content. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertWithTemplate("strElementName");
```

Usage in VBScript

```
Selection_object.InsertWithTemplate "strElementName"
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.InsertWithTemplate("P");
```

InsertWithTemplateNS

Inserts the specified element. If default content has been specified in the customization (CTM file), this content is inserted with the element; if a script was specified (CTM file), it is executed; if neither are specified then just the element is inserted (as with the insertElement method). This method can be used in 'default content' scripts to insert other elements and their default content.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.InsertWithTemplateNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.InsertWithTemplateNS "strNamespaceNameURI", "strLocalName"
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.InsertWithTemplateNS("http://www.mydomainname.com", "P");
```

JoinElementToPreceding

Joins the element containing the selection with the element immediately preceding it, if valid.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.JoinElementToPreceding();
```

Usage in VBScript

```
Selection_object.JoinElementToPreceding
```

Example

```
// XMetaL Script Language JSCRIPT:
//Take the following XML file
//<p>data1 </p>
//<p>data2 </p>
// The following script merges the
// <p> tags and contents
// into one if the document selection is
// in the second <p> tag (data2). Selection.JoinElementToPreceding();
// The result:
//<p> data1 data2 </p>
```

Layout

Applies text layout (as specified in the Text Layout property in the customization) to the current selection. This method can be used only if the document is displayed in Plain Text view. It applies the text layout and returns True if the document is in Plain Text view, and does nothing and returns False otherwise.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Selection_object.Layout();
```

Usage in VBScript

```
Selection_object.Layout
```

Example

```
// XMetaL Script Language JSCRIPT:
// Format the selection

if (ActiveDocument.ViewType == 2) {
Selection.Layout();
}
```

ListIndent

Demotes a list item to a sublist. The sublist can specify an element name, otherwise it is the same type as the current list. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.ListIndent([strElementName]);
```

Usage in VBScript

```
Selection_object.ListIndent [strElementName]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Demote current list item to a
// "SimpleList" list
Selection.ListIndent("SimpleList");
```

ListIndentNS

Demotes a list item to a sublist. The sublist can specify an element name (that is local to the specified namespace), otherwise it is the same type as the current list. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.ListIndentNS([strNamespaceNameURI], [strLocalName]);
```

Usage in VBScript

```
Selection_object.ListIndentNS [strNamespaceNameURI], [strLocalName]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Demote current list item to a
// "SimpleList" list
Selection.ListIndentNS("http://www.mydomainname.com","SimpleList");
```

ListOutdent

Promotes a list item out of its list. The list item can be promoted to a specified element, otherwise to the default paragraph element. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.ListOutdent([strElementName]);
```

Usage in VBScript

```
Selection_object.ListOutdent [strElementName]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Promote current list item to a
// "Note" element
Selection.ListOutdent("Note");
```

ListOutdentNS

Promotes a list item out of its list. The list item can be promoted to a specified element (that is local to the namespace specified), otherwise to the default paragraph element. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.ListOutdentNS([strNamespaceNameURI], [strLocalName]);
```

Usage in VBScript

```
Selection_object.ListOutdentNS [strNamespaceNameURI], [strLocalName]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Promote current list item to a
// "Note" element
Selection.ListOutdentNS("http://www.mydomainname.com", "Note");
```

MergeCellDown

Merges the table cell containing the selection with the cell below it.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MergeCellDown();
```

Usage in VBScript

```
Selection_object.MergeCellDown
```

Example

```
//XMetaL Script Language JSCRIPT:

//If currently in a table cell, merge cell down
if (Selection.InContextOfType("TableCell")){
Selection.MergeCellDown();
}
```

MergeCellLeft
Merges the table cell containing the selection with the cell to its left.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MergeCellLeft();
```

Usage in VBScript

```
Selection_object.MergeCellLeft
```

Example

```
//XMetaL Script Language JSCRIPT:

//If currently in a table cell, merge cell left
if (Selection.InContextOfType("TableCell")){
Selection.MergeCellLeft();
}
```

MergeCellRight
Merges the table cell containing the selection with the cell to its right.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MergeCellRight();
```

Usage in VBScript

```
Selection_object.MergeCellRight
```

Example

```
//XMetaL Script Language JSCRIPT:

//If currently in a table cell, merge cell right
if (Selection.InContextOfType("TableCell")){
Selection.MergeCellRight();
}
```

MergeCellUp
Merges the table cell containing the selection with the cell above it.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MergeCellUp();
```

Usage in VBScript

```
Selection_object.MergeCellUp
```

Example

```
//XMetaL Script Language JSCRIPT:

//If currently in a table cell, merge it up
if (Selection.InContextOfType("TableCell")){
Selection.MergeCellUp();
}
```

MergeTable
Merges two adjacent tables into one table.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Description

Returns true if two adjacent tables can be merged into one table.

If withTableBelow is true, it indicates that the table with the selection will be merged with the table after it.

If withTableBelow is false, it indicates that the table with the selection will be merged with the table before it.

Usage in JScript

```
Selection.MergeTable(withTableBelow);
```

Usage in VBScript

```
Selection.MergeTable(withTableBelow)
```

Example

```
// XMetaL Script Language JSCRIPT:

var withTableBelow = true;
Selection.MergeTable(withTableBelow);
```

MoveColumnLeft

Moves the column containing the selection one column to the left.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MoveColumnLeft();
```

Usage in VBScript

```
Selection_object.MoveColumnLeft
```

Example

```
// XMetaL Script Language JSCRIPT:
if (Selection.InContextOfType("Table")){
Selection.MoveColumnLeft();
}
```

MoveColumnRight

Moves the column containing the selection one column to the right.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MoveColumnRight();
```

Usage in VBScript

```
Selection_object.MoveColumnRight
```

Example

```
// XMetaL Script Language JSCRIPT:
if (Selection.InContextOfType("Table")){
Selection.MoveColumnRight();
}
```

MoveDown
Moves the selection down.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MoveDown([intSelectionType]);
```

`intSelectionType` specifies the type of resulting selection:

- sqMove (0)= Collapse the selection to an insertion point, and move it to the next line, directly below the end point of the original selection (default)
- sqExtend (1)= Extend the selection to the next line, ending directly below the end point of the original selection

Usage in VBScript

```
Selection_object.MoveDown [intSelectionType]
```

Example

```
// XMetaL Script Language JSCRIPT:
// moves selection down
Selection.MoveDown();
// moves selection down and highlights text.
Selection.MoveDown(1);
```

MoveLeft
Moves the selection to the left.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MoveLeft([intSelectionType]);
```

`intSelectionType` specifies the type of resulting selection:

- sqMove (0)= Collapse the selection to an insertion point, and move it one character to the left of the start point of the original selection (default)
- sqExtend (1)= Move the endpoint of the selection one character to the left

Usage in VBScript

```
Selection_object.MoveLeft [intSelectionType]
```

Example

```
// XMetaL Script Language JSCRIPT:
// moves selection to left
Selection.MoveLeft();
// moves selection to left and highlights text.
Selection.MoveLeft(1);
```

MoveRight

Moves the selection to the right.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MoveRight([intSelectionType]);
```

`intSelectionType` specifies the type of resulting selection:

- sqMove (0)= Collapse the selection to an insertion point, and move it one character to the right of the end point of the original selection (default)
- sqExtend (1)= Extend the selection one character to the right

Usage in VBScript

```
Selection_object.MoveRight [intSelectionType]
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.MoveRight();// moves cursor to right
// moves cursor to right and highlights text.
Selection.MoveRight(1);
```

MoveRowDown
Moves the table row containing the selection one row down.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MoveRowDown();
```

Usage in VBScript

```
Selection_object.MoveRowDown
```

Example

```
// XMetaL Script Language JSCRIPT:
if (Selection.InContextOfType("Table")){
Selection.MoveRowDown();
}
```

MoveRowUp
Moves the table row containing the selection one row up.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MoveRowUp();
```

Usage in VBScript

```
Selection_object.MoveRowUp
```


Example

```
// XMetaL Script Language JSCRIPT:
if (Selection.InContextOfType("Table")){
Selection.MoveRowUp();
}
```

MoveToDocumentEnd
Makes the selection an insertion point at the end of the document.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MoveToDocumentEnd();
```

Usage in VBScript

```
Selection_object.MoveToDocumentEnd
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.MoveToDocumentEnd();
```

MoveToDocumentStart
Makes the selection an insertion point at the start of the document.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MoveToDocumentStart();
```

Usage in VBScript

```
Selection_object.MoveToDocumentStart
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.MoveToDocumentStart();
```

MoveToElement
Moves the selection to the specified element. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
Selection_object.MoveToElement(["strElementName"], [boolForward]);
```

By default, the movement is forward; if `boolForward` is False, the movement is backward. If `strElementName` is omitted or empty, the selection moves to the next element of any type. `MoveToElement` does not wrap around the top or bottom of the document; that is, it stops if it reaches the top or bottom. `MoveToElement` can find invisible elements, such as table elements that are not displayed by the table editor, and elements that have been marked as 'Hide in Normal View'.

Usage in VBScript

```
Selection_object.MoveToElement(["strElementName"], [boolForward])
```

Example

```
// XMetaL Script Language JSCRIPT:
var rng=ActiveDocument.Range;
rng.MoveToElement("P",false);
// make the range visible
rng.Select();
```

MoveToElementNS
Moves the selection to the specified element that is local to the specified namespace. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Description

By default, the movement is forward; if `boolForward` is `false`, the movement is backward. If `strLocalName` is omitted or empty, the selection moves to the next element of any type. `MoveToElementNS` does not wrap around the top or bottom of the document; that is, it stops if it reaches the top or bottom. `MoveToElementNS` can find invisible elements, such as table elements that are not displayed by the table editor, and elements that have been marked as 'Hide in Normal View'.

Usage in JScript

```
Selection_object.MoveToElementNS([strNamespaceNameURI], [strLocalName], [boolForward]);
```

Usage in VBScript

```
Selection_object.MoveToElementNS([strNamespaceNameURI], [strLocalName], [boolForward])
```

Example

```
// XMetaL Script Language JSCRIPT:
var rng=ActiveDocument.Range;
rng.MoveToElementNS("http://www.mydomainname.com", "P", false);
// make the range visible
rng.Select();
```

MoveUp
Moves the selection up.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.MoveUp([intSelectionType]);
```

`intSelectionType` specifies the type of resulting selection:

- sqMove (0)= Collapse the selection to an insertion point, and move it to the previous line, directly above the start point of the original selection (default)
- sqExtend (1)= Extend the selection to the previous line, ending directly above the start point of the original selection

Usage in VBScript

```
Selection_object.MoveUp [intSelectionType]
```

Example

```
// XMetaL Script Language JSCRIPT:
// moves selection up
Selection.MoveUp();
// moves selection up and highlights text. Selection.MoveUp(1);
```

Outdent
This method is unsupported.

PageDown
Scrolls the document one page down.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.PageDown();
```

Usage in VBScript

```
Selection_object.PageDown
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.PageDown();
```

PageUp

Scrolls the document one page up.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.PageUp();
```

Usage in VBScript

```
Selection_object.PageUp
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.PageUp();
```

Paste

Pastes the contents of the clipboard over the selection using a ‘smart paste’ operation that attempts to split elements or make other modifications so that the paste occurs at a valid location.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.Paste();
```

Usage in VBScript

```
Selection_object.Paste
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.Paste();
```

PasteString
Pastes the specified text over the selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.PasteString("strText");
```

Usage in VBScript

```
Selection_object.PasteString "strText"
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.PasteString("hello");
```

PasteStringAsText
This method is similar to `Selection.PasteString`, except that the pasted string is not interpreted as markup. Text containing characters such as ‘<’, ‘<’, or ‘&’ are pasted as though the user had typed it.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.PasteStringAsText("strText");
```

Usage in VBScript

```
Selection_object.PasteStringAsText "strText"
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.PasteStringAsText('Characters such as "<" & ">" are pasted as
though they were typed.');
```

PasteStringWithInterpret

Replaces the selection with the content specified. Tab-formatted tables are pasted as the appropriate table format (HTML or CALS).

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.PasteStringWithInterpret("String");
```

Usage in VBScript

```
Selection_object.PasteStringWithInterpret "String"
```

Example

```
// XMetaL Script Language JSCRIPT:
var tableString;
tableString=Application.FileToString("c:\\table1.txt");

if (Selection.CanPaste(tableString,true)) {
Selection.PasteStringWithInterpret(tableString);
} else {
Selection.InsertElement("LiteralLayout");
Selection.PasteString(tableString);
}
```

RejectAllChanges

Rejects all changes made using change tracking within a selected block. By first setting the Selection to the text marked as changed, you can then use the RejectAllChanges method to reject all changes in the indicated selection. If the Selection is only an insertion point (as is allowed as a valid Selection) then there is no effect. You can also use the AcceptAllChanges method. It is recommended that you use the Range interface with this method instead of the Selection interface: `Document.Range.RejectAllChanges`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.RejectAllChanges();
```

Usage in VBScript

```
Selection_object.RejectAllChanges()
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a selection and reject all changes made within it

var sel=Application.Selection; //Selected text must contain change-tracking
 changes
sel.RejectAllChanges();

// Now observe the selection to see all changes were accepted
```

RemoveContainerTags
Removes the tags of the selection‘s container. (In effect, this removes the container.)

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection.RemoveContainerTags();
```

Usage in VBScript

```
Selection.RemoveContainerTags
```

Example

```
// XMetaL Script Language JSCRIPT:

if (Selection.ContainerName == "EM") {
Selection.RemoveContainerTags();
}
```

Select
Sets the user’s selection to the selection represented by the specified Selection or Range object. This method is designed to be used primarily with Range objects: if the specified object is already the selection, nothing happens; if a Range object is specified, it becomes the selection. (Until they are selected, Range objects are ‘invisible’.)

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.Select();
```

Usage in VBScript

```
Selection_object.Select
```

Example

```
// XMetaL Script Language JSCRIPT:
var rng=ActiveDocument.Range;
rng.SelectContainerContents();
rng.Select(); // move the user's selection
```

SelectAfterContainer
Collapses the selection to an insertion point, and moves it directly after the element that contained the original selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SelectAfterContainer();
```

Usage in VBScript

```
Selection_object.SelectAfterContainer
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.SelectAfterContainer();
```

SelectAfterNode
Collapses the selection to an insertion point, and moves it directly after the container corresponding to the specified DOMNode.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SelectAfterNode(Node);
```

Usage in VBScript

```
Selection_object.SelectAfterNode Node
```

Example

```
// XMetaL Script Language JSCRIPT:
//Moves the cursor to after the container's node.

Selection.SelectAfterNode(Selection.ContainerNode);
```

SelectAll

Selects the entire contents of the document that contains the selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SelectAll();
```

Usage in VBScript

```
Selection_object.SelectAll
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.SelectAll();
```

SelectBeforeContainer

Collapses the selection to an insertion point, and moves it directly before the original selection’s container.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SelectBeforeContainer();
```

Usage in VBScript

```
Selection_object.SelectBeforeContainer
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.SelectBeforeContainer();
```

SelectBeforeNode

Collapses the selection to an insertion point, and moves it directly before the container corresponding to the specified DOMNode.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SelectBeforeNode(Node);
```

Usage in VBScript

```
Selection_object.SelectBeforeNode Node
```

Example

```
// XMetaL Script Language JSCRIPT:
// Moves the selection to before the selection's
// container node.

Selection.SelectBeforeNode(Selection.ContainerNode);
```

SelectContainerContents

Selects the entire contents of the selection’s container.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SelectContainerContents();
```


Usage in VBScript

```
Selection_object.SelectContainerContents
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.SelectContainerContents();
```

SelectElement
Selects the entire container (element, comment, section, or processing instruction, including tags) containing the selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SelectElement();
```

Usage in VBScript

```
Selection_object.SelectElement
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.SelectElement();
```

SelectNodeContents
Selects all the contents of the node specified by DOMNode.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SelectNodeContents(DOMNode);
```

Usage in VBScript

```
Selection_object.SelectNodeContents DOMNode
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.SelectNodeContents(Selection.ContainerNode);
```

SplitCellColumn
Splits the table cell containing the selection into two columns. The original contents remain in the cell on the left.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SplitCellColumn();
```

Usage in VBScript

```
Selection_object.SplitCellColumn
```

Example

```
//XMetaL Script Language JSCRIPT:

//If currently in a table cell, split it into two columns
if (Selection.InContextOfType("TableCell")){
Selection.SplitCellColumn();
}
```

SplitCellRow
Splits the table cell that contains the selection into two rows. The original contents remain in the top row.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SplitCellRow();
```

Usage in VBScript

```
Selection_object.SplitCellRow
```


Example

```
//XMetaL Script Language JSCRIPT:

//If currently in a table cell, split it into two rows
if (Selection.InContextOfType("TableCell")){
Selection.SplitCellRow();
}
```

SplitContainer
Splits the selection's container into two containers of the same type as the original container, at the start point of the selection. The selection remains in the second part. This method performs the same action as the **Split Element** menu item in XMetaL Author.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SplitContainer();
```

Usage in VBScript

```
Selection_object.SplitContainer
```

Example

```
// XMetaL Script Language JSCRIPT:
// Example from Journalist.mcr
// If cannot insert figure, split the container and see if we can then
var node = rng.ContainerNode;

if (node) {
var elName = node.nodeName;

if (elName == "Para" || elName == "LiteralLayout" || elName ==
"ProgramListing") {
Selection.SplitContainer();
rng = ActiveDocument.Range;
var rngSave = rng.Duplicate;
rng.SelectBeforeContainer();

if (rng.CanInsert("Figure")) {
rng.Select();

if (doFigureInsert()) {
rng = null;
} else {
rngSave.Select();
Selection.JoinElementToPreceding();
rng = null;
}
} else {
// Join selection back together
Selection.JoinElementToPreceding();
}
```

```
rngSave = null;
}
}
```

SplitTable
Method that splits the table into two in the given direction under the current selection.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Nothing

Description

Split the table under the selection into two tables. If belowThisRow is true, it indicates that the table row contains the current selection will be in the 1st table after splitting. If belowThisRow is false, it indicates that the table row contains the current selection will be in the 2nd table after splitting.

Usage in JScript

```
Selection.SplitTable(belowThisRow);
```

Usage in VBScript

```
Selection.SplitTable(belowThisRow)
```

Example

```
// XMetaL Script Language JSCRIPT:

var belowThisRow = true;
Selection.SplitTable(belowThisRow);
```

SplitTableGroup
Method for splitting a table group under the current selection into two table groups.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Nothing

Description

The table group under the selection can be split into two table groups.

If belowThisRow is false, it indicates that the table row contains the current selection will be in the 1st table group after splitting.

If belowThisRow is false, it indicates that the table row contains the current selection will be in the 2nd table group after splitting.

For HTML table, this API is as same as *SplitTable* on page 425.

Usage in JScript

```
Selection.SplitTableGroup(belowThisRow);
```

Usage in VBScript

```
Selection.SplitTableGroup(belowThisRow)
```

Example

```
// XMetaL Script Language JSCRIPT:

var belowThisRow = true;
Selection.SplitTableGroup(belowThisRow);
```

SplitToElementType

This method splits the current container and sets the second element to the specified name. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Description

For example, consider a document based on the Journalist DTD with the following markup:

```
<Para>one <Emphasis>twothree</Emphasis></Para>
```

with the selection between the words ‘two’ and ‘three’.

If you were to call:

```
Selection.SplitToElementType("Strong")
```

the markup would changed to:

```
<Para>one
 <Emphasis>two</Emphasis><Strong>three</Strong></Para>
```

Usage in JScript

```
Selection_object.SplitToElementType("strElementName");
```

Usage in VBScript

```
Selection_object.SplitToElementType("strElementName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Example modified from Journalist.mcr
// If cannot insert figure, split the container
// and see if we can then

var rng = ActiveDocument.Range;
var node = rng.ContainerNode;

if (node) {
var elName = node.nodeName;

if (elName == "Para" || elName == "LiteralLayout" || elName ==
"ProgramListing") {
Selection.SplitToElementType("Para");
rng = ActiveDocument.Range;
var rngSave = rng.Duplicate;
rng.SelectBeforeContainer();

if (rng.CanInsert("Figure")) {
rng.Select();

if (doFigureInsert()) {
rng = null;
rngSave = null;
} else {
rngSave.Select();
Selection.JoinElementToPreceding();
rng = null;
rngSave = null;
}
} else {
// Join selection back together
Selection.JoinElementToPreceding();
}
rngSave = null;
}
}
```

SplitToElementTypeNS

Splits the current container and sets the second element to the specified name. The element specified is local to the namespace specified. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Description

For example, consider a document based on the Journalist DTD with the following markup:

```
<Para>one <Emphasis>twothree</Emphasis></Para>
```

with the selection between the words ‘two’ and ‘three’.

If you were to call:

```
Selection.SplitToElementTypeNS("http://www.mydomainname.com/Journalist
", "Strong")
```

the markup would changed to:

```
<Para>one
 <Emphasis>two</Emphasis><Strong>three</Strong></Para>
```

Usage in JScript

```
Selection_object.SplitToElementTypeNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.SplitToElementTypeNS "strNamespaceNameURI", "strLocalName"
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.SplitToElementTypeNS("http://www.mydomainname.com/Journalist",
 "Strong");
```

Surround

Surrounds the selection with the specified element. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.Surround("strElementName");
```

Usage in VBScript

```
Selection_object.Surround "strElementName"
```

Example

```
// XMetaL Script Language JSCRIPT:
//surrounds the selected text with <P> and
// </P>tags
Selection.Surround("Para");
ActiveDocument.Host.Alert("The selected text is now a paragraph (Para
tag).");
Selection.ToggleInline("Para");
ActiveDocument.Host.Alert("The selected text is no longer a paragraph
(Para tag).")
Selection.Surround("Citation");
ActiveDocument.Host.Alert("The selected text is now a Citation.")
```

SurroundNS

Surrounds the selection with the specified element, which is local to the specified namespace. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.SurroundNS("strNamespaceNameURI", "strLocalName")
```

Usage in VBScript

```
Selection_object.SurroundNS "strNamespaceNameURI", "strLocalName"
```

Example

```
// XMetaL Script Language JSCRIPT:
//surrounds the selected text with <P> and
// </P>tags
Selection.SurroundNS("http://www.mydomainname.com/Journalist", "Para");
ActiveDocument.Host.Alert("The selected text is now a paragraph (Para
tag).");
Selection.ToggleInlineNS("http://www.mydomainname.com/Journalist" "Para");
ActiveDocument.Host.Alert("The selected text is no longer a paragraph
(Para tag).")
Selection.SurroundNS("http://www.mydomainname.com/Journalist", "Citation");
ActiveDocument.Host.Alert("The selected text is now a Citation.")
```

ToggleInline

Toggles the selected text with the element name specified. If the selection is not inside this type of tag, this tag is applied. If the selection is already inside this tag, the tag is removed. A macro containing this method can be written and called from a button added to the toolbar. It acts like a Bold or Italic button in popular word processors. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.ToggleInline(TagName);
```

Usage in VBScript

```
Selection_object.ToggleInline(TagName)
```


Example

```
// XMetaL Script Language JSCRIPT:
//surrounds the selected text with <P> and
//</P>tags
Selection.ToggleInline("Para");
ActiveDocument.Host.Alert("The selected text is now a paragraph (Para
tag).");
Selection.ToggleInline("Para");
ActiveDocument.Host.Alert("The selected text is no longer a paragraph
(Para tag).")
```

ToggleInlineNS

Toggles the selected text with the element name specified that is local to the namespace specified. If the selection is not inside this type of tag, this tag is applied. If the selection is already inside this tag, the tag is removed. A macro containing this method can be written and called from a button added to the toolbar. It acts like a Bold or Italic button in popular word processors. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.ToggleInlineNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
Selection_object.ToggleInlineNS "strNamespaceNameURI", "strLocalName"
```

Example

```
// XMetaL Script Language JSCRIPT:
//surrounds the selected text with <P> and
// </P>tags
Selection.SurroundNS("http://www.mydomainname.com/Journalist", "Para");
ActiveDocument.Host.Alert("The selected text is now a paragraph (Para
tag).");
Selection.ToggleInlineNS("http://www.mydomainname.com/Journalist", "Para");
ActiveDocument.Host.Alert("The selected text is no longer a paragraph
(Para tag).")
Selection.SurroundNS("http://www.mydomainname.com/Journalist", "Citation");
ActiveDocument.Host.Alert("The selected text is now a Citation.")
```

ToggleTableCellType

Toggles the table cell containing the selection between the `TD` element and the `TH` element. This method applies only to documents with tables that conform to the HTML 4.0 table model.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.ToggleTableCellType();
```

Usage in VBScript

```
Selection_object.ToggleTableCellType
```

Example

```
// XMetaL Script Language JSCRIPT:

Selection_object.ToggleTableCellType();
```

TypeTab

Performs the same action as the Tab key.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.TypeTab([boolShift]);
```

If `boolShift` is True, performs the same action as Shift+Tab.

Usage in VBScript

```
Selection_object.TypeTab [boolShift]
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.TypeTab();
```

TypeText

Pastes the specified text over the selection.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.TypeText("strText");
```

Usage in VBScript

```
Selection_object.TypeText "strText"
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.TypeText("Insert text here.");
```

TypingSplit
This method has the same effect as pressing the Enter key. This could split selection's container element, insert the container's 'followed by' element, or create a line break, depending on the container. If the container is split, the selected text is moved to the new (second) container.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.TypingSplit();
```

Usage in VBScript

```
Selection_object.TypingSplit
```

Example

```
// XMetaL Script Language JSCRIPT:
// Take the following file and the
// selection covering "ata1"
// <p>data1 </p>
// The following script splits the
// <p> tag and moves "ata1"
// into the new <p> tag
Selection.TypingSplit();
// The result:
//<p>d</p>
//<p>ata1 </p>
```

Validate
Validates the selection according to the document's DTD, Schema, or rules file, just as if the **Validate Selection** command were chosen in XMetaL Author. A message is displayed to the user.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
Selection_object.Validate();
```

Usage in VBScript

```
Selection_object.Validate
```

Example

```
// XMetaL Script Language JSCRIPT:
Selection.Validate();
```

TogglingElement

The TogglingElement interface represents a collection of toggling element properties. These properties are used when an element is treated as a toggling element in a document customization. This interface can be acquired only from the TogglingElements interface.

Properties

MacroName

The name of the macro associated with this toggling element. Macros associated with toggling elements should call the `Selection.ToggleInline` or `Selection.ToggleInlineNS` methods.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strMacroName = TogglingElement_obj.MacroName;
```

Usage in VBScript

```
dim strMacroName = TogglingElement_obj.MacroName
```


Example

```
'XMetaL Script Language VBScript:
'displays all of the toggling elements in a document's customization

Set ctm = ActiveDocument.Customizations
Set objs = ctm.TogglingElements

MsgBox("Bold element is " + objs.Bold)
MsgBox("Italic element is " + objs.Italic)
MsgBox("Underline element is " + objs.Underline)

num2 = objs.Count
MsgBox("Number of toggling elements with macros is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.Item(cnt1)
MsgBox("Name = " + obj1.name + " MacroName = " + obj1.MacroName)
Next
```

name
The name of the toggling element.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strTogglingElementName = TogglingElement_obj.name;
```

Usage in VBScript

```
dim strTogglingElementName = TogglingElement_obj.name
```

Example

```
'XMetaL Script Language VBScript:
'displays all of the toggling elements in a document's customization

Set ctm = ActiveDocument.Customizations
Set objs = ctm.TogglingElements

MsgBox("Bold element is " + objs.Bold)
MsgBox("Italic element is " + objs.Italic)
MsgBox("Underline element is " + objs.Underline)

num2 = objs.Count
MsgBox("Number of toggling elements with macros is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.Item(cnt1)
MsgBox("Name = " + obj1.name + " MacroName = " + obj1.MacroName)
Next
```

TogglingElements

The TogglingElements interface stores a collection of elements treated as toggling elements. Use this interface to get the properties of each individual element treated as a toggling element. This interface can be acquired only from the Customizations interface.

Properties

Bold

The name of the Bold toggling element. The selected text is surrounded with this element when the Bold button is clicked on the Formatting toolbar in XMetaL Author. You can set this property using the Bold property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strBoldTogglingElementName = ToggligElements_obj.Bold;
```

Usage in VBScript

```
dim strBoldTogglingElementName = ToggligElements_obj.Bold
```

Example

```
'XMetaL Script Language VBScript:
'displays all of the toggling elements in a document's customization

Set ctm = ActiveDocument.Customizations
Set objs = ctm.TogglingElements

MsgBox("Bold element is " + objs.Bold)
MsgBox("Italic element is " + objs.Italic)
MsgBox("Underline element is " + objs.Underline)

num2 = objs.Count
MsgBox("Number of toggling elements with macros is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.Item(cnt1)
MsgBox("Name = " + obj1.name + " MacroName = " + obj1.MacroName)
Next
```

count

Returns the number of elements treated as toggling elements in the document's customization, excluding the Bold, Italic, and Underline toggling elements.

Applies to

XMetaL XMAXand XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
var intNumTogglingElements = TogglingElements_obj.count;
```

Usage in VBScript

```
dim intNumTogglingElements = TogglingElements_obj.count
```

Example

```
'XMetaL Script Language VBScript:
'displays all of the toggling elements in a document's customization

Set ctm = ActiveDocument.Customizations
Set objs = ctm.TogglingElements

MsgBox("Bold element is " + objs.Bold)
MsgBox("Italic element is " + objs.Italic)
MsgBox("Underline element is " + objs.Underline)

num2 = objs.Count
MsgBox("Number of toggling elements with macros is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.Item(cnt1)
MsgBox("Name = " + obj1.name + " MacroName = " + obj1.MacroName)
Next
```

Italic

The name of the Italic toggling element. The selected text is surrounded with this element when the Italic button is clicked on the Formatting toolbar in XMetaL Author. You can set this property using the Italic property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strItalicTogglingElementName = ToggligElements_obj.Italic;
```

Usage in VBScript

```
dim strItalicTogglingElementName = ToggligElements_obj.Italic
```

Example

```
'XMetaL Script Language VBScript:
'displays all of the toggling elements in a document's customization

Set ctm = ActiveDocument.Customizations
Set objs = ctm.TogglingElements

MsgBox("Bold element is " + objs.Bold)
MsgBox("Italic element is " + objs.Italic)
MsgBox("Underline element is " + objs.Underline)

num2 = objs.Count
MsgBox("Number of toggling elements with macros is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.Item(cnt1)
MsgBox("Name = " + obj1.name + " MacroName = " + obj1.MacroName)
Next
```

item
Returns the TogglingElement object for the indicated item.

Applies to

XMetaL XMAXand XMetaL Author

Access

Read only

Returns

TogglingElement object

Usage in JScript

`longIndex` is an integer between 0 and the value represented by `TogglingElements.count-1`

Usage in VBScript

Example

```
'XMetaL Script Language VBScript:
'displays all of the toggling elements in a document's customization

Set ctm = ActiveDocument.Customizations
Set objs = ctm.TogglingElements

MsgBox("Bold element is " + objs.Bold)
MsgBox("Italic element is " + objs.Italic)
MsgBox("Underline element is " + objs.Underline)

num2 = objs.Count
MsgBox("Number of toggling elements with macros is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.Item(cnt1)
```

```
MsgBox("Name = " + obj1.name + " MacroName = " + obj1.MacroName)
Next
```

Underline
The name of the Underline toggling element. The selected text is surrounded with this element when the Underline button is clicked on the Formatting toolbar in XMetaL Author. You can set this property using the Underline property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strUnderlineTogglingElementName = ToggligElements_obj.Underline;
```

Usage in VBScript

```
dim strUnderlineTogglingElementName = ToggligElements_obj.Underline
```

Example

```
'XMetaL Script Language VBScript:
'displays all of the toggling elements in a document's customization

Set ctm = ActiveDocument.Customizations
Set objs = ctm.TogglingElements

MsgBox("Bold element is " + objs.Bold)
MsgBox("Italic element is " + objs.Italic)
MsgBox("Underline element is " + objs.Underline)

num2 = objs.Count
MsgBox("Number of toggling elements with macros is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.Item(cnt1)
MsgBox("Name = " + obj1.name + " MacroName = " + obj1.MacroName)
Next
```

TreatAsImage

The TreatAsImage interface represents a collection of image properties that are used when an element is treated as an image in a document customization. This interface can be acquired only from the TreatAsImages interface.

Properties

AltText

Returns the name of the attribute used to store the alternate text for an element treated as an image.This property is set using the Alternate Text property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strAltTextAttribute = TreatAsImage_obj.AltText;
```

Usage in VBScript

```
dim strAltTextAttribute = TreatAsImage_obj.AltText
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.images
num2 = objs.count
MsgBox("Number of images is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source + " height = "
 + obj1.height + "width = " + obj1.width + " AltText = " + obj1.alttext
 + "VAlign = " + obj1.VerticalAlign + "Scale = " + obj1.Scale)
Next
```

Height

Returns the name of the attribute used to store the height of an element treated as an image. This property is set using the Height property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strHeight = TreatAsImage_obj.Height;
```

Usage in VBScript

```
dim strHeight = TreatAsImage_obj.Height
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.images
num2 = objs.count
MsgBox("Number of images is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source + " height = "
 + obj1.height + "width = " + obj1.width + " AltText = " + obj1.alttext
 + "VAlign = " + obj1.VerticalAlign + "Scale = " + obj1.Scale)
Next
```

name

Returns the name of the element treated as an image.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strName = TreatAsImage_obj.name;
```

Usage in VBScript

```
dim strName = TreatAsImage_obj.name
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.images
num2 = objs.count
MsgBox("Number of images is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source + " height = "
 + obj1.height + "width = " + obj1.width + " AltText = " + obj1.alttext
```

```
 + "VAlign = " + obj1.VerticalAlign + "Scale = " + obj1.Scale)
Next
```

Scale
Returns the name of the attribute used to store the scale value of an element treated as an image.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strScaleAttrName = TreatAsImage_obj.Scale;
```

Usage in VBScript

```
dim strScaleAttrName = TreatAsImage_obj.Scale
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.images
num2 = objs.count
MsgBox("Number of images is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source + " height = "
 + obj1.height + "width = " + obj1.width + " AltText = " + obj1.alttext
 + "VAlign = " + obj1.VerticalAlign + "Scale = " + obj1.Scale)
Next
```

Source
Returns the name of the attribute used to store the path to the image file. This property is set using the Source property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strSource = TreatAsImage_obj.Source;
```

Usage in VBScript

```
dim strSource = TreatAsImage_obj.Source
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.images
num2 = objs.count
MsgBox("Number of images is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source + " height = "
 + obj1.height + "width = " + obj1.width + " AltText = " + obj1.alttext
 + "VAlign = " + obj1.VerticalAlign + "Scale = " + obj1.Scale)
Next
```

VerticalAlign

Returns the name of the attribute used to store the vertical alignment of an element treated as an image.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strVAlignAttrName = TreatAsImage_obj.VerticalAlign;
```

Usage in VBScript

```
dim strVAlignAttrName = TreatAsImage_obj.VerticalAlign
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.images
num2 = objs.count
MsgBox("Number of images is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source + " height = "
 + obj1.height + "width = " + obj1.width + " AltText = " + obj1.alttext
```

```
 + "VAlign = " + obj1.VerticalAlign + "Scale = " + obj1.Scale)
Next
```

Width
Returns the name of the attribute used to store the width of an element treated as an image. This property is set using the Width property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strWidth = TreatAsImage_obj.Width;
```

Usage in VBScript

```
dim strWidth = TreatAsImage_obj.Width
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.images
num2 = objs.count
MsgBox("Number of images is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source + " height = "
 + obj1.height + "width = " + obj1.width + " AltText = " + obj1.alttext
 + "VAlign = " + obj1.VerticalAlign + "Scale = " + obj1.Scale)
Next
```

TreatAsImages

The TreatAsImages interface can be acquired only from the Customizations interface. It stores a collection of elements treated as images. Use the TreatAsImage interface to get the properties of each element that is treated as an image.

Properties

count
Returns the number of elements treated as images in the document's customization.

Applies to

XMetaL XMAXand XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
var intNumImages = TreatAsImages_obj.count;
```

Usage in VBScript

```
dim intNumImages = TreatAsImages_obj.count
```

Example

```
'XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.images
num2 = objs.count
MsgBox("Number of images is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source + " height = "
+obj1.height + " width = " + obj1.width + " AltText = " + obj1.alttext)
Next
```

item

Returns the TreatAsImage object for the item located at the indicated position.

Applies to

XMetaL XMAXand XMetaL Author

Access

Read only

Returns

TreatAsImage object

Usage in JScript

```
var treatAsImageItem = TreatAsImages_obj.item(i);
```

`longIndex` is an integer between 0 and the value represented by `TreatAsImages.count`-1.

Usage in VBScript

```
set treatAsImageItem = TreatAsImages_obj.item(i)
```

Example

```
'XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.images
num2 = objs.count
MsgBox("Number of images is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source + " height = "
+obj1.height + " width = " + obj1.width + " AltText = " + obj1.alttext)
Next
```

TreatAsLink

The TreatAsLink interface represents a collection of link properties that are used when an element is treated as an link in a document customization. This interface can be acquired only from the TreatAsLinks interface.

Properties

Name

Returns the name of the element treated as an link.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strName = TreatAsLink_obj.name;
```

Usage in VBScript

```
dim strName = TreatAsLink_obj.name
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.Links
num2 = objs.count
MsgBox("Number of links is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
```



```
MsgBox("Name = " + obj1.name + " source = " + obj1.source)
Next
```

Source
Returns the name of the attribute used to store the path to the link file. This property is set using the Source property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strSource = TreatAsLink_obj.Source;
```

Usage in VBScript

```
dim strSource = TreatAsLink_obj.Source
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.links
num2 = objs.count
MsgBox("Number of links is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source)
Next
```

ChooserFilterString
Returns the string for suggested usage in an open/chooser dialog's "Files of type:" combobox for an element treated as an link.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strChooserFilterString = TreatAsLink_obj.ChooserFilterString;
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.links
num2 = objs.count
MsgBox("Number of links is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Chooser Filter String = " + obj1.ChooserFilterString)
Next
```

ChooserTitleString

Returns the string for suggested usage in an open/chooser dialog's caption for an element treated as an link.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strChooserTitleString = TreatAsLink_obj.ChooserTitleString;
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.links
num2 = objs.count
MsgBox("Number of links is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Chooser Title String = " + obj1.ChooserTitleString)
Next
```

EnableBuiltInChooser

Returns true if XMetaL should provide UI for setting the Source of an element treated as an link.This property is set using the Enable BuiltIn Chooser property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var boolEnableBuiltInChooser = TreatAsLink_obj.EnableBuiltInChooser;
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.links
num2 = objs.count
MsgBox("Number of links is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Enable BuiltIn Chooser = " + obj1.EnableBuiltInChooser)
Next
```

EnableBuiltInFetcher

Returns true if XMetaL should delegate to the CMS to retrieve the target indicated by the Source of an element treated as an link. This property is set using the Enable BuiltIn Fetcher property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var boolEnableBuiltInFetcher = TreatAsLink_obj.EnableBuiltInFetcher;
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.links
num2 = objs.count
MsgBox("Number of links is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Enable BuiltIn Fetcher = " + obj1.EnableBuiltInFetcher)
Next
```

EnableFolderLink
Returns true if XMetaL should display a folder-style Chooser dialog instead of a file-style Chooser dialog. Typically, this only applies to the local file system and for use in setting up xml:base. This property is set using the Enable Folder Link property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var boolEnableFolderLink = TreatAsLink_obj.EnableFolderLink;
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.links
num2 = objs.count
MsgBox("Number of links is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Enable Folder Link = " + obj1.EnableFolderLink)
Next
```

EnableRebase
Returns true if XMetaL should adjust the Source attribute's value during Save-As operations. Typically, this property is only setup for document customization meant for the local file system. This property is set using the Enable Rebase property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var boolEnableRebase = TreatAsLink_obj.EnableRebase;
```


Example

```
' XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.Links
num2 = objs.count
MsgBox("Number of links is " + cstr(num2))
For cnt1 = 0 To num2 - 1
 MsgBox(cnt1)
 Set obj1 = objs.item(cnt1)
 MsgBox("Enable Folder Link = " + cstr(obj1.EnableRebase))
Next
```

ParentList
Returns the list of parent element names of the element treated as a link.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strParentNames = TreatAsLink_obj.ParentList;
```

Usage in VBScript

```
dim strParentNames = TreatAsLink_obj.ParentList
```

Example

```
' XMetaL Script Language VBSCRIPT:
var ctm = ActiveDocument.Customizations; var links = ctm.Links; var len = links.count;
for (var idx = 0; idx < len; ++idx) {
  var link = links.item(idx);
  Application.Alert(link.ParentList);
}
```

TreatAsLinks

The TreatAsLinks interface can be acquired only from the Customizations interface. It stores a collection of elements treated as links. Use the TreatAsLink interface to get the properties of each element that is treated as an link.

Properties

count

Returns the number of elements treated as links in the document's customization.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
var intNumLinks = TreatAsLinks_obj.count;
```

Usage in VBScript

```
dim intNumLinks = TreatAsLinks_obj.count
```

Example

```
'XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.Links
num2 = objs.count
MsgBox("Number of links is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source)
Next
```

item

Returns the TreatAsLink object for the item located at the indicated position.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

TreatAsLink object

Usage in JScript

```
var treatAsLinkItem = TreatAsLinks_obj.item(i);
```


`longIndex` is an integer between 0 and the value represented by `TreatAsLinks.count-1`.

Usage in VBScript

```
set treatAsLinkItem = TreatAsLinks_obj.item(i)
```

Example

```
'XMetaL Script Language VBSCRIPT:
Set ctm = ActiveDocument.Customizations
Set objs = ctm.Links
num2 = objs.count
MsgBox("Number of links is " + cstr(num2))
For cnt1 = 0 To num2 - 1
MsgBox(cnt1)
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " source = " + obj1.source)
Next
```

TreatAsList

The TreatAsList interface represents a collection of list properties. These properties are used when an element is treated as a list in a document customization. This interface can be acquired only from the TreatAsLists interface.

Properties

Definition

The name of the definition child element. You can set this property using the Definition property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strDefinition = TreatAsList_obj.Definition;
```

Returns the name of an element only if the list type is a definition list (that is, `TreatAsList.type = sqDefinition`). Returns an empty string if the list type is `sqUnordered` or `sqOrdered`.

Usage in VBScript

```
dim strDefinition = TreatAsList_obj.Definition;
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Lists
num2 = objs.count
MsgBox("Number of lists is " + cstr(num2))

For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)

if (obj1.type = 2) Then
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " Term = "
 + obj1.Term + " Definition = " + obj1.Definition)
Else
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " ListHeader
 = " + obj1.ListHeader + " ListItem = " + obj1.ListItem)
End If
Next
```

ListHeader
Returns the name of the element used as a list header. A list header element is typically used to store information that appears before the body of the list is displayed (such as a title). You can set this property using the ListHeader property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strListHeader = TreatAsList_obj.ListHeader;
```

Returns the name of an element only ifa list header is defined, and the list type is bulleted list (that is, `TreatAsList.type = sqUnordered`) or numbered list (`TreatAsList.type = sqOrdered`).Returns an empty string if a list header is undefined or the list type is `sqDefinition`.

Usage in VBScript

```
dim strListHeader = TreatAsList_obj.ListHeader
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Lists
num2 = objs.count
MsgBox("Number of lists is " + cstr(num2))
```



```
For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)

if (obj1.type = 2) Then
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " Term = "
 + obj1.Term + " Definition = " + obj1.Definition)
Else
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " ListHeader
 = " + obj1.ListHeader + " ListItem = " + obj1.ListItem)
End If
Next
```

ListItem
Returns the name of the child element used as a list item. You can set this property using the ListItem property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strListItem = TreatAsList.ListItem;
```

Returns the name of an element only ifthe list type is bulleted list (that is, `TreatAsList.type = sqUnordered`) or numbered list (`TreatAsList.type = sqOrdered`).Returns an empty string if the list type is `sqDefinition`.

Usage in VBScript

```
dim strListItem = TreatAsList.ListItem
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Lists
num2 = objs.count
MsgBox("Number of lists is " + cstr(num2))

For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)

if (obj1.type = 2) Then
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " Term = "
 + obj1.Term + " Definition = " + obj1.Definition)
Else
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " ListHeader
 = " + obj1.ListHeader + " ListItem = " + obj1.ListItem)
```

```
End If
Next
```

name
Returns the name of the list element.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strName = TreatAsList_obj.name;
```

Usage in VBScript

```
dim strName = TreatAsList_obj.name
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Lists
num2 = objs.count
MsgBox("Number of lists is " + cstr(num2))

For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)

if (obj1.type = 2) Then
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " Term = "
 + obj1.Term + " Definition = " + obj1.Definition)
Else
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " ListHeader
 = " + obj1.ListHeader + " ListItem = " + obj1.ListItem)
End If
Next
```

Term
Returns the name of the term child element. You can set this property using the Term property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strTerm = TreatAsList_obj.Term;
```

Returns the name of an element only if the list type is a definition list (that is, `TreatAsList.type = sqDefinition`).Returns an empty string if the list type is `sqUnordered` or `sqOrdered`.

Usage in VBScript

```
dim strTerm = TreatAsList_obj.Term
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Lists
num2 = objs.count
MsgBox("Number of lists is " + cstr(num2))

For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)

if (obj1.type = 2) Then
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " Term = "
 + obj1.Term + " Definition = " + obj1.Definition)
Else
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " ListHeader
 = " + obj1.ListHeader + " ListItem = " + obj1.ListItem)
End If
Next
```

type

Returns the type of the list. You can set this property using the Term property in the customization file editor.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Integer

Definition

Usage in JScript

```
varintType = TreatAsList_obj.type;
```

The allowed values are:

- sqUnordered (0) = bulleted list
- sqOrdered (1) = numbered list
- sqDefinition (2) = definition list

If the list is bulleted (`sqUnordered`) or numbered (`sqOrdered`), the ListHeader and ListItem properties return non-empty values. If the list is of type definition (`sqDefinition`), the Term and Definition properties return non-empty values.

Usage in VBScript

```
dimintType = TreatAsList_obj.type
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Lists
num2 = objs.count
MsgBox("Number of lists is " + cstr(num2))

For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)

if (obj1.type = 2) Then
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " Term = "
 + obj1.Term + " Definition = " + obj1.Definition)
Else
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " ListHeader
 = " + obj1.ListHeader + " ListItem = " + obj1.ListItem)
End If
Next
```

TreatAsLists

The TreatAsLists interface can be acquired only from the Customizations interface. It stores a collection of elements treated as lists. Use the TreatAsList interface to get the properties of each element that is treated as a list.

Properties

count

Returns the number of elements treated as lists in the document's customization.

Applies to

XMetaL XMAXand XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
var intNumLists = TreatAsLists_obj.count;
```

Usage in VBScript

```
dim intNumLists = TreatAsLists_obj.count
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Lists
num2 = objs.count
MsgBox("Number of lists is " + cstr(num2))

For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)

if (obj1.type = 2) Then
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " Term = "
 + obj1.Term + " Definition = " + obj1.Definition)
Else
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " ListHeader
 = " + obj1.ListHeader + " ListItem = " + obj1.ListItem)
End If
Next
```

item
Returns the TreatAsList object for the item specified.

Applies to

XMetaL XMAXand XMetaL Author

Access

Read only

Returns

TreatAsList object

Usage in JScript

```
var listObject = TreatAsLists_obj.item(longIndex);
```

`longIndex` is an integer between 0 and the value represented by `TreatAsLists.count-1` .

Usage in VBScript

```
set listObject = TreatAsLists_obj.item(longIndex)
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Lists
num2 = objs.count
MsgBox("Number of lists is " + cstr(num2))

For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)

if (obj1.type = 2) Then
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " Term = "
 + obj1.Term + " Definition = " + obj1.Definition)
Else
MsgBox("Type = " + cstr(obj1.type) + " Name = " + obj1.name + " ListHeader
 = " + obj1.ListHeader + " ListItem = " + obj1.ListItem)
End If
Next
```

TreatAsParagraph

The TreatAsParagraph interface represents a collection of paragraph properties. These properties are used when an element is treated as a paragraph in a document customization. This interface can be acquired only from the TreatAsParagraphs interface.

Properties

name

Returns the name of the element treated as a paragraph.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strName = TreatAsParagraph_obj.name;
```

Usage in VBScript

```
dim strName = TreatAsParagraph_obj.name
```


Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Paragraphs

num2 = objs.count
MsgBox("Number of paragraphs is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " Parent = """ + obj1.Parent + """")
Next
```

Parent
Returns the name of the parent of the element treated as a paragraph. Returns the empty string if no parent exists.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strParentName = TreatAsParagraph_obj.Parent;
```

Usage in VBScript

```
dim strParentName = TreatAsParagraph_obj.Parent
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Paragraphs

num2 = objs.count
MsgBox("Number of paragraphs is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " Parent = """ + obj1.Parent + """")
Next
```

TreatAsParagraphs

The TreatAsParagraphs interface can be acquired only from the Customizations interface. It stores a collection of elements treated as paragraphs. Use the TreatAsParagraph interface to get the properties of each individual element treated as a paragraph.

Properties

count

Returns the number of elements treated as paragraphs in the document's customization.

Applies to

XMetaL XMAXand XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
var intNumImages = TreatAsParagraphs_obj.count;
```

Usage in VBScript

```
dim intNumImages = TreatAsParagraphs_obj.count
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Paragraphs

num2 = objs.count
MsgBox("Number of paragraphs is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " Parent = """ + obj1.Parent + """")
Next
```

item

Returns the TreatAsParagraph object for the item located at the indicated position.

Applies to

XMetaL XMAXand XMetaL Author

Access

Read only

Returns

TreatAsParagraph object

Usage in JScript

```
var paragraph = TreatAsParagraphs_obj.item(longIndex);
```

`longIndex` is an integer between 0 and the value represented by `TreatAsParagraphs.count-1` .

Usage in VBScript

```
dim paragraph = TreatAsParagraphs_obj.item(longIndex)
```

Example

```
'XMetaL Script Language VBScript:

Set ctm = ActiveDocument.Customizations
Set objs = ctm.Paragraphs

num2 = objs.count
MsgBox("Number of paragraphs is " + cstr(num2))
For cnt1 = 0 To num2-1
Set obj1 = objs.item(cnt1)
MsgBox("Name = " + obj1.name + " Parent = """ + obj1.Parent + """")
Next
```

ValidationError

The ValidationError interface provides information about individual validation errors in a selection or document. This interface can be be acquired only through the ValidationErrorList interface.

Properties

ErrorLevel

Returns an integer representing the severity of the validation error.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
var myInteger = ActiveDocument.ValidationErrorList.Item(1).ErrorLevel;
```

The following return values are possible:

- velInfo (0): information message
- velWarning (1): warning message
- velError (2): error message
- velFatalError (3): fatal error, the validation process cannot continue

Errors of level 2 and 3 are the most common. These errors indicate that the underlying XML markup is invalid.

Usage in VBScript

```
dim myInteger = ActiveDocument.ValidationErrorList.Item(1).ErrorLevel
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
// web page.
// Validates the active document, and displays a list of validation errors.
// When you click OK, you go to the error numbered gErrNum.
// gErrNum persists (if the script is run in an Internet Explorer web
page),
// and you can run validate() a number of times to visit each
// validation error in the document in sequence.

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myXMControl" HEIGHT="500" WIDTH="500">
</OBJECT>

<SCRIPT>
// define global variable
var gErrnum = 0;

function assert(bool) {
if (bool == false) {
ActiveDocument.Host.Alert("There was a programming error somewhere.");
}
}

function validate() {
var doc = myXMControl.Document;
if (doc) {
doc.Validate();
var vel = doc.ValidationErrorList;

if (vel) {

if (vel.Count == 0) { // valid document
assert(doc.IsValid);
ActiveDocument.Host.Alert("The document is valid.");

} else { // one or more validation errors
assert(!doc.IsValid);
var msg = "Errors:\n\nItem\tLevel\tNumber\tMessage\n";
var index = 1;
var ve = vel.Item(index++);

while (ve) {
msg += index-1 + "\t" + ve.ErrorLevel
+ "\t" + ve.ErrorType
+ "\t" + ve.ErrorMessage;
```



```
ve = vel.Item(index++);
msg += "\n";
}
gErrNum++;

if (gErrNum > vel.Count) {
gErrNum = 1; // go back to the first error
}
ve = vel.Item(gErrNum);
msg += "\nPress OK to go to error " + gErrNum + " (Item " + gErrNum + ")";
ActiveDocument.Host.Alert(msg);

// There are 2 ways to go to the error:
//ValidationErrorList.GoToError(Index)
//ValidationError.Range.Select()
var doItWithGoToError = true;
if (doItWithGoToError) {
vel.GoToError(gErrNum);
} else {
ve.Range.Select();
}
}
}
}
}
</SCRIPT>
```

ErrorMessage
Returns the default error message corresponding to the validation error.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var myString = ActiveDocument.ValidationErrorList.Item(1).ErrorMessage;
```

Usage in VBScript

```
dimmyString = ActiveDocument.ValidationErrorList.Item(1).ErrorMessage
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
// web page.
// Validates the active document, and displays a list of validation errors.
// When you click OK, you go to the error numbered gErrNum.
// gErrNum persists (if the script is run in an Internet Explorer web
page),
// and you can run validate() a number of times to visit each
// validation error in the document in sequence.
```

```
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myXMControl" HEIGHT="500" WIDTH="500">
</OBJECT>

<SCRIPT>
// define global variable
var gErrnum = 0;

function assert(bool) {
if (bool == false) {
ActiveDocument.Host.Alert("There was a programming error somewhere.");
}
}

function validate() {
var doc = myXMControl.Document;
if (doc) {
doc.Validate();
var vel = doc.ValidationErrorList;

if (vel) {

if (vel.Count == 0) { // valid document
assert(doc.IsValid);
ActiveDocument.Host.Alert("The document is valid.");

} else { // one or more validation errors
assert(!doc.IsValid);
var msg = "Errors:\n\nItem\tLevel\tNumber\tMessage\n";
var index = 1;
var ve = vel.Item(index++);

while (ve) {
msg += index-1 + "\t" + ve.ErrorLevel
+ "\t" + ve.ErrorType
+ "\t" + ve.ErrorMessage;
ve = vel.Item(index++);
msg += "\n";
}
gErrNum++;

if (gErrNum > vel.Count) {
gErrNum = 1; // go back to the first error
}
ve = vel.Item(gErrNum);
msg += "\nPress OK to go to error " + gErrNum + " (Item " + gErrNum + ")";
ActiveDocument.Host.Alert(msg);

// There are 2 ways to go to the error:
//ValidationErrorList.GoToError(Index)
//ValidationError.Range.Select()
var doItWithGoToError = true;
if (doItWithGoToError) {
vel.GoToError(gErrNum);
} else {
ve.Range.Select();
}
}
}
}
}
</SCRIPT>
```

ErrorType
Returns the integer error type of the message.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Integer

Description

The following values are possible:

ErrorType constant	ErrorType integer	Abbreviated default error message (contains a substring of the default error message)
vetNoError	0	
vetNameTooLong	1	Name too long.
vetBadTagName	2	Bad or missing tag name.
vetInvalidTag	3	Invalid tag name.
vetUnknownTag	4	Unknown tag name.
vetBadMarkup	5	Bad Markup Declaration.
vetNotAComment	6	Bad Comment Declaration.
vetSDTooLong	7	System Data Too Long.
vetBadCharType	8	Character is of the wrong type.
vetBadDelim	9	Bad literal delimiter. Expected ' or ".
vetBadExternID	10	Bad external identifier.
vetBadDocType	11	Bad DOCTYPE.
vetNotAccepting	12	Missing required elements.
vetStackOverflow	13	Stack Overflow.
vetStackUnderflow	14	Stack Underflow.
vetStackNotEmpty	15	Stack Not Empty.
vetNotMarkup	16	Not Markup.
vetNoSGML	17	No SGML.
vetNoDOCTYPE	18	No DOCTYPE.
vetPEntNameTooLong	19	Parameter entity name too long.
vetPEntUndefined	20	Parameter entity used but not defined.
vetEntUndefined	21	Entity <entity name> is referenced but not defined.

vetEntNestTooDeep	22	Entity nesting limit exceeded.
vetBadEntName	23	Bad or missing entity name.
vetMissingPSep	24	Missing parameter separator.
vetNoMinimization	25	No Minimization. Expected '-' or '0'.
vetBadElemDec	26	Bad element declaration.
vetBadDeclaredContent	27	Bad declared content.
vetBadContentModel	28	Bad content model.
vetMissingElement	29	Missing element.
vetBadNameGroup	30	Bad Name Group.
vetTooManyNames	31	Too many names in name group.
vetRedefnElement	32	Tag name has already been declared.
vetBadMinimization	33	Bad Minimization. Expected '-' or '0'.
vetExcededGRPLVL	34	Model groups nested too deeply.
vetEmptyModelGroup	35	Empty Model Group.
vetBadDTDecSub	36	Bad characters in Declaration Subset. Expected '<!'.
vetNoContent	37	Element used but not defined.
vetElemNotUsed	38	Element declaredbut not used.
vetBadAttList	39	Bad ATTLIST declaration.
vetAttNameNotGI	40	Attempting to declare attributes for undeclared element.
vetBadAttrName	41	Bad or missing attribute name.
vetBadAttDecValue	42	Bad declared value for attribute.
vetBadAttrDflt	43	Bad default for attribute.
vetBadAttValue	44	Bad or missing attribute value.
vetExcededATTCNT	45	Too many names and name tokens in attribute list.
vetAttNameRedefn	46	Attribute name already used in this ATTLIST declaration.
vetAttTokenReused	47	Name (token) already used in this attribute definition list.
vetLitTooLong	48	Literal too long.
vetAttListRedfn	49	Redefining attributes for an element.
vetCommentEOF	50	End of file in comment.
vetEOFInAttSpec	51	End of file in attribute value (probably a missing quote).
vetBadPEntName	52	Bad or missing parameter entity name.
vetBadNotationName	53	Bad or missing notation name.
vetBadSystemID	54	Bad system identifier. Expected \ or '.

vetNotationRedefined	55	NOTATION can only be declared once in an attribute definition list.
vetIDRedefined	56	ID can only be declared once in an attribute definition list.
vetBadNotation	57	NOTATION cannot be declared for an element whose declared content is empty.
vetBadCONREF	58	CONREF cannot be declared for an element whose declared content is EMPTY.
vetBadIDDflt	59	An attribute with declared value type of ID must have a default value type of either IMPLIED or REQUIRED.
vetRCharDataEe	60	Ending entity containing repleacdeable character data without ending teh data.
vetBadTokenType	61	Character(s) of incorrect type found in string.
vetBadFunctionName	62	Bad function name in character reference.
vetDTSpec	63	Document type specification not allowed unless CONCUR or EXPLICIT are YES.
vetBadStartTag	64	Bad start tag.
vetBadEndTag	65	Bad end tag.
vetBadAttSpec	66	Bad attribute specification.
vetCharNotAllowed	67	Text not allowed here.
vetBadEntityText	68	Error in entity text.
vetPITooLong	69	Processing instruction is too long.
vetBadMarkedSection	70	Bad Marked Section.
vetNoQStatusKW	71	Marked section status cannot be qualified.
vetBadMSEnd	72	Found a marked section end, "]]>", without a marked section start.
vetIgnoreMSEe	73	An entity cannot end in an ignored version or marked section.
vetTSEe	74	Bad entity end in token separators.
vetNameReused	75	A token can only occur once in a single name group or name token group.
vetPEntOpen	76	A parameter entity must end in the same group it started in.
vetBadConnector	77	Only one kind of connector can appear in a single model group.
vetBadEe	78	Bad entity end.
vetRedecNotation	79	NOTATION redeclared.
vetGRPGTCNTExceeded	80	Too many tokens in model group.
vetEmptyAttVLit	81	Empty attribute value.

vetBadAttSpecList	82	Bad attribute specification list.
vetBadSGMLDec	83	Bad SGML declaration.
vetBadCSyntaxScope	84	Bad Concrete Syntax Scope specification.
vetBadCSyntax	85	Bad Concrete Syntax specification.
vetBadQuantitySet	86	Bad Quantity Set specification.
vetBadDocCharSet	87	Bad Document Character Set specification.
vetBadCapacitySet	88	Bad Capacity Set specification.
vetBadFeatureUse	89	Bad Feature Use specification.
vetBadFormalID	90	Bad Formal Public Identifier.
vetAmbiguity	91	Ambiguous content model.
vetNonuniqueID	92	The ID attribute has already been used in a previous tag.
vetMissingID	93	Missing referenced ID.
vetNotDataEntity	94	Only references to data entities are allowed in ENTITY/ENTITIES attributes.
vetNotationUndefined	95	Referenced entity uses undefined notation.
vetBadNameChar	96	Error in NAMING rules section of the SGML Declaration.
vetDTNameUndefined	97	The document type name must be a genericidentifier.
vetBadAppInfo	98	Error in the APPINFO section of the SGML Declaration.
vetEntAlreadyOpen	99	A referenced entity is already open.
vetNoSuchAttrName	100	Undefined attribute.
vetNoSuchAttrValue	101	Unrecognized attribute value.
vetNoXMLMinimization	102	XML does not allow tag minimization specifications.
vetBadCDATASection	103	Marked section start does not match CDATA section.
vetCharNotSupported	104	Unsupported Unicode character.
vetErrorEncountered	105	The document is not valid.
vetImpliedStartTag	106	Implied missing start-tag.
vetImpliedEndTag	107	Implied missing end-tag.
vetIgnoringEndTag	108	Ignoring end-tag.
vetIgnoringStartTag	109	Ignoring start-tag.
vetChangedStartTag	110	Changed start-tag.
vetTooManyErrors	111	Can't continue. Too many errors.
vetTooManyWarnings	112	Too many warnings.
vetAttrValueTooLong	113	Attribute value is too long.

vetUnquotedAttrValue	114	Attribute value is not quoted.
vetNotAttrNoSupport	115	notation attributes are not supported.
vetNonDeterministic	116	Non deterministic content model.
vetEncodingNotSupported	117	Unsupported encoding.
vetDuplicateAttribute	118	Duplicate attribute.
vetNoSuchXMLAttrName	119	Undefined attribute.
vetBadSchemaLocation	120	Invalid value for attribute "schemaLocation".
vetBadNoNSSchemaLocation	121	Invalid value for attribute "noNamespaceLocation".
vetUndeclaredNSPrefix	122	Undeclared namespace prefix.
vetBadSimpleTypeAttrValue	123	Bad simple type attribute value.
vetBadSimpleContentElemValue	124	Bad simple content element value.
vetBadSchema	125	An error occurred while loading XML Schema.
vetNoSemiColonAtEndOfRef	126	A semi-colong was expected. Maybe an "&" character was not encoded as "&"?
vetBadNamespaceDecl	127	Invalid Namespace declaration.
vetBadContent	128	Bad content.
vetBadCharacterRef	129	Bad character reference.
vetExternEntRefInAttVal	130	Attribute values cannot contain external entity references.
vetLINK	131	LINK not supported.
vetCONCUR	132	CONCUR not supported.
vetDataTag	133	DataTag not supported.
vetBadMagic	134	Rules file is not compiled.
vetBadRelease	135	Incorrect release.
vetBadFileType	136	Incorrect file format.
vetBadMachineType	137	Incorrect machine type.
vetOutOfMemory	138	Out Of Memory.
vetBadEOF	139	Unexpected End of File.
vetCantWriteRulesFile	140	Unable to create rules file.
vetSilentFatalError	141	eSilentFatalError.
vetGeneralError	142	General Error.
vetCancel	143	Cancel.
vetOpenWellFormed	144	Open document in well-formed editing mode.
vetAmpersandInAttributeValue	145	An "&" must begin an entity reference. Write "&" for a "&".

vetLessThanInAttributeValue	146	A "<" must begin a start tag. Write "<" for a "<".

Usage in JScript

```
var myInteger = ActiveDocument.ValidationErrorList.Item(1).ErrorType;
```

Usage in VBScript

```
dim myInteger = ActiveDocument.ValidationErrorList.Item(1).ErrorType
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
// web page.
// Validates the active document, and displays a list of validation errors.
// When you click OK, you go to the error numbered gErrNum.
// gErrNum persists (if the script is run in an Internet Explorer web
page),
// and you can run validate() a number of times to visit each
// validation error in the document in sequence.

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myXMControl" HEIGHT="500" WIDTH="500">
</OBJECT>

<SCRIPT>
// define global variable
var gErrnum = 0;

function assert(bool) {
if (bool == false) {
ActiveDocument.Host.Alert("There was a programming error somewhere.");
}
}

function validate() {
var doc = myXMControl.Document;
if (doc) {
doc.Validate();
var vel = doc.ValidationErrorList;

if (vel) {

if (vel.Count == 0) { // valid document
assert(doc.IsValid);
ActiveDocument.Host.Alert("The document is valid.");

} else { // one or more validation errors
assert(!doc.IsValid);
var msg = "Errors:\n\nItem\tLevel\tNumber\tMessage\n";
var index = 1;
var ve = vel.Item(index++);

while (ve) {
msg += index-1 + "\t" + ve.ErrorLevel
+ "\t" + ve.ErrorType
+ "\t" + ve.ErrorMessage;
ve = vel.Item(index++);
msg += "\n";
}
gErrNum++;

if (gErrNum > vel.Count) {
gErrNum = 1; // go back to the first error
}
```

```
ve = vel.Item(gErrNum);
msg += "\nPress OK to go to error " + gErrNum + " (Item " + gErrNum + ")";
ActiveDocument.Host.Alert(msg);

// There are 2 ways to go to the error:
//ValidationErrorList.GoToError(Index)
//ValidationError.Range.Select()
var doItWithGoToError = true;
if (doItWithGoToError) {
vel.GoToError(gErrNum);
} else {
ve.Range.Select();
}
}
}
}
}
</SCRIPT>
```

Range
Returns the Range object corresponding to the location of the ValidationError. Calling `Range.Select(i)` on `ValidationErrorList` has the same effect as calling `ValidationErrorList.GoToError(i)`.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Range object

Usage in JScript

```
var myRange = ActiveDocument.ValidationErrorList.Item(1).Range;
```

Usage in VBScript

```
dim myRange = ActiveDocument.ValidationErrorList.Item(1).Range
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
// web page.
// Validates the active document, and displays a list of validation errors.
// When you click OK, you go to the error numbered gErrNum.
// gErrNum persists (if the script is run in an Internet Explorer web
page),
// and you can run validate() a number of times to visit each
// validation error in the document in sequence.
<OBJECT CLASSID="xmetal.pg_CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="xmetal.pg_myXMControl" HEIGHT="500" WIDTH="500">
</OBJECT>

<SCRIPT>
// define global variable
```

```
var gErrnum = 0;

function assert(bool) {
if (bool == false) {
ActiveDocument.Host.Alert("There was a programming error somewhere.");
}
}

function validate() {
var doc = myXMControl.Document;
if (doc) {
doc.Validate();
var vel = doc.ValidationErrorList;

if (vel) {

if (vel.Count == 0) { // valid document
assert(doc.IsValid);
ActiveDocument.Host.Alert("The document is valid.");

} else { // one or more validation errors
assert(!doc.IsValid);
var msg = "Errors:\n\nItem\tLevel\tNumber\tMessage\n";
var index = 1;
var ve = vel.Item(index++);

while (ve) {
msg += index-1 + "\t" + ve.ErrorLevel
+ "\t" + ve.ErrorType
+ "\t" + ve.ErrorMessage;
ve = vel.Item(index++);
msg += "\n";
}
gErrNum++;

if (gErrNum > vel.Count) {
gErrNum = 1; // go back to the first error
}
ve = vel.Item(gErrNum);
msg += "\nPress OK to go to error " + gErrNum + " (Item " + gErrNum + ")";
ActiveDocument.Host.Alert(msg);

// There are 2 ways to go to the error:
//ValidationErrorList.GoToError(Index)
//ValidationError.Range.Select()
var doItWithGoToError = false;
if (doItWithGoToError) {
vel.GoToError(gErrNum);
} else {
ve.Range.Select();
}
}
}
}
}
</SCRIPT>
```

ValidationErrorList

The ValidationErrorList interface is a collection of validation errors received during the most recent document or selection validation. You can acquire this interface from the Document or Selection interfaces.

Properties

Count

Returns the number of elements in the ValidationErrorList.

Applies to

XMetaL Author and XMetaL XMAX

Access

Read only

Returns

Integer

Usage in JScript

```
ActiveDocument.ValidationErrorList.Count;
```

Usage in VBScript

```
ActiveDocument.ValidationErrorList.Count
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
// web page.
// Validates the active document, and displays a list of validation errors.
// When you click OK, you go to the error numbered gErrNum.
// gErrNum persists (if the script is run in an Internet Explorer web
page),
// and you can run validate() a number of times to visit each
// validation error in the document in sequence.

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myXMControl" HEIGHT="500" WIDTH="500">
</OBJECT>
<SCRIPT>
// define global variable
var gErrnum = 0;

function assert(bool) {
if (bool == false) {
ActiveDocument.Host.Alert("There was a programming error somewhere.");
}
}

function validate() {
var doc = myXMControl.Document;
if (doc) {
doc.Validate();
var vel = doc.ValidationErrorList;

if (vel) {

if (vel.Count == 0) { // valid document
assert(doc.IsValid);
ActiveDocument.Host.Alert("The document is valid.");
```

```
} else { // one or more validation errors
assert(!doc.IsValid);
var msg = "Errors:\n\nItem\tLevel\tNumber\tMessage\n";
var index = 1;
var ve = vel.Item(index++);

while (ve) {
msg += index-1 + "\t" + ve.ErrorLevel
+ "\t" + ve.ErrorType
+ "\t" + ve.ErrorMessage;
ve = vel.item(index++);
msg += "\n";
}
gErrNum++;

if (gErrNum > vel.Count) {
gErrNum = 1; // go back to the first error
}
ve = vel.item(gErrNum);
msg += "\nPress OK to go to error " + gErrNum + " (Item " + gErrNum + ")";
ActiveDocument.Host.Alert(msg);

// There are 2 ways to go to the error:
//ValidationErrorList.GoToError(Index)
//ValidationError.Range.Select()
var doItWithGoToError = true;
if (doItWithGoToError) {
vel.GoToError(gErrNum);
} else {
ve.Range.Select();
}
}
}
}
}
</SCRIPT>
```

item
Returns the validation error located at the specified position in the ValidationErrorList. .

Applies to

XMetaL Author and XMetaL XMAX

Returns

ValidationError object

Usage in JScript

```
var myItem = ActiveDocument.ValidationErrorList.item(1);
```

`intIndex` is an integer between 1 and the value represented by `ValidationErrorList.count`. If you pass an integer outside this range to Item, it will return null. Item will also return null if the ValidationError located at `intIndex` is null.

Usage in VBScript

```
dim myItem = ActiveDocument.ValidationErrorList.item(1)
```


Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
// web page.
// Validates the active document, and displays a list of validation errors.
// When you click OK, you go to the error numbered gErrNum.
// gErrNum persists (if the script is run in an Internet Explorer web
page),
// and you can run validate() a number of times to visit each
// validation error in the document in sequence.

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myXMControl" HEIGHT="500" WIDTH="500">
</OBJECT>

<SCRIPT>
// define global variable
var gErrnum = 0;

function assert(bool) {
if (bool == false) {
ActiveDocument.Host.Alert("There was a programming error somewhere.");
}
}

function validate() {
var doc = myXMControl.Document;
if (doc) {
doc.Validate();
var vel = doc.ValidationErrorList;

if (vel) {

if (vel.Count == 0) { // valid document
assert(doc.IsValid);
ActiveDocument.Host.Alert("The document is valid.");

} else { // one or more validation errors
assert(!doc.IsValid);
var msg = "Errors:\n\nItem\tLevel\tNumber\tMessage\n";
var index = 1;
var ve = vel.item(index++);

while (ve) {
msg += index-1 + "\t" + ve.ErrorLevel
+ "\t" + ve.ErrorType
+ "\t" + ve.ErrorMessage;
ve = vel.Item(index++);
msg += "\n";
}
gErrNum++;

if (gErrNum > vel.Count) {
gErrNum = 1; // go back to the first error
}
ve = vel.item(gErrNum);
msg += "\nPress OK to go to error " + gErrNum + " (Item " + gErrNum + ")";
ActiveDocument.Host.Alert(msg);

// There are 2 ways to go to the error:
//ValidationErrorList.GoToError(Index)
//ValidationError.Range.Select()
var doItWithGoToError = true;
if (doItWithGoToError) {
vel.GoToError(gErrNum);
} else {
ve.Range.Select();
}
```

```
}
}
}
}
</SCRIPT>
```

Methods

GoToError

Moves the user's selection to the ValidationError specified.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
ActiveDocument.ValidationErrorList.GoToError(longIndex);
```

`longIndex` should be an integer between 1 and the value represented by `ValidationErrorList.count`.

Usage in VBScript

```
ActiveDocument.ValidationErrorList.GoToError(longIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
// web page.
// Validates the active document, and displays a list of validation errors.
// When you click OK, you go to the error numbered gErrNum.
// gErrNum persists (if the script is run in an Internet Explorer web
page),
// and you can run validate() a number of times to visit each
// validation error in the document in sequence.

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myXMControl" HEIGHT="500" WIDTH="500">
</OBJECT>

<SCRIPT>
// define global variable
var gErrnum = 0;

function assert(bool) {
if (bool == false) {
ActiveDocument.Host.Alert("There was a programming error somewhere.");
}
}

function validate() {
var doc = myXMControl.Document;
if (doc) {
doc.Validate();
var vel = doc.ValidationErrorList;
```



```
if (vel) {

if (vel.Count == 0) { // valid document
assert(doc.IsValid);
ActiveDocument.Host.Alert("The document is valid.");

} else { // one or more validation errors
assert(!doc.IsValid);
var msg = "Errors:\n\nItem\tLevel\tNumber\tMessage\n";
var index = 1;
var ve = vel.item(index++);

while (ve) {
msg += index-1 + "\t" + ve.ErrorLevel
+ "\t" + ve.ErrorType
+ "\t" + ve.ErrorMessage;
ve = vel.item(index++);
msg += "\n";
}
gErrNum++;

if (gErrNum > vel.Count) {
gErrNum = 1; // go back to the first error
}
ve = vel.item(gErrNum);
msg += "\nPress OK to go to error " + gErrNum + " (Item " + gErrNum + ")";
ActiveDocument.Host.Alert(msg);

// There are 2 ways to go to the error:
//ValidationErrorList.GoToError(Index)
//ValidationError.Range.Select()
var doItWithGoToError = true;
if (doItWithGoToError) {
vel.GoToError(gErrNum);
} else {
ve.Range.Select();
}
}
}
}
}
</SCRIPT>
```

DOM interfaces

You can read and modify the structure of a document using the DOM interfaces. All DOM interfaces are available to both XMetaL Author and XMetaL XMAX.

XMetaL supports XML namespaces. In namespace aware APIs, elements and attributes are referred to by their local names, which are bound to a namespace URI. Do not use APIs that are namespace aware with APIs that are not namespace aware.

DOMAttr

A DOMAttr node corresponds to an attribute. It inherits the DOMNode interface.

To create a DOMAttr node, use the `Document.createAttribute` method. When an attribute is created, it is not associated with any element. This association is made using the setAttributeNode method.

DOMAttr nodes are not child nodes of an element node in the same way that nodes representing sub-elements are. Instead of saying that a DOMAttr is the child of a DOMElement node, we say that it is ‘associated with’ the DOMElement. A DOMAttr node does not appear in the document ‘tree’; it is obtained from a DOMElement object using its getAttributeNode method.

A single DOMAttr node can be associated with only one DOMElement node at a time. A DOMAttr node can be used only in the document that it was created in.

A DOMAttr node can have DOMText and DOMEntityReference nodes as child nodes; these correspond to text and entity references in the attribute value.

Properties

localName

Returns the local name of the attribute. This name is local to the corresponding namespaceURI. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMAttr_object.localName;
```

Usage in VBScript

```
DOMAttr_object.localName
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the first attribute of the current
// element that has a value

var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd) {
var ndAttrs = nd.attributes;
if (ndAttrs.length>0)
ActiveDocument.Host.Alert(ndAttrs.item(0).localName);
}
```

name
The attribute name. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMAttr_object.name;
```

Usage in VBScript

```
DOMAttr_object.name
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the first attribute of the current
// element that has a value

var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd) {
var ndAttrs = nd.attributes;
if (ndAttrs.length>0)
ActiveDocument.Host.Alert(ndAttrs.item(0).name);
}
```

namespaceURI
Returns the namespace URI of the attribute. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMAttr_object.namespaceURI;
```

Usage in VBScript

```
DOMAttr_object.namespaceURI
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the namespace of first attribute of the current
// element that has a value

var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd) {
var ndAttrs = nd.attributes;
if (ndAttrs.length>0)
ActiveDocument.Host.Alert(ndAttrs.item(0).namespaceURI);
}
```

ownerElement

Returns the DOMElement that the DOMAttr object associated with. Returns null if the DOMAttr has no owner.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMElement object

Usage in JScript

```
DOMAttr_object.ownerElement;
```

Usage in VBScript

```
DOMAttr_object.ownerElement
```


Example

```
// XMetaL Script Language JSCRIPT:

var Id = ActiveDocument.createAttribute("Id");
Id.value = "Xyzzy";
var rng = ActiveDocument.Range;
rng.GotoNext(sqElement);
var nd = rng.ContainerNode;

// Make sure the node exists and represents
// an element
if (nd && nd.nodeType=="1") {
var dt = ActiveDocument.doctype;

// Check whether Id is a valid attribute
if (dt.hasAttribute(nd.nodeName,"Id")) {
nd.setAttributeNode(Id);
}
IdOwner = Id.ownerElement;
if (IdOwner)
// Should be same as nd.nodeName
ActiveDocument.Host.Alert(IdOwner.nodeName);
else
ActiveDocument.Host.Alert("Id has no owner.");
}
```

prefix
Returns or sets the prefix of the attribute. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
DOMAttr_object.prefix;
```

Usage in VBScript

```
DOMAttr_object.prefix
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the prefix of first attribute of the current
// element that has a value

var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd) {
var ndAttrs = nd.attributes;
if (ndAttrs.length>0)
```

```
ActiveDocument.Host.Alert(ndAttrs.item(0).prefix);
}
```

specified
If the attribute node is associated with an element, this property returns False if the attribute exists simply because it was declared in the DTD or schema with a default value. Otherwise, it returns True. If the attribute node has not been associated with an element (ownerElement is null), it always returns True.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
DOMAttr_object.specified;
```

Usage in VBScript

```
DOMAttr_object.specified
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste after a Para in a "Journalist" DTD document

var rng = ActiveDocument.Range;
if (rng.CanInsert("ItemizedList")) {
rng.InsertElement("ItemizedList");
var nd = rng.ContainerNode;
var styleNode = nd.getAttributeNode("Style");
ActiveDocument.Host.Alert(styleNode.specified);
}
```

value
Gets or sets the value of the attribute represented by the node.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
vbl = DOMAttr_object.value;
DOMAttr_object.value = val;
```

Usage in VBScript

```
vbl = DOMAttr_object.value
DOMAttr_object.value = val
```

Example

```
// XMetaL Script Language JSCRIPT:
// Get a DOMAttr node's value

var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd && nd.nodeType=="1") {
var IdAttr = nd.getAttributeNode("Id");
if (IdAttr)
ActiveDocument.Host.Alert(IdAttr.value);
}

// XMetaL Script Language JSCRIPT:
// Create a DOMAttr node and set its value
var newAttr = ActiveDocument.createAttribute("Id");
newAttr.value = "xyzzy";
```

DOMCDATASection

Represents a CDATA section in an XMLor a marked section in an SGML document.

CDATA sections in a XML document are used to surround text to prevent it from being interpreted as XML markup. This enables you to use samples of XML and SGML markup in your documents without having to convert the markup delimiters to entity references.

Marked sections are supported only in SGML documents. They are similar to CDATA sections used in XML documents, but give you more options for indicating how the text they surround should be processed. The most common use of marked sections is to surround parts of the document so that they are sometimes processed by an application that is reading the document, and sometimes ignored.

To create a node representing a CDATA section, use `Document.createCDATASection`. To insert a CDATA section directly, use `Selection.InsertCDATASection`.

DOMCDATASection inherits the DOMCharacterData interface through the DOMText interface. It has no other properties or methods.

A DOMCDATASection object has no child nodes. To obtain the content represented by the object, use the `data` property.

DOMCharacterData

DOMCharacterData is a generic interface with a set of properties and methods for accessing character data. No part of a document structure corresponds to DOMCharacterData directly, but several other interfaces

inherit from it: DOMText, DOMCDATASection, and DOMComment. DOMCharacterData inherits the DOMNode interface.

Properties

data

Sets or gets the character data of the node.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
vbl = DOMCharacterData_object.data;
DOMCharacterData_object.data = strText;
```

Usage in VBScript

```
vbl = DOMCharacterData_object.data
DOMCharacterData_object.data = strText
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in an element whose contents are the
// string "[Date]" (no quotes)
var rng = ActiveDocument.Range;
var d = new Date();
// getMonth() starts at zero!
var month = d.getMonth() + 1;
var today = d.getDate() + "/" + month + "/" + d.getFullYear();
var curNode = rng.ContainerNode;
if (curNode) {
var textNode = curNode.firstChild;
if (textNode && textNode.data=="[Date]") {
textNode.data = today;
}
}
```

length

The number of characters available through the data property and the substringData method.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
DOMCharacterData_object.length;
```

Usage in VBScript

```
DOMCharacterData_object.length
```

Example

```
// XMetaL Script Language JSCRIPT:
// Truncate a text node's data to the first 32 chars
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
if (curNode) {
var textNode = curNode.firstChild;
if (textNode && textNode.nodeType=="3" &&
textNode.length>32) {
textNode.data = textNode.substringData(0,31);
}
}
```

Methods

appendData

Appends string data to the end of the character data of the node.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMCharacterData_object.appendData("strData");
```

Usage in VBScript

```
DOMCharacterData_object.appendData "strData"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste into an element that contains only text
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
if (curNode) {
var textNode = curNode.firstChild;
if (textNode) {
// Check that the node is a text node
if (textNode.nodeType=="3")
```

```
textNode.appendData(" MondoBongo!");
}
}
```

deleteData
Remove a range of characters from the node.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMCharacterData_object.deleteData(longOffset, longCount);
```

`longOffset`, specifies the starting point and `longCount` specifies the number of characters. If the operation succeeds, data and length reflect the change. An error occurs if `longOffset` is greater than the string's length, or is negative, or if `longCount` is negative. Offsets start at 0 (zero); therefore the offset of the Nth character in the node's data is N-1, and the offset of the last character is length-1.

Usage in VBScript

```
DOMCharacterData_object.deleteData longOffset, longCount"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Truncate a node's data after 10 characters
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
if (curNode) {
var textNode = curNode.firstChild;
// Check that the node really is a text node
if (textNode && textNode.nodeType=="3") {
// Get the node data's length
var dataLen = textNode.data.length;
if (dataLen > 10) {
// Start deletion at offset 10 (the 11th
// character in the data)
textNode.deleteData(10,dataLen-10);
}
}
}
```

insertData
Inserts string data into the node.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMCharacterData_object.insertData(longOffset, "strData");
```

`longOffset` specifies the starting point. An error occurs if `longOffset` is greater than the string's length, or is negative. Offsets start at 0 (zero); therefore the offset of the Nth character in the node's data is N-1, and the offset of the last character is length-1.

Usage in VBScript

```
DOMCharacterData_object.insertData longOffset, "strData"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Insert a string in a node's data
// after 10 characters
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
if (curNode) {
var textNode = curNode.firstChild;
// Check that the node really is a text node
if (textNode && textNode.nodeType=="3") {
// Get the node data's length
dataLen = textNode.data.length;
if (dataLen >= 10) {
// Insert at offset 10 (after the 10th
// character in the data)
textNode.insertData(10,"--");
}
}
}
```

replaceData

Replaces the specified number of characters with string data.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMCharacterData_object.replaceData(longOffset, longCount, "strData");
```

`longCount` specifies the number of characters, starting at `longOffset`. If the sum of `longOffset` and `longCount` exceeds length, then all characters to the end of the data are replaced. An error occurs if `longOffset` is greater than the string's length, or is negative, or if `longCount` is negative. Offsets start at 0 (zero); therefore the offset of the Nth character in the node's data is N-1, and the offset of the last character is length-1.

Usage in VBScript

```
DOMCharacterData_object.replaceData longOffset, longCount, "strData"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Replace the first 5 characters in a node's data
// with a specified string

var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;

if (curNode) {
var textNode = curNode.firstChild;

// Check that the node really is a text node
if (textNode && textNode.nodeType=="3") {
// Get the node data's length
dataLen = textNode.data.length;

if (dataLen >= 5) {
// Replace starting at offset 0 (the first
// character in the data)
textNode.replaceData(0,5,"xxxxxxxxxx");
}
}
}
```

substringData

Returns a range of data from the node.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
DOMCharacterData_object.substringData(longOffset, longCount);
```

`longOffset` specifies the starting point, and `longCount` the number of characters. If the sum of `longOffset` and `longCount` exceeds the length, then all the characters to the end of the data are returned.

Usage in VBScript

```
DOMCharacterData_object.substringData(longOffset, longCount)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Truncate a text node's data to the first 32 chars

var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;

if (curNode) {
var textNode = curNode.firstChild;
```



```
if (textNode && textNode.nodeType=="3" &&
textNode.length>32) {
textNode.data = textNode.substringData(0,31);
}
}
```

DOMCharacterReference

The DOMCharacterReference interface represents character references, that is, characters that are identified by their location in the Unicode ISO/IEC 10646 character set. For example, the character reference `&#233;` represents the lowercase e-acute character. This interface is an XMetaL extension to the DOM interface.

DOMCharacterReference nodes are created using the `Document.createCharacterReference` method; character references can be specified using a decimal or hexadecimal value.

DOMCharacterReference inherits the DOMNode interface; it has no properties and methods of its own. DOMCharacterReference nodes cannot have children; therefore the useful properties are:

- nextSibling, ownerDocument, parentNode, previousSibling - DOM tree navigation properties.
- nodeName - returns '#character-reference'
- nodeType - returns '505'
- nodeValue - returns a string representing the decimal number of the character reference.

DOMComment

The DOMComment interface represents the content of a XML or SGML comment (all characters between the starting '<!--' and the ending '-->'). This interface inherits the DOMCharacterData interface; it has no other properties or methods.

A DOMComment object has no child nodes; to obtain the text of the comment, use the data property.

To create a node corresponding to a comment, use `Document.createComment`. To insert a comment directly, use `Selection.InsertComment`.

DOMDocument

The DOMDocument interface provides a way to work with a DOM document.

Properties

doctype

Returns a DOMDocumentType object, which contains information about the DTD or schema and document type declaration. Returns a null string if the document has no document type declaration.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMDocumentType object

Usage in JScript

```
Document_object.doctype;
```

Usage in VBScript

```
Document_object.doctype
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the document type name

ActiveDocument.Host.Alert(ActiveDocument.doctype.name);
```

documentElement

Provides access to the node corresponding to the root (top-level) element of the document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMElement object

Usage in JScript

```
Document_object.documentElement;
```

Usage in VBScript

```
Document_object.documentElement
```

Example

```
// XMetaL Script Language JSCRIPT:

var elem;
elem=ActiveDocument.documentElement;
ActiveDocument.Host.Alert(elem.nodeName);
```

implementation

The DOMImplementation object that handles this document.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMImplementation object

Usage in JScript

```
Document_object.implementation;
```

Usage in VBScript

```
Document_object.implementation
```

Example

```
// XMetaL Script Language JSCRIPT:

var curImpl, xmlFeature;
curImpl= ActiveDocument.implementation;
xmlFeature=curImpl.hasFeature("XML","1.0");
ActiveDocument.Host.Alert(xmlFeature);
xmlFeature=curImpl.hasFeature("XML","2.1");
ActiveDocument.Host.Alert(xmlFeature);
```

Methods

createAttribute

Creates a DOMAttr node corresponding to the specified attribute. To associate a DOMAttr node with an element, use `DOMElement.setAttributeNode`. Attributes can also be inserted with the `Selection.ContainerAttribute` and `Selection.ElementAttribute` methods. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMAttr object

Usage in JScript

```
Document_object.createAttribute("strName");
```

Usage in VBScript

```
Document_object.createAttribute("strName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create the node
IdNode = ActiveDocument.createAttribute("Id");
// Assign the attribute a value
IdNode.value = "xyzzy";
// Get the element node
elemNode = Selection.ContainerNode;
// Associate the attribute node with the element node
elemNode.setAttributeNode(IdNode);
```

createAttributeNS

Creates a DOMAttr node corresponding to the specified attribute that is associated with the specified namespace. To associate a DOMAttr node with an element, use `DOMElement.setAttributeNodeNS`. Attributes can also be inserted with the `Selection.ContainerAttributeNS` and `Selection.ElementAttributeNS` methods. You can use these methods when you want to provide a local name (instead of a qualified name) for an attribute. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMAttr object

Usage in JScript

```
DOMDocument_object.createAttributeNS("strNamespaceURI", "strQualifiedName");
```

`strQualifiedName` is in the form `prefix:localName`.

Usage in VBScript

```
DOMDocument_object.createAttributeNS("strNamespaceURI", "strQualifiedName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create the node
IdNode =
ActiveDocument.createAttributeNS("http://www.mydomainname.com/Journalist",
 "myns:Id");
// Assign the attribute a value
IdNode.value = "xyzzy";
// Get the element node
elemNode = Selection.ContainerNode;
// Associate the attribute node with the element node
elemNode.setAttributeNode(IdNode);
```

createCDATASection

Creates a DOMCDATASection node containing the specified string. CDATA sections can also be created with `Selection.InsertCDATASection`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMCDATASection object

Usage in JScript

```
Document_object.createCDATASection("strData");
```

Usage in VBScript

```
Document_object.createCDATASection("strData")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a CDATA section node and insert it
// after the current node
// Create the node
var cdSect = ActiveDocument.createCDATASection("<Enter>");
var rng = ActiveDocument.Range;
// Get the node after the current node
var nextNode = rng.ContainerNode.nextSibling;
// Insert the CDATA Section node before the "next" node
nextNode.parentNode.insertBefore(cdSect,nextNode);
```

createComment

Creates a node containing the specified string. Comments can also be inserted with the `Selection.InsertComment` method.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMComment object

Usage in JScript

```
Document_object.createComment("strText");
```

Usage in VBScript

```
Document_object.createComment("strText")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a comment node and insert it
// at the end of the current node
// Create the comment node
var comment = ActiveDocument.createComment("Correct?");
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
// Insert the comment node
curNode.appendChild(comment);
```

createDocumentFragment
Creates an empty DOMDocumentFragment object.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMDocumentFragment object

Usage in JScript

```
Document_object.createDocumentFragment();
```

Usage in VBScript

```
Document_object.createDocumentFragment
```

Example

```
// XMetaL Script Language JSCRIPT:
// Succeeds if pasted anywhere inside a Sect1
// element in a Journalist DTD document
// Create the document fragment
var docFrag = ActiveDocument.createDocumentFragment();
// Add Sect and Title nodes
var sect1Node = ActiveDocument.createElement("Sect1");
docFrag.appendChild(sect1Node);
var titleNode = ActiveDocument.createElement("Title");
sect1Node.appendChild(titleNode);
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;

// If the current node is not a Sect1, try to find one
if (curNode) {

while (curNode.nodeName!="Sect1" && curNode.NodeName!="Article") {
rng.SelectElement();
var curNode = rng.ContainerNode;
}
}

// Insert the document fragment before the Sect1 node
if (curNode.nodeName=="Sect1"){
curNode.parentNode.insertBefore(docFrag,curNode);
}
```

createElement
Creates a DOMElement object corresponding to a specified element. Elements can also be inserted with `Selection.InsertElement`, `Selection.InsertElementWithRequired`, and `Selection.InsertWithTemplate`. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMElement object

Usage in JScript

```
Document_object.createElement("strName");
```

Usage in VBScript

```
Document_object.createElement("strName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a Para element node and insert
// it at the end of the document
var paraElem = ActiveDocument.createElement("Para");
var docElem = ActiveDocument.documentElement;
docElem.appendChild(paraElem);
```

createElementNS
Creates a DOMElement object corresponding to a specified element in a specified namespace. Use this method only with global (top-level) elements. Elements can also be inserted with `Selection.InsertElementNS`, `Selection.InsertElementWithRequiredNS`, and `Selection.InsertWithTemplateNS`. You can use these methods if you want to provide a local name (instead of a qualified name) for an element, or if you do not want to create a global element. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMElement object

Usage in JScript

```
DOMDocument_object.createElementNS("strNamespaceNameURI", "strQualifiedName");
```

`strQualifiedName` is in the form `prefix:localName`.

Usage in VBScript

```
DOMDocument_object.createElementNS("strNamespaceNameURI","strQualifiedName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a Para element node and insert
// it at the end of the document
var paraElem =
ActiveDocument.createElementNS("http://www.mydomainname.com/Journalist",
 "myns:Para");
var docElem = ActiveDocument.documentElement;
docElem.appendChild(paraElem);
```

createEntityReference
Creates a DOMEntityReference object corresponding to a reference to the specified entity. Entity references can also be inserted with `Selection.InsertEntity`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMEntityReference object

Usage in JScript

```
Document_object.createEntityReference("strName");
```

Usage in VBScript

```
Document_object.createEntityReference("strName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create an entity reference node and insert it
// at the end of the current container
var newER = ActiveDocument.createEntityReference("Acirc");
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
curNode.appendChild(newER);
```

createProcessingInstruction

Creates a DOMProcessingInstruction node corresponding to a processing instruction with the specified target and data. Processing instructions can also be inserted with `Selection.InsertProcessingInstruction`.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMProcessingInstruction object

Usage in JScript

```
Document_object.createProcessingInstruction("strTarget", "strData");
```

Usage in VBScript

```
Document_object.createProcessingInstruction("strTarget", "strData")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a PI node and insert it at the end of the
// current container
var newPI = ActiveDocument.createProcessingInstruction(
"my_formatter", "endpage");
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
curNode.appendChild(newPI);
```


createTextNode

Creates a DOMText node containing the specified string.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMText object

Usage in JScript

```
Document_object.createTextNode("strText");
```

Usage in VBScript

```
Document_object.createTextNode("strText")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a new text node and insert it
// at the beginning of the current element
var strText = "Hello world.";
textNode = ActiveDocument.createTextNode(strText);
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
curNode.insertBefore(textNode,curNode.firstChild);
```

getElementsByTagName

Returns a DOMNodeList of DOMElement objects, representing all elements in the document with the specified element name. Use '*' to match all elements. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
Document_object.getElementsByTagName("strTagName");
```

Usage in VBScript

```
Document_object.getElementsByTagName("strTagName")
```

Example

```
// XMetaL Script Language JSCRIPT:

var elemList;
var allElemList;
```

```
// Get all "Title" elements
elemList=ActiveDocument.getElementsByTagName("Title");
ActiveDocument.Host.Alert(elemList.length);
// Get all elements in the document
allElemList=ActiveDocument.getElementsByTagName("*");
ActiveDocument.Host.Alert(allElemList.length);
```

getElementsByTagNameNS
Returns a DOMNodeList of DOMElement objects, representing all elements in the document with the specified element name in the specified namespace. Use '*' to match all elements. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
DOMDocument_object.getElementsByTagNameNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
DOMDocument_object.getElementsByTagNameNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:

var elemList;
var allElemList;
// Get all "Title" elements in the http://www.mydomainname.com/Journalist
 namespace
elemList=ActiveDocument.getElementsByTagNameNS("http://www.mydomainname.com/Journalist",
 "Title");
ActiveDocument.Host.Alert(elemList.length);
// Get all elements in the http://www.mydomainname.com/Journalist namespace
allElemList=ActiveDocument.getElementsByTagNameNS("http://www.mydomainname.com/Journalist",
 "*");
ActiveDocument.Host.Alert(allElemList.length);
```

getNodesByXPath
Returns a DOMNodeList of DOMElement objects, matching a specified expression.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
DOMDocument_object.getNodesByXPath(strXPathExpression);
```


The string `strXPathExpression` contains an expression in the `XPath` 1.0 language. The context node is the DOMDocument node. `getNodesByXPath(strXPathExpression)` always returns a `DOMNodeList`, which may be empty if no nodes match the expression.

Usage in VBScript

```
DOMDocument_object.getNodesByXPath(strXPathExpression)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Using the CamerasInFocus sample in XMetaL Author,
// copy and paste this script example to get a list of titles.
// The XPath string searches for a list of "Title" nodes that has
// a "*/Sect1" parent with the DOMDocument node as the context node.

var titles = ActiveDocument.getNodesByXPath("*/Sect1/Title");
var tableOfContents = "";
var currTitleNumber = 1;
for (var index = 0; index < titles.length; index++)
{
var title = titles.item(index);
tableOfContents += currTitleNumber + ". " + title.childNodes.item(0).data
 + "\n";
currTitleNumber++;
}
Application.Alert(tableOfContents);
```

DOMDocumentFragment

The DOMDocumentFragment interface provides a way to work with a document (DOM) structure without opening it in a document window.

This interface creates a document ‘tree’ that is not linked to the DOM structure of any open document. This is faster than working with a document opened in the normal way because no formatting or validation is done. Since the document tree created with a DOMDocumentFragment object exists only in memory, if it is to be saved it must eventually be added to a node in a displayed document and that document saved.

DOMDocumentFragment objects are created with the `Document.createDocumentFragment` method. This creates a DOMNode that functions as the root node for the document fragment. DOMDocumentFragment inherits the DOMNode interface. You can add child nodes to this root node using DOMNode methods. You can then use any methods that modify the DOM tree to modify these child nodes and their descendants. These include the DOMNode methods appendChild, insertBefore, removeChild, and replaceChild, and the DOMElement methods removeAttributeNode, setAttribute, and setAttributeNS.

Note: You can insert entity references in a document fragment, but their subtrees are not generated until the fragment is inserted in an open document.

When you have built the desired tree, you can use some of the DOMNode methods listed above to insert it into the DOM structure of the displayed document. When a DOMDocumentFragment is inserted into a Document the children of the DOMDocumentFragment node and not the node itself are inserted into the DOMNode.

Note: The Selection and Range objects cannot be used with a DOMDocumentFragment object.

Example

The following script creates a document fragment, adds some nodes to it, and then inserts it into the main document tree.

```
// XMetaL Script Language JSCRIPT:
// Succeeds if pasted anywhere inside a Sect1
// element in a Journalist DTD document
// Create the document fragment
var docFrag = ActiveDocument.createDocumentFragment();
// Add Sect and Title nodes
var sect1Node = ActiveDocument.createElement("Sect1");
docFrag.appendChild(sect1Node);
var titleNode = ActiveDocument.createElement("Title");
sect1Node.appendChild(titleNode);
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;

// If the current node is not a Sect1, try to find one
if (curNode) {
while (curNode.nodeName!="Sect1" &&
curNode.NodeName!="Article") {
rng.SelectElement();
var curNode = rng.ContainerNode;
}
}

// Insert the document fragment before the Sect1 node
if (curNode.nodeName=="Sect1"){
curNode.parentNode.insertBefore(docFrag,curNode);
}
```

DOMDocumentType

A DOMDocumentType object provides information from the document's DTD and document type declaration.

There is only one DOMDocumentType object for each document; this object is returned by the Document.doctype or the `Application.NewDocumentType` property. DOMDocumentType inherits the DOMNode interface.

You should not use the DOMDocumentType interface for documents using XML Schemas. XMetaL may behave erratically or return unpredictable results.

Properties

attributeDefaultType
Returns an integer indicating the type of default value of the specified attribute of the specified element, as declared in the DTD.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
DOMDocumentType_object.attributeDefaultType(strElemName, strAttrName);
```

The default value can be an explicit literal value or one of several keywords, as indicated below. The possible return values are:

- sqDTUNKNOWN (-1): unknown
- sqDTIMPLIED (0): #IMPLIED
- sqDTREQUIRED (1): #REQUIRED
- sqDTSPECIFIED (2): specified explicitly
- sqDTFIXED (3): #FIXED
- sqDTCONREF (4): #CONREF (SGML only)
- sqDTCURRENT (5): #CURRENT (SGML only)

Usage in VBScript

```
DOMDocumentType_object.attributeDefaultType(strElemName, strAttrName)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste into a "Journalist" DTD document
// Change an attribute of the current element,
// unless it is a #FIXED value
var dt = ActiveDocument.doctype;
var rng = ActiveDocument.Range;
var curElem = rng.ContainerName;

if (curElem!=".DOCUMENT") {
var attrDefType = dt.attributeDefaultType(curElem,"Id");

if (attrDefType!=sqDTUNKNOWN && attrDefType!=sqDTFIXED){
rng.ContainerAttribute("Id") = "Xyzzy";
}
}
```

attributeDefaultValue
If the default value of the specified attribute of the specified element, as declared in the DTD, is a specific value, returns that value. Otherwise, returns a null string. This property returns a value only if the default value, as indicated by DOMDocumentType.attributeDefaultType is 2 (specified) or 3 (#FIXED).

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMDocumentType_object.AttributeDefaultValue(strElemName, strAttrName);
```

Usage in VBScript

```
DOMDocumentType_object.AttributeDefaultValue(strElemName, strAttrName)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Change an attribute of the current element,
// unless it is a #FIXED value

var dt = ActiveDocument.doctype;
var rng = ActiveDocument.Range;
var curElem = rng.ContainerName;

if (curElem != ".DOCUMENT") {
var attrDefType = dt.attributeDefaultType(curElem,"Id");

if (attrDefType == sqDTUNKNOWN) {
ActiveDocument.Host.Alert("Unknown attribute.");
} else if (attrDefType != sqDTFIXED) {
var defVal = dt.attributeDefaultValue(curElem,"Id");
ActiveDocument.Host.Alert(curElem + " has a fixed value of " + defVal);
} else {
rng.ContainerAttribute("Id") = "Xyzzy";
}
}
```

attributeType

Returns an integer indicating the attribute type of the specified attribute of the specified element.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
DOMDocumentType_object.attributeType(strElemName, strAttrName);
```

The return value of `attributeType` indicates the kind of value that the attribute can have. The possible return values are:

- sqUNKNOWN (-1): unknown
- sqCDATA (0): CDATA (character data—ordinary text)
- sqID (1): ID (identifier)
- sqIDREF (2): IDREF (reference to an identifier)
- sqIDREFS (3): IDREFS (references to one or more identifiers)

- sqENTITY (4): ENTITY (an entity name)
- sqENTITIES (5): ENTITIES (one or more entity names)
- sqNMTOKEN (6): NMTOKEN (a name token)
- sqNMTOKENS (7): NMTOKENS (one or more name tokens)
- sqNOTATION (8): NOTATION (a notation name)
- sqNAMETOKENGROUP (9): name token group (one of a set of name tokens, specified in the DTD)
- sqNAME (10): NAME (a name; SGML only)
- sqNAMES (11): NAMES (one or more names; SGML only)
- sqNUMBER (12): NUMBER (a number; SGML only)
- sqNUMBERS (13): NUMBERS (one or more numbers; SGML only)
- sqNUTOKEN (14): NUTOKEN (a number token; SGML only)
- sqNUTOKENS (15): NUTOKENS (one or more number tokens; SGML only)

Usage in VBScript

```
DOMDocumentType_object.attributeType(strElemName, strAttrName)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Assign an attribute value depending on whether
// its type is CDATA or ENTITY
var imgElem = "FIG";
var imgAttr = "FILENAME";
var imgEntName = "FACE01";
var rng = ActiveDocument.Range;
rng.MoveToElement(imgElem);

if (rng.ContainerName == imgElem) {
var docType = ActiveDocument.doctype;
var entMap = docType.entities;
var imgEnt = entMap.getNamedItem(imgEntName);
var imgFileName = imgEnt.systemId;

if (docType.AttributeType(imgElem,imgAttr) == 1) {
rng.ContainerAttribute(imgAttr) = imgFileName;
} else if (docType.attributeType(imgElem,imgAttr) == 4) {
rng.ContainerAttribute(imgAttr) = imgEntName;
}
}
```

childElementType
Returns an item of the array of all child elements that are specified for the specified element type in the document's DTD.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMDocumentType_object.childElementType(strElemType, longIndex);
```

An item can be accessed by specifying its index `longIndex`. The array index starts at 0. An empty string is returned if the index is greater than the array length.

Usage in VBScript

```
DOMDocumentType_object.childElementType(strElemType, longIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:

var dt = ActiveDocument.doctype;
var elemType = "Para";
ActiveDocument.Host.Alert(dt.childElementType(elemType,0));
```

childElementTypes

Returns a Variant object representing an array of element names.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Variant

Usage in JScript

```
DOMDocumentType_object.childElementTypes(strElemType);
```

Usage in VBScript

```
DOMDocumentType_object.childElementTypes(strElemType)
```

Example

```
' XMetaL Script Language VBSCRIPT:
set dt = ActiveDocument.doctype
elemArray = dt.childElementTypes("Para")

For Each el in elemArray
MsgBox(el)
Next
```

elementAttribute

Provides access to an array containing the names of all the attributes that are declared in the document's DTD for the specified element, in the same order as they appear in the Attribute Inspector.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMDocumentType_object.elementAttribute(strElemName, longIndex);
```

An attribute name can be accessed by specifying its index `longIndex`. The array index starts at 0. The empty string is returned if the index is greater than the array length.

Usage in VBScript

```
DOMDocumentType_object.elementAttribute(strElemName, longIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:

var dt=ActiveDocument.doctype;
ActiveDocument.Host.Alert(dt.elementAttribute("Para",0));
```

elementAttributes

Returns a Variant object that represents an array of attribute names.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Variant

Usage in JScript

```
DOMDocumentType_object.elementAttributes(strElemName);
```

In VBScript, this Variant object behaves like an ordinary array; in JScript, use the `object.ubound()` method to get the get the highest index used in the array, and `object.getItem(i)` to access an item in the array.

Usage in VBScript

```
DOMDocumentType_object.elementAttributes(strElemName)
```

Example

```
' XMetaL Script Language VBScript:
' Display the names of all the attributes of TABLE
' Paste in a Journalist DTD document
typs = ActiveDocument.doctype.elementAttributes("TABLE")

For i = LBound(typs) To UBound(typs)
MsgBox typs(i)
Next
```

elementContentType
Returns an integer that indicates the content type of the specified element.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
DOMDocumentType_object.elementContentType("strElemName");
```

The possible return values are:

- sqCTUNKNOWN (-1): Unknown. This value is returned if the selection is inside a comment, processing instruction, CDATA or marked section, at root level, or anywhere in Plain Text view.
- sqCTMIXED (0): Mixed content. This element can contain both text and elements
- sqCTELEMENT (1): Element content. This element can contain only other elements (that is, it cannot contain text directly)
- sqCTEMPTY (2): EMPTY element. This element can have no content
- sqCTANY (3): ANY. This element can contain text and can have any element as an immediate subelement.
- sqCTCDATA (4): CDATA content. This element's text is interpreted as CDATA. Any markup characters in the element content is interpreted as text. (SGML only)
- sqCTRCDATA (5): RCDATA content. This element's text is interpreted as RCDATA. Any markup characters in the element content, with the exception of entity references, is interpreted as text. (SGML only)
- sqCTCONREF (6): CONREF element. The element has an attribute of type CONREF. If this attribute has an explicit value, the element behaves like an EMPTY element. (SGML only)

Usage in VBScript

```
DOMDocumentType_object.elementContentType("strElemName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// .... // If the current node represents an EMPTY
```



```
// element, no need to do "tag exit" processing. // Example assumes
existence of "do_tagEnter"
// and "do_tagExit" functions. var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
var tagName = curNode.nodeName;
var dt = ActiveDocument.doctype;
do_tagEnter(curNode);

if (dt.elementContentType(tagName) != 2) {
do_tagExit(curNode);
}
```

elementType
Provides access to an array containing the names of all the elements that are declared in the document's DTD, in alphabetical order.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMDocumentType_object.elementType(longIndex);
```

A name can be accessed by specifying its index `longIndex`. The array index starts at 0. The empty string is returned if the index is greater than the array length.

Usage in VBScript

```
DOMDocumentType_object.elementType(longIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:

var dt = ActiveDocument.doctype;
ActiveDocument.Host.Alert(dt.elementType(0));
```

elementTypes
Returns a Variant object that represents the array.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Variant

Usage in JScript

```
DOMDocumentType_object.elementTypes;
```

In VBScript, this Variant object behaves like an ordinary array; in JScript, use the `object.ubound()` method to get the get the highest index used in the array, and `object.getItem(i)` to access an item in the array.

Usage in VBScript

```
DOMDocumentType_object.elementTypes
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the number of elements in the DTD

typs = ActiveDocument.doctype.elementTypes;
ActiveDocument.Host.Alert(typs.ubound()+1);
```

entities

Returns a DOMNamedNodeMap of DOMEntity objects representing the general entities, both external and internal, declared in the DTD. Duplicates are discarded.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNamedNodeMap object

Usage in JScript

```
DOMDocumentType_object.entities;
```

Usage in VBScript

```
DOMDocumentType_object.entities
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the number of entities declared
// in the DTD

var nodemap = ActiveDocument.doctype.entities;
ActiveDocument.Host.Alert(nodemap.length);
```


enumeratedAttributeType
If the attribute type of the specified attribute of the specified element is ‘enumerated’, then enumeratedAttributeType provides access to an array containing all the possible values for specified attribute, as declared in the DTD, in alphabetical order.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMDocumentType_object.enumeratedAttributeType("strElemName", "strAttrName", longIndex);
```

A name can be accessed by specifying its index `longIndex`. The array index starts at 0. The empty string is returned if the index is greater than the array length.

Usage in VBScript

```
DOMDocumentType_object.enumeratedAttributeType("strElemName", "strAttrName", longIndex);
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display all the possible values for an attribute
// Paste into a "Journalist" DTD document

var docType = ActiveDocument.doctype;
var idx = 0;
var token = docType.enumeratedAttributeType("ItemizedList",
"Style", idx);

while (token != "") {
ActiveDocument.Host.Alert(token);
idx++;
token = docType.enumeratedAttributeType("ItemizedList",
"Style", idx);
}
```

enumeratedAttributeTypes
Returns a Variant object that represents the array.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Variant

Usage in JScript

```
varEnumAttrType = DOMDocumentType_object.enumeratedAttributeTypes("strElemName",
"strAttrName");
```

In VBScript, this Variant object behaves like an ordinary array. In JScript, use the `object.ubound()` method to get the highest index used in the array, and `object.getItem(i)` to access an item in the array.

Usage in VBScript

```
varEnumAttrType = DOMDocumentType_object.enumeratedAttributeTypes("strElemName",
"strAttrName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display all the possible values for an attribute
// Paste into a "Journalist" DTD document

var docType = ActiveDocument.doctype;
var idx;
var tokenString;
var tokenArray = docType.enumeratedAttributeTypes("ItemizedList",
"Style");
var numTokens = tokenArray.ubound()+1;

if (numTokens>0){
tokenString = tokenArray.getItem(0);
}

for (idx=1; idx<numTokens; idx++) {
tokenString = tokenString + ", ";
tokenString = tokenString + tokenArray.getItem(idx);
}
ActiveDocument.Host.Alert(tokenString);
```

hasAttribute

Returns True if the specified attribute has been declared for the specified element.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
DOMDocumentType_object.hasAttribute(strElemName, strAttrName);
```


Usage in VBScript

```
DOMDocumentType_object.hasAttribute(strElemName, strAttrName)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste into a "Journalist" DTD document

var dt, rootElem;
var dt = ActiveDocument.doctype;
var rootElem = ActiveDocument.documentElement.nodeName;

if (dt.hasAttribute(rootElem,"Id")){
ActiveDocument.Host.Alert("Document has Id attribute.");
}
```

hasChildElementType
Returns a boolean indicating whether the specified element has the specified child element.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
DOMDocumentType_object.hasChildElementType(strElemName, strChildElemName);
```

Usage in VBScript

```
DOMDocumentType_object.hasChildElementType(strElemName, strChildElemName)
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set dt = ActiveDocument.doctype
MsgBox(dt.hasChildElementType("Para", "Subscript"))
```

hasElementType
Returns True if the specified element has been declared in the DTD.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
DOMDocumentType_object.hasElementType(strElemName);
```

Usage in VBScript

```
DOMDocumentType_object.hasElementType(strElemName)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Check document's table model
// Note: Check uppercase only for brevity

var dt = ActiveDocument.doctype;
if (dt.hasElementType("TGROUP")){
ActiveDocument.Host.Alert("Document uses CALS tables.");
} else if (dt.hasElementType("TD") &&
dt.hasElementType("TH")){
ActiveDocument.Host.Alert("Document uses HTML tables.");
} else {
ActiveDocument.Host.Alert("Document does not use tables.");
}
```

hasParentElementType

Returns a boolean indicating whether the specified element has the specified parent.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
DOMDocumentType_object.hasParentElementType(strElemName, strParentElemName);
```

Usage in VBScript

```
DOMDocumentType_object.hasParentElementType(strElemName, strParentElemName)
```

Example

```
' XMetaL Script Language VBSCRIPT:
Set dt = ActiveDocument.doctype
MsgBox(dt.hasParentElementType("Para", "Sect1"))
```


name
The document type; that is, the name immediately following the DOCTYPE keyword.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMDocumentType_object.name;
```

Usage in VBScript

```
DOMDocumentType_object.name
```

Example

```
// XMetaL Script Language JSCRIPT:

ActiveDocument.Host.Alert(ActiveDocument.doctype.name);
```

notations
A DOMNamedNodeMap of DOMNotation objects representing the notations declared in the document's DTD. Duplicates are discarded.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNamedNodeMap object

Usage in JScript

```
DOMDocumentType_object.notations;
```

Usage in VBScript

```
DOMDocumentType_object.notations
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the number of notations declared in the DTD

var nodemap = ActiveDocument.doctype.notations;
ActiveDocument.Host.Alert(nodemap.length);
```

parentElementType
Returns an item of the array of all parent elements that are specified for the specified element type in the document's DTD.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Description

Usage in JScript

```
DOMDocumentType_object.parentElementType(strElemName, longIndex);
```

A name can be accessed by specifying its index `longIndex`. The array index starts at 0. An empty string is returned if the index is greater than the array length.

Usage in VBScript

```
DOMDocumentType_object.parentElementType(strElemName, longIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:

var dt = ActiveDocument.doctype;
var elemType = Selection.ContainerName;
ActiveDocument.Host.Alert(dt.parentElementType(elemType, 0));
```

parentElementTypes
Returns the array of all parent elements that are specified for the specified element type in the document's DTD.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Array

Usage in JScript

```
DOMDocumentType_object.parentElementTypes(strElemName);
```

Usage in VBScript

```
DOMDocumentType_object.parentElementTypes(strElemName)
```

Example

```
' XMetaL Script Language VBSCRIPT:
set dt = ActiveDocument.doctype
elemArray = dt.parentElementTypes("Para")

For Each el in elemArray
MsgBox(el)
Next
```

publicId

Returns the public identifier in the document type declaration. Public identifiers are optional.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMDocumentType_object.publicId;
```

Usage in VBScript

```
DOMDocumentType_object.publicId
```

Example

```
// XMetaL Script Language JSCRIPT:

var docType = ActiveDocument.doctype;
if (docType.publicId) {
ActiveDocument.Host.Alert(docType.publicId);
} else {
ActiveDocument.Host.Alert("No public identifier.");
}
```

systemId
Returns the system identifier in the document type declaration. System identifiers are mandatory.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMDocumentType_object.systemId;
```

Usage in VBScript

```
DOMDocumentType_object.systemId
```

Example

```
// XMetaL Script Language JSCRIPT:

var docType = ActiveDocument.doctype;
if (docType.systemId) {
ActiveDocument.Host.Alert(docType.systemId);
} else {
ActiveDocument.Host.Alert("No system identifier.");
}
```

Methods

addAttribute
Creates the specified new attribute on the specified element available to all documents that use the document's DTD. This method is typically used in the On_DTD_Open_Complete event macro. This addition is not permanent; it persists only for the current XMetaL editing session. To add attributes that have an enumerated set of possible values, use addEnumeratedAttribute.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMDocumentType_object.addAttribute(strElemName, strAttrName, [strHelp], [intAttrType],
[intDeclType], [strDefault=]);
```


`strHelp` specifies a help string for the Attribute Inspector.

`intAttrType` specifies the attribute type (as defined by `DOMDocumentType.attributeType`); the default is CDATA (0).

`intDeclType` specifies the attribute default type (as defined by `DOMDocumentType.attributeDefaultType`); the default is #IMPLIED (0).

`strDefault` specifies the attribute's default value (as returned by `DOMDocumentType.attributeDefaultValue`).

Usage in VBScript

```
DOMDocumentType_object.addAttribute strElemName, strAttrName, [strHelp], [intAttrType], [intDeclType], [strDefault]
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste into a "Journalist" DTD document
var docType = ActiveDocument.doctype;
docType.addAttribute("Para","Xref","Cross-reference",2,0);
```

addElement

Makes the specified new element available to all documents that use the document's DTD. This method is typically used in the On_DTD_Open_Complete event macro. This addition is not permanent; it persists only for the current XMetaL editing session. After you have declared this element, you need to use addElementToInclusions to add it to an element's content model.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMDocumentType_object.addElement(strElemName,[strHelp], [boolAny], [boolRetainSpace]);
```

`strHelp` specifies a help string for the Element List.

If `boolAny` is True, the element has a content model of ANY; otherwise, it has a content model of EMPTY (these are the only two content models you can specify). If `boolRetainSpace` is `true`, 'pretty-printing', as specified in the Text Layout tab of the XMetaL Developer Customizations dialog box, is applied to this element; otherwise, whitespace inside this element is not changed.

Usage in VBScript

```
DOMDocumentType_object.addElement strElemName, [strHelp], [boolAny], [boolRetainSpace]
```

Example

```
// XMetaL Script Language JSCRIPT:
var docType = ActiveDocument.doctype;
docType.addElement("TechNote","Technical Note",true);
```

addElementToInclusions
You can use this method to specify that the specified element can be inserted anywhere in the specified target element. (This functionality is similar to the SGML inclusion mechanism.) Once you have specified ths temporary rule, all documents using this document's DTD are affected. This method can be considered a temporary override of a document's DTD-defined rules. This method is typically used in the On_DTD_Open_Complete event macro. This override is not permanent; it persists only until XMetaL is closed.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMDocumentType_object.addElementToInclusions ("strIncludedElem", "strTargetElem");
```

`strIncludedElem` and `strTargetElem` must be already declared in the DTD via `addElement`.

Usage in VBScript

```
DOMDocumentType_object.addElementToInclusions "strIncludedElem","strTargetElem"
```

Example

```
// XMetaL Script Language JSCRIPT:

var docType = ActiveDocument.doctype;
docType.addElementToInclusions("Graphic","Sect1");
```

addEnumeratedAttribute
Makes the specified new attribute of the specified element, which has an enumerated group of tokens as its possible values, available to all documents that use the document's DTD. This method is typically used in the On_DTD_Open_Complete event macro. This method is used to add attributes whose attributeType is 8 (NOTATION) or 9 (NAMETOKENGROUP); use addAttribute to add other types of attributes. This addition is not permanent; it persists only for the current XMetaL editing session.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMDocumentType_object.addEnumeratedAttribute(strElemName, strAttrName, strHelp,
intAttrType, intDeclType, strDefault, strEnum1, [strEnum2], [strEnum3], ...);
```

`strHelp` specifies a help string for the Attribute Inspector.

`intAttrType` specifies the attribute type (as defined by DOMDocumentType.attributeType).

`intAttrType` can have two values: "9" to create an ordinary enumeration, and "8" to create a NOTATION attribute.

`intDeclType` specifies the attribute default type (as defined by DOMDocumentType.attributeDefaultType).

If `intDeclType` is not equal to 2 (explicitly specified default value), then the value specified by **strDefault** is ignored. This enables you do create an attribute with default value #IMPLIED or #REQUIRED, for example.

`strDefault` specifies the attribute's default value (as returned by DOMDocumentType.attributeDefaultValue).

The attribute's possible values are specified by `strEnum1`, `strEnum2`, ...; you can provide as many as are necessary.

Usage in VBScript

```
DOMDocumentType_object.addEnumeratedAttribute strElemName, strAttrName, strHelp,
intAttrType, intDeclType, strDefault, strEnum1, [strEnum2], [strEnum3], ...);
```

Example

```
// XMetaL Script Language JSCRIPT:
// Equivalent to:
// <!ATTLIST Para Status (N|CL|TS|UN) "N" >
var docType = ActiveDocument.doctype;
docType.addEnumeratedAttribute("Para","Status","Security level",9,2,
"N","N","CL","TS","UN");
```

addInPlaceControlOverride
Dynamically adds an in-place ActiveX control, with the specified ProgId, to the list of controls that can represent the specified element in the Normal and Tags On views. This does the same as specifying a control in the customization editor's Treat As tab, but you can use this method to specify multiple controls for the same element. This method should be called from the On_Document_Open_Complete event macro.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
addInPlaceControlOverride(strElemName, strProgId, strPrefix , [strData] ,
[boolBmpPrinting]);
```

`strPrefix` specifies the prefix for the special macros that communicate between XMetaL and the control.

`strData` specifies a string that the special macros can use. This string can be read with the `InPlaceControl.UserData` property.

If `strBmpPrinting` is True, XMetaL uses a special printing mechanism when printing in-place controls.

Usage in VBScript

```
addInPlaceControlOverride(strElemName, strProgId, strPrefix , [strData] , [boolBmpPrinting])
```

Example

```
// XMetaL Script Language JSCRIPT:
var docType = ActiveDocument.doctype;
docType.addInPlaceControlOverride("Author",
"Shell.Explorer", "AuthorForm");
```

addTableElementMap
Declares a collection of elements to treat as a table and provide table editing semantics in Tags On or Normal view modes. This method is typically used in the On_DTD_Open_Complete event macro. This addition is not permanent; it persists only for the current XMetaL editing session.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMDocumentType_object.addTableElementMap(strTableType, strJSON);
```

`strTableType` specifies either "CALS", "HTML" or "Custom".

`strJSON` is an object that provides mapping from element types to table semantics such as a title, row, column, etc.

Example "CALS"

```
// XMetaL Script Language JSCRIPT:
var json =
"{\
 \"table\":{ \"elem\":\"calstbl2\", \"alias\":\"CALS Table 2\" }, \
 \"tgroup\":\"tgroup2\", \
 \"tbody\":\"tbody2\", \
 \"colspec\":\"colspec2\", \
 \"entry\":\"entry2\", \
 \"entrytbl\":\"entrytbl2\", \
 \"row\":\"row2\", \
 \"spanspec\":\"spanspec2\", \
 \"thead\":\"thead2\", \
 \"tfoot\":\"tfoot2\" \
 }"

var docType = ActiveDocument.doctype;
docType.addTableElementMap("CALS", json);
```


Examples "HTML"

```
// XMetaL Script Language JSCRIPT;
var json =
"{\
 \"table\":{ \"elem\":\"TABLE2\", \"alias\":\"HTML Table 2\" }, \
 \"tbody\":\"TBODY2\", \
    \"caption\":\"CAPTION2\", \
    \"col\":\"COL2\", \
    \"colgroup\":\"COLGROUP2\", \
    \"td\":\"TD2\", \
    \"tfoot\":\"TFOOT2\", \
    \"th\":\"TH2\", \
    \"thead\":\"THEAD2\", \
    \"tr\":\"TR2\" \
 }"

var docType = ActiveDocument.doctype;
docType.addTableElementMap("HTML", json);
```

Example "Custom"

```
var json =
"{\
 \"table\":{ \"elem\":\"simpletable\" }, \
    \"headRow\":\"sthead\", \
    \"row\":\"strow\", \
    \"cell\":\"stentry\" \
 }"

var docType = ActiveDocument.doctype;
docType.addTableElementMap("Custom", json);


\var json =
"{\
 \"table\":{ \"elem\":\"reltable\" }, \
    \"row\":\"relrow\", \
    \"cell\":\"relcell\" \
    \"caption\":\"title\" \
 }"

var docType = ActiveDocument.doctype;
docType.addTableElementMap("Custom", json);

var json =
"{\
 \"minCols\":2, \
 \"maxCols\":2, \
 \"table\":{ \"elem\":\"lcMatchTable\" }, \
 \"headRow\":\"lcMatchingHeader\", \
    \"row\":\"lcMatchingPair\", \
    \"cells\":[\"lcItem\", \"lcMatchingItem\"] \
 }"

var docType = ActiveDocument.doctype;
docType.addTableElementMap("Custom", json);
```

DOMElement

This interface represents an element in a document. It inherits the DOMNode interface.

A DOMElement node can have the following child nodes, in any number and order:

- DOMElement
- DOMText
- DOMComment
- DOMProcessingInstruction
- DOMCDATASection
- DOMEntityReference

These objects correspond to the XML constructs that you typically find inside an element: sub-elements, text, comments, processing instructions, sections, and entity references.

Consider the following example:

```
<Title>Sales Budget, <emph>Q4 2003<emph> &company; <Title>
```

The DOMElement object for the <Title> element contains the following child nodes:

- A DOMText node representing ‘Sales Budget,’
- A DOMElement node representing the <emph> element. This node contains:
  - A DOMText node representing ‘Q4 2003’
- A DOMEntityReference node representing ‘&company;’. This node contains:
  - A DOMText node representing the entity’s replacement text.

Properties

localName

Returns the local name of the element. This name is local to the namespace specified by namespace URI. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMElement_object.localName;
```


Usage in VBScript

```
DOMElement_object.localName
```

Example

```
// XMetaL Script Language JSCRIPT:

var docElem = ActiveDocument.documentElement;
if (docElem) {
var lname = docElem.localName;
ActiveDocument.Host.Alert(lname);
}
```

namespaceURI
Returns the URI of the namespace of the element. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMElement_object.namespaceURI;
```

Usage in VBScript

```
DOMElement_object.namespaceURI
```

Example

```
// XMetaL Script Language JSCRIPT:

var docElem = ActiveDocument.documentElement;
if (docElem) {
var namespace = docElem.namespaceURI;
ActiveDocument.Host.Alert(namespace);
}
```

prefix
Returns the prefix of the element. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMElement_object.prefix;
```

Usage in VBScript

```
DOMElement_object.prefix
```

Example

```
// XMetaL Script Language JSCRIPT:

var docElem = ActiveDocument.documentElement;
if (docElem) {
var prefix = docElem.prefix;
ActiveDocument.Host.Alert(prefix);
}
```

querySelector
This property is reserved for internal use.

querySelectorAll
This property is reserved for internal use.

tagName
The name of the element represented by this node. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMElement_object.tagName;
```

Usage in VBScript

```
DOMElement_object.tagName
```

Example

```
// XMetaL Script Language JSCRIPT:

var docElem = ActiveDocument.documentElement;
if (docElem) {
```



```
var tname = docElem.tagName;
ActiveDocument.Host.Alert(tname);
}
```

xml

Returns the XML information set for a given DOMElement, including default or fixed attribute values.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
Document_object.documentElement.xml;
```

Usage in VBScript

```
Document_object.documentElement.xml
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert(ActiveDocument.documentElement.xml);
```

Related Links

W3C XML InfoSet

Methods

getAttribute

Returns the value of the element's specified attribute. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
DOMElement_object.getAttribute("strAttrName");
```

`getAttribute` returns the empty string if the attribute is unset. Since this is indistinguishable from an empty string that is really the attribute value, you should use `DOMElement.hasAttribute` if you need to test whether the attribute is set.

Usage in VBScript

```
DOMElement_object.getAttribute("strAttrName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in "Journalist" DTD document

var tag=ActiveDocument.documentElement;
if (tag) {
ActiveDocument.Host.Alert(tag.getAttribute("Id"));
}
```

getAttributeNode

Returns a DOMAttr node corresponding to the specified attribute. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMAttr object

Usage in JScript

```
DOMElement_object.getAttributeNode("strAttrName");
```

Usage in VBScript

```
DOMElement_object.getAttributeNode("strAttrName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in "Journalist" DTD document

var tag=ActiveDocument.documentElement;
if (tag) {
var attrNode = tag.getAttributeNode("Id");
if (attrNode) {
ActiveDocument.Host.Alert(attrNode.nodeValue);
}
}
```

getAttributeNodeNS

Returns a DOMAttr node corresponding to the specified attribute. The attribute is local to the specified namespace. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMAttr object

Usage in JScript

```
DOMElement_object.getAttributeNodeNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
DOMElement_object.getAttributeNodeNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in "Journalist" schema document

var tag=ActiveDocument.documentElement;
if (tag) {
var attrNode =
tag.getAttributeNodeNS("http://www.mydomainname.com/Journalist", "Id");
if (attrNode) {
ActiveDocument.Host.Alert(attrNode.nodeValue);
}
}
```

getAttributeNS

Returns the value of the element's specified attribute. The attribute is local to the specified namespace. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

String

Usage in JScript

```
DOMElement_object.getAttributeNS("strNamespaceNameURI", "strLocalName");
```

`getAttributeNS` returns the empty string if the attribute is unset. Since this is indistinguishable from an empty string that is really the attribute value, you should use `DOMElement.hasAttributeNS` if you need to test whether the attribute is set.

Usage in VBScript

```
DOMElement_object.getAttributeNS("strNamespaceNameURI", "strLocalName");
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in "Journalist" schema document

var tag=ActiveDocument.documentElement;
if (tag) {
ActiveDocument.Host.Alert(tag.getAttributeNS("http://www.mydomainname.com/Journalist",
```

```
 "Id"));
}
```

getElementsByTagName
Returns a DOMNodeList of DOMElement objects, representing all the specified sub-elements contained in this element. The special value '*' matches all elements. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
DOMElement_object.getElementsByTagName("strTagName");
```

Usage in VBScript

```
DOMElement_object.getElementsByTagName("strTagName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the number of elements in the document
// (add 1 for top-level element)

var tag=ActiveDocument.documentElement;
if (tag) {
var nodelist = tag.getElementsByTagName("*");
ActiveDocument.Host.Alert(nodelist.length + 1);
}
else
ActiveDocument.Host.Alert("0");
```

getElementsByTagNameEx
Returns either a live, snapshot or forward-iterator-only DOMNodeList containing all the descendant elements of this element that have the name 'tagname'. The special tagname '*' matches all elements.

The "mode" parameter affects the DOMNodeList behavior as follows below:

- mode == 3 returns a "snapshot" list; document mutations will not cause list to change though nodes may become invalid.
- mode == 4 returns a "live" list; document mutations can cause list to change.

mode == 5 returns a "forward-iterator-only" list; special case of "snapshot" list. DOMNodeList::item(0) method gets first in-sequence item and moves iterator to the head of the list, whereby calling DOMNodeList::item() method gets next in-sequence item regardless of index parameter value (must be different from 0) provided.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
DOMElement_object.getElementsByTagNameEx(tagname);
```

Usage in VBScript

```
DOMElement_object.getElementsByTagNameEx(tagname)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Using the CamerasInFocus sample in XMetaL Author,
// copy and paste this script example to get a list of titles.
// The XPath string searches for a list of all "Title" nodes.
// "mode" 2 parameter forces to  return "forward-iterator-only" "titles" list;
// calling DOMNodeList::item(0) method moves "forward-iterator-only" to the first item.
// calling DOMNodeList::item(1) method gets the next item in sequence .

var tableOfContents = "";
var currTitleNumber = 1;

var titles = ActiveDocument.getElementsByTagNameEx("Title",5);
// Select first element from the titles list
var title = titles.item(0);
while(title){
            tableOfContents += currTitleNumber + ". " + title.childNodes.item(0).data +
"\n";
            currTitleNumber++;
            title = titles.item(1); // Select first element from the titles list
}
Application.Alert(tableOfContents);
```

getElementsByTagNameNS

Returns a DOMNodeList of DOMElement objects, representing all the specified sub-elements contained in this element. The element is local to the specified namespace. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
DOMElement_object.getElementsByTagNameNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
DOMElement_object.getElementsByTagNameNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Display the number of elements belonging to the
// namespace named http://www.mydomainname.com in the document
// (add 1 for top-level element)

var tag=ActiveDocument.documentElement;
if (tag) {
var nodelist = tag.getElementsByTagNameNS("http://www.mydomainname.com",
 "*");
ActiveDocument.Host.Alert(nodelist.length + 1);
}
else
ActiveDocument.Host.Alert("0");
```

getElementsByTagNameNSEx
Namespace-aware version of the getElementsByTagNameEx API method.

See *getElementsByTagNameEx* on page 529 for details.

getNodesByXPath
Returns a DOMNodeList of DOMElement objects, matching the specified expression.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
DOMElement_object.getNodesByXPath(strXPathExpression);
```

The string `strXPathExpression` contains an expression in the `XPath` 1.0 language. The context node is the DOMDocument node. The DOMNodeList may be empty if no nodes match the expression.

Usage in VBScript

```
DOMElement_object.getNodesByXPath(strXPathExpression)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Using the CamerasInFocus sample in XMetaL Author,
// copy and paste this script example to get a list of titles
// The XPath string searches for a list of "Title" nodes that has a
// "Sect1" parent with the root element node as the context node.

var objNode = ActiveDocument.documentElement;
var titles = objNode.getNodesByXPath("./Sect1/Title");
var tableOfContents = "";
var currTitleNumber = 1;
for (var index = 0; index < titles.length; index++)
{
var title = titles.item(index);
tableOfContents += currTitleNumber + ". " + title.childNodes.item(0).data
 + "\n";
currTitleNumber++;
```



```
}
Application.Alert(tableOfContents);
```

getNodesByXPathEx
Returns either a live, snapshot or forward-iterator-only DOMNodeList containing all the descendant nodes of this element that match up with the XPath expression provided.

The "mode" parameter affects the DOMNodeList behavior as follows below:

- mode == 3 returns a "snapshot" list; document mutations will not cause list to change though nodes may become invalid.
- mode == 4 returns a "live" list; document mutations can cause list to change.

mode == 5 returns a "forward-iterator-only" list; special case of "snapshot" list. DOMNodeList::item(0) method gets first in-sequence item and moves iterator to the head of the list, whereby calling DOMNodeList::item() method gets next in-sequence item regardless of index parameter value (must be different from 0) provided.

Applies To

XMetaL XMAX and XMetaL Author

Returns

DOMNodeList object

Usage in JScript

```
DOMElement_object.getNodesByXPathEx(strXPathExpression);
```

Usage in VBScript

```
DOMElement_object.getNodesByXPathEx(strXPathExpression)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Using the CamerasInFocus sample in XMetaL Author,
// copy and paste this script example to get a list of titles.
// The XPath string searches for a list of "Title" nodes that has
// a "*/Sect1" parent with the DOMDocument node as the context node.
// "mode" 2 parameter forces to  return "forward-iterator-only" "titles" list;
// calling DOMNodeList::item(0) method moves "forward-iterator-only" to the first item.
// calling DOMNodeList::item(1) method gets the next item in sequence .

var tableOfContents = "";
var currTitleNumber = 1;

var titles = ActiveDocument.getNodesByXPathEx("*/Sect1/Title",5);
// Select first element from the titles list
var title = titles.item(0);
while(title){
            tableOfContents += currTitleNumber + ". " + title.childNodes.item(0).data +
"\n";
            currTitleNumber++;
            title = titles.item(1); // Select first element from the titles list
}
Application.Alert(tableOfContents);
```

hasAttribute
Returns True if a value (which could be the empty string) has been set for the specified attribute. This method is preferred over simply testing the attribute value, because the empty string is indistinguishable from an unset attribute value. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
DOMElement_object.hasAttribute("strAttrName");
```

Usage in VBScript

```
DOMElement_object.hasAttribute("strAttrName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in "Journalist" DTD document
var docElem = ActiveDocument.documentElement;
if (docElem && !docElem.hasAttribute("Id")) {
docElem.setAttribute("Id","xyzzy");
}
```

hasAttributeNS
Returns True if a value (which could be the empty string) has been set for the specified attribute. The attribute is local to the specified namespace. This method is preferred over simply testing the attribute value, because the empty string is indistinguishable from an unset attribute value. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
DOMElement_object.hasAttributeNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
DOMElement_object.hasAttributeNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in "Journalist" schema document
```



```
var docElem = ActiveDocument.documentElement;
if (docElem &&
!docElem.hasAttributeNS("http://www.mydomainname.com/Journalist", "Id"))
 {
docElem.setAttributeNS("http://www.mydomainname.com/Journalist", "Id",
"xyzzy");
}
```

normalize
Joins adjacent DOMText nodes in the DOMElement tree structure into a single node. In this ‘normal’ form only markup (elements, comments, processing instructions, sections, and entity references) separates DOMText nodes. After normalization, a sequence of adjacent text nodes is represented by a single node. Referencing a pointer to one of the other nodes in the sequence could lead to an error and should be avoided.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMElement_object.normalize();
```

Usage in VBScript

```
DOMElement_object.normalize
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in an *empty* Para element in a "Journalist"
// DTD document
// Insert three text nodes, then normalize them

var textNode1 = ActiveDocument.CreateTextNode("Hello world. ");
var textNode2 = ActiveDocument.CreateTextNode("Have a nice day. ");
var textNode3 = ActiveDocument.CreateTextNode("Good-bye.");
var curNode = Selection.ContainerNode;
if (curNode && curNode.nodeName=="Para" &&
curNode.firstChild==null) {
curNode.appendChild(textNode1);
curNode.appendChild(textNode2);
curNode.appendChild(textNode3);
// Displays first node's original content
ActiveDocument.Host.Alert(textNode1.data);
curNode.normalize();
// First node now contains the joined content
// from all three original nodes
ActiveDocument.Host.Alert(textNode1.data);
}
```

removeAttribute
Removes the specified attribute from the element. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMElement_object.removeAttribute("strAttrName");
```

Usage in VBScript

```
DOMElement_object.removeAttribute "strAttrName"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in Para element with Id set to some value,
// in a Journalist DTD document
// Remove "Id" attribute
var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd && nd.nodeName=="Para") {
nd.removeAttribute("Id");
}
```

removeAttributeNode
Removes a DOMAttr node from the element. This DOMAttr node is returned.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMAttr object

Usage in JScript

```
DOMElement_object.removeAttributeNode(nodeDOMAttr);
```

Usage in VBScript

```
DOMElement_object.removeAttributeNode(nodeDOMAttr)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in a Para element with Id set to some value,
// in a Journalist DTD document
// Remove "Id" attribute node
var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd && nd.nodeName=="Para") {
var ndAttrs = nd.attributes;
var IdAttr = ndAttrs.getNamedItem("Id");
```



```
if (IdAttr)
nd.removeAttributeNode(IdAttr);
}
```

removeAttributeNS
Removes the specified attribute from the element. The attribute is local to the specified namespace. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMElement_object.removeAttributeNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
DOMElement_object.removeAttributeNS "strNamespaceNameURI", "strLocalName"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in Para element with Id set to some value,
// in a Journalist schema document
// Remove "Id" attribute
var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd &&
rng.ContainerNameNS=="{http://www.mydomainname.com/Journalist}Para") {
nd.removeAttributeNS("http://www.mydomainname.com/Journalist", "Id");
}
```

setAttribute
Creates the specified attribute for the element, with the specified value. This method is not namespace aware. In an XML file, you can set an attribute's value to be an empty string (" ").

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMElement_object.setAttribute("strAttrName", "strAttrValue");
```

Usage in VBScript

```
DOMElement_object.setAttribute "strAttrName", "strAttrValue"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in a Para element in a Journalist DTD document
// Set "Id" attribute
var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd && nd.nodeName=="Para") {
nd.setAttribute("Id","tg888");
}
```

setAttributeNode
Adds the attribute node DOMAttr to the element. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMElement_object.setAttributeNode(Attr);
```

Usage in VBScript

```
DOMElement_object.setAttributeNode(Attr)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in a Para element in a Journalist DTD document
// Set "Id" attribute
var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd && nd.nodeName=="Para") {
var IdAttr = ActiveDocument.createAttribute("Id");
IdAttr.nodeValue = "tg888";
nd.setAttributeNode(IdAttr);
}
```

setAttributeNodeNS
Adds the attribute node DOMAttr to the element. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMElement_object.setAttributeNodeNS(domAttr);
```


Usage in VBScript

```
DOMElement_object.setAttributeNodeNS domAttr
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in a Para element in a Journalist schema document
// Set "Id" attribute
var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd && rng.ContainerNameNS ==
"{http://www.mydomainname.com/Journalist}Para") {
var IdAttr =
ActiveDocument.createAttributeNS("http://www.mydomainname.com/Journalist",
 "Id");
IdAttr.nodeValue = "tg888";
nd.setAttributeNodeNS(IdAttr);
}
```

setAttributeNS

Creates the specified attribute for the element, with the specified value. The attribute is local to the specified namespace. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

No return value

Usage in JScript

```
DOMElement_object.setAttributeNS("strNamespaceNameURI", "strLocalName", "strAttrValue");
```

Usage in VBScript

```
DOMElement_object.setAttributeNS "strNamespaceNameURI", "strLocalName", "strAttrValue"
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste in a Para element in a Journalist schema document
// Set "Id" attribute
var rng = ActiveDocument.Range;
var nd = rng.ContainerNode;
if (nd &&
rng.ContainerNameNS=="{http://www.mydomainname.com/Journalist}Para") {
nd.setAttributeNS("http://www.mydomainname.com/Journalist", "Id", "tg888");
}
```

DOMEntity

A DOMEntity object corresponds to an entity definition in a DTD or document. DOMEntity inherits the DOMNode interface. A DOMEntity object has no child nodes.

A DOMEntity object is obtained from a DOMDocumentType object; the `DOMDocumentType.entities` property returns a DOMNamedNodeMap of DOMEntity objects. An individual DOMEntity is obtained from the DOMNamedNodeMap via the getNamedItem property.

For example, here we assume that we know the name of the entity that we want, and that that name is stored in the variable `entName`:

```
// XMetaL Script Language JSCRIPT:
var doctype = ActiveDocument.doctype;
var entList = doctype.entities;
var thisEntity = entList.getNamedItem(entName);
```

`thisEntity` is a DOMEntity object corresponding to the definition of the entity named by `entName`. Further information can be extracted from `thisEntity` using the DOMEntity properties.

Properties

notationName
For unparsed entities, returns the name of the notation for the entity. For parsed entities, returns null.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMEntity_object.notationName;
```

Usage in VBScript

```
DOMEntity_object.notationName
```

Example

```
// XMetaL Script Language JSCRIPT:
// assume that we know the name of the entity that we want,
// and that that name is stored in the variable entName

var doctype = ActiveDocument.doctype;
var entList = doctype.entities;
var thisEntity = entList.getNamedItem(entName);
ActiveDocument.Host.Alert("Notation name: " + thisEntity.notationName);
```

publicId
The entity's public identifier. If a public identifier was not specified, returns null.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMEntity_object.publicId;
```

Usage in VBScript

```
DOMEntity_object.publicId
```

Example

```
// XMetaL Script Language JSCRIPT:
// assume that we know the name of the entity that we want,
// and that that name is stored in the variable entName

var doctype = ActiveDocument.doctype;
var entList = doctype.entities;
var thisEntity = entList.getNamedItem(entName);
ActiveDocument.Host.Alert("Public ID: " + thisEntity.publicId);
```

systemId
The entity's system identifier. If a system identifier was not specified, returns null.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMEntity_object.systemId;
```

Usage in VBScript

```
DOMEntity_object.systemId
```

Example

```
// XMetaL Script Language JSCRIPT:
// assume that we know the name of the entity that we want,
// and that that name is stored in the variable entName

var doctype = ActiveDocument.doctype;
```

```
var entList = doctype.entities;
var thisEntity = entList.getNamedItem(entName);
ActiveDocument.Host.Alert("System ID: " + thisEntity.systemId);
```

DOMEntityReference

A DOMEntityReference object corresponds to an entity reference in the document. This interface inherits the DOMNode interface; it has no other properties or methods.

A DOMEntityReference node that corresponds to a text entity can have the following child nodes, in any number and order:

- DOMElement
- DOMText
- DOMComment
- DOMProcessingInstruction
- DOMCDATASection
- DOMEntityReference

In the common case in which the replacement text of the entity is a text string, the node has one child node, a DOMText node representing the string.

A DOMEntityReference node that corresponds to an external entity has no child nodes. To get information about the entity, you must obtain the corresponding DOMEntity object, as explained in the section on the DOMEntity interface.

DOMImplementation

The DOMImplementation interface provides a way of testing whether the current DOM implementation supports a particular feature. There is only one DOMImplementation object for a document; it is obtained by using the Document.implementation property. A DOMImplementation object has no child nodes, and inherits no other interfaces.

Methods

hasFeature

Tests if the DOM implementation implements a specific feature.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
DOMImplementation.hasFeature("strFeature", "strVersion");
```


The allowed values for `strFeature` are 'HTML' and 'XML'. `strVersion` is the DOM version number to be tested against.

Usage in VBScript

```
DOMImplementation.hasFeature("strFeature", "strVersion")
```

Example

```
// XMetaL Script Language JSCRIPT:

// displays true
ActiveDocument.Host.Alert(ActiveDocument.implementation("XML","1.0"));
// displays false
ActiveDocument.Host.Alert(ActiveDocument.implementation("XML","2.1"));
```

DOMNamedNodeMap

A DOMNamedNodeMap object is an unordered collection of DOMNode objects that can be accessed via their object names. The object names correspond to the nodeName property of the DOMNodes. Individual DOMNodes are obtained using the getNamedItem property.

Only nodes for which `DOMNode.nodeName` is meaningful can appear in a DOMNamedNodeMap. In the current implementation, the following properties generate a DOMNamedNodeMap object:

- `DOMDocumentType.entities`
- `DOMDocumentType.notations`
- `DOMNode.attributes`

DOMNamedNodeMaps are read-only except when they represent a collection of attributes (DOMAttr nodes). In this case the named node map functions as an alternate way of accessing an element's attributes, and when a DOMAttr node is added or removed from the named node map, it is also added or removed from the element itself.

Properties

length

The number of DOMNodes in the DOMNamedNodeMap.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
DOMNamedNodeMap_object.length;
```

Usage in VBScript

```
DOMNamedNodeMap_object.length
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste into a 'Journalist' DTD document

var elemList;
var allElemList;
// Get all "Title" elements and display number
elemList=ActiveDocument.getElementsByTagName("Title");
ActiveDocument.Host.Alert(elemList.length);
// Get all elements in the document and display number
allElemList=ActiveDocument.getElementsByTagName("*");
ActiveDocument.Host.Alert(allElemList.length);
```

Methods

getNamedItem

Retrieves a specified DOMNode. The names used to address nodes in a DOMNamedNodeMap are the same as those returned by the `DOMNode.nodeName` property. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNamedNodeMap_object.getNamedItem("strName");
```

Usage in VBScript

```
DOMNamedNodeMap_object.getNamedItem("strName")
```

Example

```
// XMetaL Script Language JSCRIPT:

var mynodelist;
mynodelist = ActiveDocument.getElementsByTagname("*");
if (mynodelist.length>0) {
// grab the first node in the node list
var aNode = mynodelist.item(0);
// get the NamedNodeMap of attributes for this node. var myNodeMap =
aNode.attributes;
// use the getNamedItem() method to obtain
// the attribute node "Id"
var namedItem = myNodeMap.getNamedItem("Id");
```



```
if (namedItem)
ActiveDocument.Host.Alert(namedItem.nodeValue);
}
```

getNamedItemNS
Retrieves a specified DOMNode. The node is local to the specified namespace. The names used to address nodes in a DOMNamedNodeMap are the same as those returned by the `DOMNode.nodeName` property. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNamedNodeMap_object.getNamedItemNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
DOMNamedNodeMap_object.getNamedItemNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:

var mynodelist;
mynodelist =
ActiveDocument.getElementsByTagnameNS("http://www.mydomainname.com/Journalist",
 "*");
if (mynodelist.length>0) {
// grab the first node in the node list
var aNode = mynodelist.item(0);
// get the NamedNodeMap of attributes for this node. var myNodeMap =
aNode.attributes;
// use the getNamedItem() method to obtain
// the attribute node "Id"
var namedItem =
myNodeMap.getNamedItemNS("http://www.mydomainname.com/Journalist", "Id");
if (namedItem)
ActiveDocument.Host.Alert(namedItem.nodeValue);
}
```

item
Returns an indexed DOMNode object in the DOMNamedNodeMap. DOMNamedNodeMaps do not store the nodes in any particular order. This method is provided as a convenient way to enumerate the DOMNamedNodeMap.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNamedNodeMap_object.item(longIndex);
```

Valid indexes range from 0 to the value represented by `length`-1.

Usage in VBScript

```
DOMNamedNodeMap_object.item(longIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:

var myNodeList;
var myNodeMap;
myNodeList = ActiveDocument.getElementsByTagname("*");
if (myNodeList.length>0) {
// grab the first node in the node list
var aNode = myNodeList.item(0);
// get the NamedNodeMap of attributes for this node. var myNodeMap =
aNode.attributes;
// grab the first attribute in the node map
if (myNodeMap.length>0) {
var namedItem = myNodeMap.item(0);
ActiveDocument.Host.Alert(namedItem.nodeValue);
}
}
```

removeNamedItem
Removes a DOMNode with the specified name. This method applies only to named node maps of DOMAttr nodes. Removing an attribute with this method removes it from the corresponding element. This property is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNamedNodeMap_object.removeNamedItem("strNodeName");
```

Usage in VBScript

```
DOMNamedNodeMap_object.removeNamedItem("strNodeName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Remove a DOMAttr node from a node map
```

```
var docElem = ActiveDocument.documentElement;
var attrMap = docElem.attributes;
attrMap.removeNamedItem("Id");
```

removeNamedItemNS
Removes a DOMNode with the specified name. This node is local to the specified namespace. This method applies only to named node maps of DOMAttr nodes. Removing an attribute with this method removes it from the corresponding element. This property is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNamedNodeMap_object.removeNamedItemNS("strNamespaceNameURI", "strLocalName");
```

Usage in VBScript

```
DOMNamedNodeMap_object.removeNamedItemNS("strNamespaceNameURI", "strLocalName")
```

Example

```
// XMetaL Script Language JSCRIPT:
// Remove a DOMAttr node from a node map
var docElem = ActiveDocument.documentElement;
var attrMap = docElem.attributes;
attrMap.removeNamedItemNS("http://www.mydomainname.com/Journalist", "Id");
```

setNamedItem
Adds a DOMNode to the DOMNamedNodeMap. If the new node replaces an existing node with the same name, the previously existing node is returned; otherwise this method returns null. This method applies only to named node maps of DOMAttr nodes. Adding an attribute with this method adds it to the element from which the node map was derived. This method is not namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNamedNodeMap_object.setNamedItem(DOMNode);
```

Usage in VBScript

```
DOMNamedNodeMap_object.setNamedItem(DOMNode)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a new DOMAttr node and add it to a node map
var docElem = ActiveDocument.documentElement;
var attrMap = docElem.attributes;
// Create a DOMAttr node
var newId = ActiveDocument.createAttribute("Id");
newId.value = "xyzzy";
// Add the node to the node map
attrMap.setNamedItem(newId);
// Verify that the attribute has been associated
// with the element
ActiveDocument.Host.Alert(docElem.getAttribute("Id"));
```

setNamedItemNS

Adds a DOMNode to the DOMNamedNodeMap. If the new node replaces an existing node with the same name, the previously existing node is returned; otherwise this method returns null. This method applies only to named node maps of DOMAttr nodes. Adding an attribute with this method adds it to the element from which the node map was derived. This method is namespace aware.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNamedNodeMap_object.setNamedItemNS(domNode);
```

Usage in VBScript

```
DOMNamedNodeMap_object.setNamedItemNS(domNode)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a new DOMAttr node and add it to a node map
var docElem = ActiveDocument.documentElement;
var attrMap = docElem.attributes;
// Create a DOMAttr node
var newId =
ActiveDocument.createAttributeNS("http://www.mydomainname.com/Journalist",
 "Id");
newId.value = "xyzzy";
// Add the node to the node map
attrMap.setNamedItemNS(newId);
// Verify that the attribute has been associated
// with the element
ActiveDocument.Host.Alert(docElem.getAttributeNS("http://www.mydomainname.com/Journalist",
 "Id"));
```


DOMNode

DOMNode is the basic interface of the DOM model. It is a generic interface that most other DOM interfaces inherit (that is, they inherit its properties and methods). In effect, these other interfaces are DOMNodes with different values for the nodeType property.

DOMNode provides the basic properties and methods (firstChild, lastChild, nextSibling, parentNode, previousSibling, and hasChildNodes) for navigating the node structure that the DOM uses to represent documents.

It also provides the following methods for modifying the structure:

- appendChild
- insertBefore
- removeChild
- replaceChild

Note: The DOM specification places restrictions on how nodes can be manipulated. For example, a node can be the child of only one parent node at a time; therefore, if a node is already the child of some node, it is removed from that node's tree before being inserted in the tree of another node. Furthermore, nodes can be used only in the document that they were created in. See the information regarding exceptions in the specification for full details.

Node example: A simple tree-walker

This function walks the DOM tree representation of a document. It uses three user-defined functions, do_starttag, do_endtag, and do_text that perform "enter element", "exit element", and "text" processing, respectively.

```
function readtree(node) {
// Read the DOM tree

// If this is a DOMElement node, do "enter
// element" processing
if (node.nodeType==1) {
do_starttag(node);
}

// Do if this is a DOMText node
if (node.nodeType==3) {
do_text(node);
}

// Process this node's children
if (node.hasChildNodes()) {
readtree(node.firstChild);
}

// If this is a DOMElement node, do "exit
// element" processing
if (node.nodeType==1) {
do_endtag(node);
}

// Continue with this node's siblings
if (node.nextSibling!=null) {
readtree(node.nextSibling)
}
}
```

Properties

attributes

A DOMNamedNodeMap of DOMAttr objects corresponding to the attributes that are set for this node. Returns null if the current node is not a DOMElement.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNamedNodeMap object

Usage in JScript

```
DOMNode_object.attributes;
```

Usage in VBScript

```
DOMNode_object.attributes
```

Example

```
// XMetaL Script Language JSCRIPT:

var elem=ActiveDocument.documentElement;
var attrs = elem.attributes;
//get number of attributes set on this element
ActiveDocument.Host.Alert(attrs.length);
```

childNodes

A DOMNodeList that contains all children of this node. If the node has no children, childNodes still returns a non-null object. Use `DOMNode.hasChildNodes` to test whether a node has children.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNodeList object

Usage in JScript

```
DOMNode_object.childNodes;
```


Usage in VBScript

```
DOMNode_object.childNodes
```

Example

```
// XMetaL Script Language JSCRIPT:

var children=ActiveDocument.documentElement.childNodes;
ActiveDocument.Host.Alert(children.length);
```

firstChild
The first child of the node. If there is no such node, null is returned.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNode object

Usage in JScript

```
DOMNode_object.firstChild;
```

Usage in VBScript

```
DOMNode_object.firstChild
```

Example

```
// XMetaL Script Language JSCRIPT:

var parent = ActiveDocument.documentElement;
if (parent) {
var firstChild = parent.firstChild;
if (firstChild) {
ActiveDocument.Host.Alert(firstChild.nodeName);
}
}
```

lastChild
The last child of the node. If there is no such node, null is returned.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNode

Usage in JScript

```
DOMNode_object.lastChild;
```

Usage in VBScript

```
DOMNode_object.lastChild
```

Example

```
// XMetaL Script Language JSCRIPT:

var parent = ActiveDocument.documentElement;
if (parent) {
var lastChild=parent.lastChild;
if (lastChild) {
ActiveDocument.Host.Alert(lastChild.nodeName);
}
}
```

nextSibling

The node immediately following this node. If there is no such node, null is returned.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

```
DOMNode
```

Usage in JScript

```
DOMNode_object.nextSibling;
```

Usage in VBScript

```
DOMNode_object.nextSibling
```

Example

```
// XMetaL Script Language JSCRIPT:

var parent = ActiveDocument.documentElement;
if (parent) {
var child = parent.firstChild;
if (child) {
nextSibling = child.nextSibling;
if (nextSibling) {
ActiveDocument.Host.Alert(child.nextSibling.nodeName);
```



```
}
}
}
```

nodeName
The name of this node. Returns different names depending on what type of node it is.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Description

The returned name is a qualified name in the namespace of the form `prefix:local name`. If a namespace is not specified, the namespace prefix and colon are omitted. The DOMspecification details the node names for each type of node:

- DOMElement: the element name
- DOMAttr: the attribute name
- DOMText: the string "#text"
- DOMCDATASection: the string "#cdata-section"
- DOMEntityReference: the entity name
- DOMCharacterReference: the string "#character-reference"
- DOMEntity: the entity name
- DOMProcessingInstruction: the PI target
- DOMComment: the string "#comment"
- Document (DOMDocument): the string "#document"
- DOMDocumentType: the document type name
- DOMDocumentFragment: the string "#document-fragment"
- DOMNotation: the notation name

Usage in JScript

```
DOMNode_object.nodeName;
```

Usage in VBScript

```
DOMNode_object.nodeName
```

Example

```
// XMetaL Script Language JSCRIPT:
```

```
var nd = Selection.ContainerNode;
if (nd) {
ActiveDocument.Host.Alert(nd.nodeName);
}
```

nodeType
An integer representing the type of the underlying object.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Integer

Description

The allowed values are specified by the DOM specification:

- 1: DOMElement
- 2: DOMAttr
- 3: DOMText
- 4: DOMCDATASection
- 5: DOMEntityReference
- 6: DOMEntity
- 7: DOMProcessingInstruction
- 8: DOMComment
- 9: Document (DOMDocument)
- 10: DOMDocumentType
- 11: DOMDocumentFragment
- 12: DOMNotation
- 505: DOMCharacterReference (extension to DOM)

Usage in JScript

```
DOMNode_object.nodeType;
```

Usage in VBScript

```
DOMNode_object.nodeType
```

Example

```
// XMetaL Script Language JSCRIPT:

var parent=ActiveDocument.documentElement;
if (parent) {
```



```
var child=parent.firstChild;
if (child) {
ActiveDocument.Host.Alert(child.nodeType);
}
}
```

nodeValue
Returns or sets the value of the node.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Description

The DOM specification details which types of nodes have values, and where the values come from:

- DOMAttr: the attribute value
- DOMText: the text
- DOMCDATASection: content of the section
- DOMProcessingInstruction: everything in the PI except the target.
- DOMComment: content of the comment.
- DOMCharacterReference: the *decimal* number corresponding to the character (extension to DOM)
- All other nodes: `null`.

Set applies only to nodes whose value is not `null` by definition.

Usage in JScript

```
vbl = DOMNode_object.nodeValue;
DOMNode_object.nodeValue = "strNewValue";
```

Usage in VBScript

```
vbl = DOMNode_object.nodeValue
DOMNode_object.nodeValue = "strNewValue"
```

Example

```
// XMetaL Script Language JSCRIPT:

var elem=ActiveDocument.documentElement;

if (elem) {
var attrs = elem.attributes;

//confirm that the first attribute has a value
if (attrs.item(0)){
```

```
//display the value of the first attribute
ActiveDocument.Host.Alert(attrs.item(0).nodeValue);
} else { // the first attribute has no value
ActiveDocument.Host.Alert("The first attribute of the document element <"
 + elem.nodeName + "> has no value.");
}
}

// XMetaL Script Language JSCRIPT:
// Set a comment's text to "" (an empty string)
var rng = ActiveDocument.Range;
curNode = rng.ContainerNode;
if (curNode){
if (curNode.nodeType==8) {
curNode.nodeValue="";
}
}
```

ownerDocument
The Document object associated with this node. If the current node is a Document object, this property is null.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Document object

Usage in JScript

```
DOMNode_object.ownerDocument;
```

Usage in VBScript

```
DOMNode_object.ownerDocument
```

Example

```
// XMetaL Script Language JSCRIPT:

var nd = Selection.ContainerNode;
if (nd) {
var doc = nd.ownerDocument;
ActiveDocument.Host.Alert(doc.doctype.name);
}
```

parentNode
The parent of this node. If a parent node does not exist, this property is null.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNode object

Usage in JScript

```
DOMNode_object.parentNode;
```

Usage in VBScript

```
DOMNode_object.parentNode
```

Example

```
// XMetaL Script Language JSCRIPT:

var nd = Selection.ContainerNode;
if (nd) {
var parent = nd.parentNode;
ActiveDocument.Host.Alert(parent.nodeName);
}
```

previousSibling
The node immediately preceding this node. If there is no such node, this property is null.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

DOMNode object

Usage in JScript

```
DOMNode_object.previousSibling;
```

Usage in VBScript

```
DOMNode_object.previousSibling;
```

Example

```
// XMetaL Script Language JSCRIPT:

var parent = ActiveDocument.documentElement;
if (parent) {
var child = parent.lastChild;
if (child) {
```

```
prevSibling = child.previousSibling;
if (prevSibling) {
ActiveDocument.Host.Alert(prevSibling.nodeName);
}
}
}
```

Methods

appendChild

Adds the specified node to the end of the list of children of this node. A node can be the child of only one parent node at a time; therefore, if nodeNewChild is already the child of some other node, it is removed from that node's tree before being appended to the current node.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNode_object.appendChild(nodeNewChild);
```

Usage in VBScript

```
DOMNode_object.appendChild(nodeNewChild)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Insert a new Para element (node) at the end of
// the document. // Paste this example in a "Journalist" DTD
// document

// Create the new element node
var newElem = ActiveDocument.createElement("Para");
// Get the document's top-level node
var topElem = ActiveDocument.documentElement;
// Append the new node to the top-level node
if (topElem) {
topElem.appendChild(newElem);
}
```

cloneNode

Returns a duplicate of this node. The duplicate node has no parent. Cloning a DOMElement copies all attributes and their values, including those generated by the XML processor to represent defaulted attributes.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNode_object.clone(booleanDeep);
```

If `booleanDeep` is set to True, a recursive clone of the subtree under the specified node is done. If it is set to False, only the node itself is cloned.

Usage in VBScript

```
DOMNode_object.clone(booleanDeep)
```

Example

```
//XMetaL Script Language JSCRIPT:

var elem = ActiveDocument.documentElement;
var clone = elem.cloneNode(true);

if (clone) {
ActiveDocument.Host.Alert(elem.nodeName + " was cloned successfully.");
}
```

hasChildNodes

Returns True if the node has children, and False otherwise.

Applies to

XMetaL XMAX and XMetaL Author

Returns

Boolean

Usage in JScript

```
DOMNode_object.hasChildNodes();
```

Usage in VBScript

```
DOMNode_object.hasChildNodes
```

Example

```
// XMetaL Script Language JSCRIPT:

var topElem=ActiveDocument.documentElement;
if (topElem) {
ActiveDocument.Host.Alert(topElem.hasChildNodes());
}
```

insertBefore

Inserts the specified node before the existing specified child node.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNode_object.insertBefore(nodeNewChild, nodeRefChild);
```

If `nodeRefChild` is null, inserts `nodeNewChild` at the end of the list of children. In that case, this method behaves like appendChild. A node can be the child of only one parent node at a time; therefore, if `nodeNewChild` is already the child of some other node, it is removed from that node's tree before being inserted in the current node.

Usage in VBScript

```
DOMNode_object.insertBefore(nodeNewChild, nodeRefChild)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Insert a new Sect1 element (node) before the
// first Sect1 in the document. Paste this example
// in a "Journalist" DTD document

// Get the top level node
var topElem = ActiveDocument.documentElement;
if (topElem) {
// Create the new Sect1 node
var newSect1 = ActiveDocument.createElement("Sect1");
// Visit each child node of the top-level node
// until the desired node (if it exists) is found
var nextChild=topElem.firstChild;
while (nextChild && nextChild.nodeName!="Sect1") {
nextChild = nextChild.nextSibling;
}
// If the desired node was found, insert the new
// node before it. if (nextChild){
var firstSect1 = nextChild;
topElem.insertBefore(newSect1,firstSect1);
}
}
```

removeChild

Removes and returns the specified child node.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNode_object.removeChild(nodeChildToRemove);
```

Usage in VBScript

```
DOMNode_object.removeChild(nodeChildToRemove)
```

Example

```
// XMetaL Script Language JSCRIPT:
// Use tree-walker to remove all elements of
// a particular type.

function readtree(node) {

// If the node represents an "InlineGraphic"
// element, remove it

if (node.nodeName=="InlineGraphic") {
// Save a pointer to the next sibling
var nextNode = node.nextSibling;
// Remove the node
node.parentNode.removeChild(node);
// Continue with this node's siblings
if (nextNode) {
readtree(nextNode)
}
}
else {
// Process this node's children
if (node.hasChildNodes()) {
readtree(node.firstChild);
}
// Continue with this node's siblings
if (node.nextSibling!=null) {
readtree(node.nextSibling)
}
}
}
readtree(ActiveDocument.documentElement);
```

replaceChild
Replaces the specified old child node with the specified new child node in the list of children, and returns the old child node.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNode_object.replaceChild(nodeNewChild, nodeOldChild);
```

A node can be the child of only one parent node at a time; therefore, if `nodeNewChild` is already the child of some other node, it is removed from that node's tree before being inserted in the current node.

Usage in VBScript

```
DOMNode_object.replaceChild(nodeNewChild, nodeOldChild)
```

Example

```
//XMetaL Script Language JSCRIPT:

var parent = ActiveDocument.documentElement;

if (parent){

if (parent.firstChild){
var oldFirstChild = parent.replaceChild(parent.firstChild,
parent.lastChild);
ActiveDocument.Host.Alert(oldFirstChild.nodeName + " is no longer the
first child of parent element " + parent.nodeName + ".");
} else {
ActiveDocument.Host.Alert(parent.nodeName + " has no children to
replace.");
} else {
ActiveDocument.Host.Alert("No node to demonstrate replaceChild method.");
}
```

DOMNodeList

A DOMNodeList object represents an ordered collection of DOMNode objects. Each object is addressable by its index number.

In the current implementation, the following properties and methods return a DOMNodeList object:

- `Document.ChangedNodes`
- `Document.ChangedNodesByKey`
- `Document.getElementsByTagName`
- `Document.getElementsByTagNameNS`
- `DOMDocument.getElementsByTagName`
- `DOMDocument.getElementsByTagNameNS`
- `DOMDocument.getNodesByXPath`
- `DOMNode.childNodes`
- `DOMElement.getElementsByTagName`
- `DOMElement.getElementsByTagNameNS`
- `DOMElement.getNodesByXPath`

Properties

length

The number of DOMNodes in the DOMNodeList.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
DOMNodeList_object.length;
```

Usage in VBScript

```
DOMNodeList_object.length
```

Example

```
// XMetaL Script Language JSCRIPT:

var mynodelist;
// return a NodeList containing all the elements
// of the active document
mynodelist = ActiveDocument.getElementsByTagname("*");
ActiveDocument.Host.Alert(mynodelist.length);
```

Methods

item

Returns the indexed DOMNode object.

Applies to

XMetaL XMAX and XMetaL Author

Returns

DOMNode object

Usage in JScript

```
DOMNodeList_object.item(longIndex);
```

Valid index range from 0 to the value represented by `length`-1.

Usage in VBScript

```
DOMNodeList_object.item(longIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:

var mynodelist;
mynodelist = ActiveDocument.getElementsByTagname("*");
// get the first node in the list using the item() method
```

```
var firstNode = mynodelist.item(0);
ActiveDocument.Host.Alert(firstNode.nodeName);
```

DOMNotation

A DOMNotation object represents a notation declared in theDTD. DOMNotation inherits the DOMNode interface.

A DOMNotation object is obtained from a DOMDocumentType object; the `DOMDocumentType.notations` property returns a DOMNamedNodeMap of DOMNotation objects. An individual DOMNotation object is obtained from the DOMNamedNodeMap via the getNamedItem(strName) property.

For example, here we assume that we know the name of the notation that we want, and that that name is stored in the variable `notName`:

```
// XMetaL Script Language JSCRIPT:
var doctype=ActiveDocument.doctype;
var notList=doctype.notations;
var thisNotation=entList.getNamedItem(notName);
```

`thisNotation` is a DOMNotation object corresponding to the notation named by `entName`. Further information can be extracted from `thisNotation` using the DOMNotation properties.

Properties

publicId
The notation's public identifier. If a public identifier was not specified, returns null.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMNotation_object.publicId;
```

Usage in VBScript

```
DOMNotation_object.publicId
```

Example

```
// XMetaL Script Language JSCRIPT:
// assume that we know the name of the notation that we want,
// and that that name is stored in the variable notName
```



```
var doctype=ActiveDocument.doctype;
var notList=doctype.notations;
var thisNotation=entList.getNamedItem(notName);
ActiveDocument.Host.Alert("Public ID: " + thisNotation.publicId);
```

systemId
The notation's system identifier. If a system identifier was not specified, returns null.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMNotation_object.systemId;
```

Usage in VBScript

```
DOMNotation_object.systemId
```

Example

```
// XMetaL Script Language JSCRIPT:
// assume that we know the name of the notation that we want,
// and that that name is stored in the variable notName

var doctype=ActiveDocument.doctype;
var notList=doctype.notations;
var thisNotation=entList.getNamedItem(notName);
ActiveDocument.Host.Alert("System ID: " + thisNotation.systemId);
```

DOMProcessingInstruction

A DOMProcessingInstruction object corresponds to a processing instruction. This interface inherits the DOMNode interface. DOMProcessingInstruction has no child nodes.

To create a node representing a processing instruction, use `Document.createProcessingInstruction`.

To insert a processing instruction directly, use `Selection.InsertProcessingInstruction` to create a processing instruction.

Properties

data
The content of the processing instruction. This starts with the first non-whitespace character after the target, and ends with the character immediately preceding the '?>' delimiter.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
DOMProcessingInstruction_object.data;
```

Usage in VBScript

```
DOMProcessingInstruction_object.data
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a PI node and insert it at the end of the
// current container

var newPI = ActiveDocument.createProcessingInstruction(
"my_formatter", "endpage");
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
curNode.appendChild(newPI);

// display the content of the processing instruction
ActiveDocument.Host.Alert(newPI.data);
```

target
The target of the processing instruction. XML defines this as being the first token following the markup that begins the processing instruction.

Applies to

XMetaL XMAX and XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
DOMProcessingInstruction_object.target;
```

Usage in VBScript

```
DOMProcessingInstruction_object.target
```

Example

```
// XMetaL Script Language JSCRIPT:
// Create a PI node and insert it at the end of the
// current container

var newPI = ActiveDocument.createProcessingInstruction(
"my_formatter", "endpage");
var rng = ActiveDocument.Range;
var curNode = rng.ContainerNode;
curNode.appendChild(newPI);

// display the target of the processing instruction
ActiveDocument.Host.Alert(newPI.target);
```

DOMText

The DOMText interface represents a text string. It inherits the DOMCharacterData interface (and via DOMCharacterData, DOMNode), and has one method of its own, splitText.

DOMText nodes have no child nodes. To obtain the text represented by a DOMText node, use the data property.

Methods

splitText
Breaks the DOMText node into two DOMText nodes at the specified offset, keeping both in the tree as siblings.

Applies to

XMetaL Author and XMetaL XMAX

Returns

DOMText object

Usage in JScript

```
DOMText_object.splitText(longOffset);
```

The current node then contains only the content up to the `longOffset` point. The new DOMText node, which is inserted as the next sibling of this node, contains all the content at and after the `longOffset` point.

Usage in VBScript

```
DOMText_object.splitText(longOffset)
```

XInclude interfaces

Properties

This section lists all XInclude-specific properties.

ActiveXIncludeNode

Provides the access to the current XInclude element when its target doc fragment is being attached. This property should only be accessed in the On_Document_Process_XInclude_Complete macro.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Access

Read only

Returns

DOMElement object

Usage in JScript

```
ActiveDocument.ActiveXIncludeNode
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.ActiveXIncludeNode;
```

ConvertXIncludeTargetToXML

This property returns a string that represents the XML fragment that is resolved based on the target document location (strHref) and XPointer expression string (strXPtr).

If strHref is empty, the ActiveDocument is used. The XPointer expression can be passed in as an empty string if it is not needed. The target document is required to be parsable by ActiveDocument's DTD or schema. The XInclude elements in the target document will not be resolved recursively, only the top level XInclude is resolved. If the resolution fails, an empty string is returned. One usage of this API is to provide an XML string for previewing selected target in user interfaces.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Returns

String

Usage in JScript

```
ActiveDocument.ConvertXIncludeTargetToXML(strHref,
    strXPointer)
```

Example

```
//XMetaL Script Language JSCRIPT:
//The file test.xml must exist and be compatible with the current doctype.
var strHref = "C:\\temp\\test.xml";
var strXPointer = "";
var xmlFrag = ActiveDocument.ConvertXIncludeTargetToXML(strHref, strXPointer);
ActiveDocument.Host.Alert(xmlFrag);
```

ExportWithXIncludesToFile

This property exports the combined infoset to a file. The combined infoset contains the ActiveDocument and all its xincluded target document fragments. The file should be writable.

Note: This property should not be used with an XML document displayed in source view.

Applies to

XMetaL® Author

Returns

No return value

Usage in JScript

```
ActiveDocument.ExportWithXIncludesToFile(filepath)
```

Example

```
XMetaL Script Language JSCRIPT:
//Run this on a document containing one or more XInclusion(s)
var dirname  = "C:\\temp";
var filename = "test.xml";
var filepath = dirname + "\\" + filename;
if (Application.WritableDirExists(dirname)) {
 ActiveDocument.ExportWithXIncludesToFile(filepath);
}
```

ExportWithXIncludesToXML

This property returns the combined infoset as a string. The combined infoset contains ActiveDocument and all its xincluded target document fragments.

Note: This property should not be used with an XML document displayed in source view.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Returns

String

Usage in JScript

```
ActiveDocument.ExportWithXIncludesToXML()
```

Example

```
//XMetaL Script Language JSCRIPT:
//Run this on a document containing one or more XInclusion(s)
var xmlStr = ActiveDocument.ExportWithXIncludesToXML();
ActiveDocument.Host.Alert(xmlStr);
```

GetXIncludeTargetState

This property returns the state regarding to resolving XInclude target document.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Returns

Integer

Name	Value	Description
sqXITgtUndefined	-1	the passed in param is not an xinclude node.
sqXITgtNone	0	xinclude node does not have target doc associated.
sqXITgtValid	1	xinclude node has target doc associated.
sqXITgtFallback	2	xinclude node has fallback content associated.

Usage in JScript

```
ActiveDocument.GetXIncludeTargetState(xincludeNode)
```

Example

```
//XMetaL Script Language JSCRIPT:
//Run this after moving the current selection into and out of
//XInclude and non-XInclude nodes
var xiState = ActiveDocument.GetXIncludeTargetState(Selection.ContainerNode);
switch (xiState) {
  case -1 :
    ActiveDocument.Host.Alert("-1: not an xinclude node");
    break;
  case 0 :
    ActiveDocument.Host.Alert("0: node does not have target doc associated");
    break;
  case 1 :
    ActiveDocument.Host.Alert("1: node has target doc associated");
    break;
  case 2 :
    ActiveDocument.Host.Alert("2: node has fallback content associated");
    break;
}
```

HideXInclude

This property hides the associated target document fragment of an XInclude element in the source document if it was displayed.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Returns

No return value

Description

HideXInclude behaves as follows:

- If `ctxNode` is an XInclude element in the source document, its associated target document fragment is hidden if it was displayed.
- if `ctxNode` is not an XInclude element in the source document and it has an XInclude element in the source document as an ancestor, the associated target document fragment of the immediate ancestor in the source document (XInclude element) is hidden if it was displayed.
- If `ctxNode` is not an XInclude element and has no XInclude element as an ancestor, all displayed xincluded target document fragments in its descendants (if any) are hidden.
- If `ctxNode` is NULL, XMetaL returns an exception.

Note: This property only affects the visibility of the XInclude target document. It does not have effect on the freshness of the XInclude target document.

Usage in JScript

```
ActiveDocument.HideXInclude(ctxNode)
```

Example

```
//XMetaL Script Language JSCRIPT:
//Run this when the current selection is in an XInclude
//that is showing XIncluded content
ActiveDocument.HideXInclude(Selection.ContainerNode);
```

IsNamespaceAware

This property returns True if the `ActiveDocument` is aware of the XML namespace (e.g., constrained by an XML Schema). If the XML is constrained by a DTD, the API returns False.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Access

Read only

Returns

Boolean

Usage in JScript

```
ActiveDocument.IsNamespaceAware
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.IsNamespaceAware;
```

IsXIncludeNode

This property returns the value True if `domNode` is an XInclude element.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Returns

Boolean

Usage in JScript

```
ActiveDocument.IsXIncludeNode(domNode)
```

Example

```
//XMetaL Script Language JSCRIPT:
//Run this after moving the current selection into and out of
//XInclude and non-XInclude nodes
var isXINode = ActiveDocument.IsXIncludeNode(Selection.ContainerNode);
ActiveDocument.Host.Alert(isXINode);
```

LinkResolutionErrorList

This property returns a ValidationErrorList object containing the list of errors found in the most recent updating xincluded target document fragments. If there is no LinkResolutionErrorList to return, the property will return null or an empty ValidationErrorList object.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Access

Read only

Returns

ValidationErrorList object

Usage in JScript

```
ActiveDocument.LinkResolutionErrorList
```


Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.LinkResolutionErrorList;
```

OpenXIncludeTarget

This property opens the target document referenced by the node if it is an XInclude element or contains an XInclude element in XMetaL Author.

Applies to

XMetaL® Author

Returns

No return value

Usage in JScript

```
ActiveDocument.OpenXIncludeTarget
(xincludeNode)
```

Example

```
// XMetaL Script Language JSCRIPT:
//Run this after moving the current selection into an XInclude with a valid href.
//The document to open should be an XML document, TXT documents will not open.
if (ActiveDocument.IsXIncludeNode(Selection.ContainerNode)) {
 ActiveDocument.OpenXIncludeTarget(Selection.ContainerNode);
}
else {
 ActiveDocument.Host.Alert("selection is not inside an XInclude");
}
```

PurgeXIncludeCache

This property clears all documents that are cached during updating xincluded target fragments. It does not affect any resolved xincluded target fragments associated with the XInclude elements.

Applies to

XMetaL® Author

Returns

No return value

Usage in JScript

```
ActiveDocument.PurgeXIncludeCache()
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.PurgeXIncludeCache();
```

SetXIncludeFeature

This property overrides the corresponding settings in XMetaL.ini on a per-document basis.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Returns

No return value

Description

Name	Value	Description
sqXIProcessSingleLevel (1)	true/false	Resolves one level deep for xincluded target fragment
sqXIRefsPropFixup (2)	true/false	Enables references Property Fixup (4.5.3)
sqXINamespaceFixup (3)	true/false	Enables Namespace Fixup (4.5.4)
sqXIBaseUriFixup (4)	true/false	Enables Base URI Fixup (4.5.5)
sqXILangFixup(5)	true/false	Enables Language Fixup (4.5.6)
sqXIUseResolvedImagePath (6)	true/false	Uses resolved absolute filepath as image.href value in xincluded target fragment.
sqXIStrictMode (7)	true/false	XIncluded target fragment needs to be valid within the context (by replacing its corresponding XInclude element).
sqXIUseResolvedImageUrl (8)	true/false	Uses resolved absolute URL as image.href value in xincluded target fragment.

Usage in JScript

```
ActiveDocument.SetXIncludeFeature(2, false);
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.SetXIncludeFeature(2, false);
```

ShowXInclude

This property displays the associated target document fragment of an XInclude element in the source document if it was hidden.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Returns

No return value

Description

ShowXInclude behaves as follows:

- If `ctxNode` is an xinclude element in the source document, its associated target document fragment is displayed if it was hidden.
- if `ctxNode` is not an xinclude element in the source document and it has an xinclude element in the source document as an ancestor, the associated target document fragment of the immediate ancestor in the source document(xinclude element) is displayed if it was hidden.
- If `ctxNode` is not an xinclude element and has no xinclude element as ancestor, all hidden xincluded target document fragments in its descendants(if any) are displayed.
- If `ctxNode` is NULL, XMetaL returns an exception.

Note: This property only affects the visibility of the XInclude target document. It does not have effect on the freshness of the XInclude target document.

Usage in JScript

```
ActiveDocument.ShowXInclude(ctxNode)
```

Example

```
//XMetaL Script Language JSCRIPT:
//Run this when the current selection is in an XInclude
//element that is not showing the XIncluded content
ActiveDocument.ShowXInclude(Selection.ContainerNode);
```

UpdateXInclude

This property updates the associated target document fragment of an XInclude element in the source document.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Returns

No return value

Description

UpdateXInclude behaves as follows:

- If `ctxNode` is an XInclude element in the source document, its associated target document fragment is updated.
- if `ctxNode` is not an XInclude element in the source document and it has an XInclude element in the source document as an ancestor, the associated target document fragment of the immediate ancestor in the source document (XInclude element) is updated.
- If `ctxNode` is not an XInclude element in the source document and has no XInclude element as ancestor, all displayed xincluded target document fragments in its descendants(if any) are updated.
- If `ctxNode` is NULL, XMetaL returns an exception.

Note: This property only affects the freshness of the XInclude target document. It does not have effect on the visibility of the XInclude target document.

Usage in JScript

```
ActiveDocument.UpdateXInclude
(ActiveDocument.documentElement);
```

Example

```
// XMetaL Script Language JSCRIPT:
ActiveDocument.UpdateXInclude(ActiveDocument.
documentElement);
```

TransState

A DOMNode property that returns the transclusion state of the node.

Applies to

XMetaL® XMAX™ and XMetaL® Author

Access

Read only

Returns

Integer

Description

Name	Value	Description
sqTSSrcInit	9	Regular node
sqXISourceT	1024	XInclude node with its target XML fragment shown in the source document.
sqXITarget	2048	Non-top-level node of the xincluded target XML fragment.
sqXITargetRoot	4096	Top-level node of the xincluded target XML fragment.

Usage in JScript

```
node.TransState
```

Example

```
//XMetaL Script Language JSCRIPT:
//Run this after moving the current selection around inside
//a document that includes XInclude nodes
var ts = Selection.ContainerNode.TransState;
ActiveDocument.Host.Alert(ts);
```

ResultsManager

This application property returns the ResultsManager object, which provides an interface for various customizations of the XMetaL Results pane.

Applies to

XMetaL® Author

Access

Read only

Returns

ResultsManager object

Usage in JScript

```
var resultsManager =
    Application.ResultsManager
```

Example

```
// XMetaL Script Language JSCRIPT:
var resultsManager =
    Application.ResultsManager
```

Events

This section lists all XInclude-specific events.

OnLinkResolutionError

This event is initiated after updating XInclude targets. If passed in error list is empty, it means no error. The error list has the type of ValidationErrorList.

Applies to

XMetaL® XMAX™

Usage in JScript

```
myXMControl::OnLinkResolutionError(errorList)
    {}
```

Usage in VBScript

```
Sub myXMControl_OnLinkResolutionError(errorList)
'handle event EndSub
```

Example

```
' VBScript example:
Sub myXMControl_OnLinkResolutionError(errorList)
'handle event EndSub
```

OnResolveLinkURLAndPath

This event is used to customize link URL/Path resolution via script. The corresponding macro for XMetaL Author is `On_Application_Resolve_Link_URL_And_Path`.

Applies to

XMetaL® XMAX™

Description

There are four pieces of information which are passed in.

ResolveNode	The node contains the attribute, which represents a link, e.g. xinclude node.	read-only
ResolveUrl	The value of the attribute, which represents a link. It can be replaced with a new value provided by the script.	read/write
ResolvePath	The value of the physical filepath to which the link points. The value is provided by the script. It is initially empty.	read/write
ParentUrl	The URL of the document that contains the link node.	read-only

Customization

The values that the script provides to ResolveUrl and/or ResolvePath should be absolute.

The resolved values are used by XMetaL® XMAX™. However, they are NOT physically replaced the actual value of the link attribute unless the script does so.

The script can customize ResolveUrl and/or ResolvePath with the following combinations. When either value changes, XMetaL® XMAX™ treats the situation such that both have new values. Consequently, the built-in link resolution will be skipped.

ResolveUrl = original value ResolvePath = ""	• XMetaL® XMAX™ uses built-in link resolution system to resolve links.
ResolveUrl = new value ResolvePath = new value	• XMetaL® XMAX™ skips internal link resolution and uses these values directly. • XMetaL® XMAX™ uses ResolveUrl as ParentUrl to resolve possible nested links. • XMetaL® XMAX™ uses ResolvePath to access the physical file.
ResolveUrl = new value ResolvePath = ""	• If ResolveUrl has a new value, XMetaL® XMAX™ uses this new value. • If ResolveUrl is a file url or unc path. The value is converted to an absolute filepath and set to ResolvePath. • If ResolveUrl is a http url. XMetaL® XMAX™ tries to download the file from HTTP server to local cache. If it is successful, the filepath of the downloaded file is set to ResolvePath. • XMetaL® XMAX™ skips internal link resolution and uses these values directly.
ResolveUrl = "" ResolvePath = new value	• XMetaL® XMAX™ converts ResolvePath to file url (file:///) and set it to ResolveUrl.

	• XMetaL® XMAX™ uses ResolvePath to access the physical file. • XMetaL® XMAX™ skips internal link resolution and uses these values directly.

Built-in Support

XMetaL® XMAX™ triggers this macro in the following cases:

- Resolving xinclude.href.
- Resolving image inside an xincluded target doc.
  1. Images in the top-level source doc does not trigger this macro
     - Use `OnResolveImageUrl` to handle xml:base if needed.
  2. Images in the xincluded target doc triggers this macro
     - Use this macro to resolve nested xml:base if needed.
     - It does not trigger `OnResolveImageUrl` unless ResolveUrl != "" and ResolvePath == ""
- Resolving xml:base while resolving xincluded target doc.
  1. XMetaL® XMAX™ by default only resolves one level xml:base.
  2. Use this macro to resolve nested xml:base if needed.

Usage in JScript

```
myXMControl::OnApplicationResolveLinkURLAndPath
(ResolveNode, ResolveUrl, ResolvePath, ParentUrl){}
```

Usage in VBScript

```
Sub myXMControl_OnApplicationResolveLinkURLAndPath
(ResolveNode, ResolveUrl, ResolvePath, ParentUrl)
  'event handling code
  EndSub
```

Example

```
// XMetaL Script Language JSCRIPT:
myXMControl::OnApplicationResolveLinkURLAndPath
(ResolveNode, ResolveUrl, ResolvePath, ParentUrl){}
```

Methods

This section lists all XInclude-specific methods.

HideLinkLogTab (ResultsManager)

This event closes the Link Log tab in the Results pane.

Applies to

XMetaL® Author

Returns

No return value

Usage in JScript

```
Application.ResultsManager.CloseLinkLogTab()
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.ResultsManager.CloseLinkLogTab();
```

ShowLinkLogTab (ResultsManager)

This method opens the Link Log tab in the Results pane.

Applies to

XMetaL® Author

Returns

No return value

Usage in JScript

```
Application.ResultsManager.ShowLinkLogTab()
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.ResultsManager.ShowLinkLogTab()
```


SQExtras interfaces

The SQExtras interfaces enable you to display dialog boxes for opening and saving files, and choosing folders.

You can use these interfaces to display the XMetaL Color Chooser and the following Windows dialog boxes:

- Open File
- Save File
- Open File (with image preview)
- Save File (with image preview)
- Choose Folder
- Open File (modified to choose folders)

These dialog boxes are supported by the SQExtrasDLL provided with XMetaL. Therefore, before you use any properties or methods, you must instantiate the relevant object, FileDlg, FolderDlg, or ColorChooser. For example:

```
var dlg=new ActiveXObject("SQExtras.FileDlg");
```

Then use the object dialog to call the FileDlg methods to display various dialog boxes.

Each of the dialog boxes that you can display chooses a file or folder and provides its name. It does not open or save the file or folder by itself; your script must do this.

The SQExtras interfaces also provide a method for causing a script to pause for a specified length of time.

ColorChooser

The ColorChooser interface enables you to display the Choose Color dialog box, which provides a browser-safe palette for choosing colors, and enables you to define one or more custom palettes.

Properties

color

Returns the color last chosen using the specified object, in #RRGGBB format. The returned color is obtained from ChooseColor or ChooseColor2.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strColor = ColorChooser_object.color;
```

Usage in VBScript

```
dim strColor = ColorChooser_object.color
```

Example

```
// XMetaL Script Language JSCRIPT:
var myColorChooser = new ActiveXObject("SQExtras.ColorChooser");

if (myColorChooser.ChooseColor()) {
Application.Alert(myColorChooser.color);
}
```

Color2

Returns the color last chosen using the specified ColorChooser object, as a color reference. The returned color is obtained from ChooseColor or ChooseColor2. A color reference (COLORREF) is a 32-bit value used to specify an RGB color. It is used in some programming languages, such as C++.

Applies to

XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
var longColor = ColorChooser_object.Color2;
```

Usage in VBScript

```
dim longColor = ColorChooser_object.Color2
```

Example

```
// XMetaL Script Language JSCRIPT:
var myColorChooser = new ActiveXObject("SQExtras.ColorChooser");

if (myColorChooser.ChooseColor2()) {
Application.Alert(myColorChooser.Color2);
}
```


Methods

ChooseColor

Displays the XMetaL Choose Color dialog box. Returns True if the user clicks **OK**, and False if the user clicks **Cancel**. Use the color or Color2 properties to obtain the chosen color (if any).

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
var boolResult=ColorChooser_object.ChooseColor(["strInitColor"]);
```

`strInitColor` (optional) specifies the color initially selected, in #RRGGBB format.

Usage in VBScript

```
dim boolResult=ColorChooser_object.ChooseColor ["strInitColor"]
```

Example

```
// XMetaL Script Language JSCRIPT:
var myColorChooser = new ActiveXObject("SQExtras.ColorChooser");
if (myColorChooser.ChooseColor("#AABBCC")) {
Application.Alert(myColorChooser.color);
}
```

ChooseColor2

Displays the XMetaL Choose Color dialog box. Returns True if the user clicks **OK**, and False if the user clicks **Cancel**. Use the colorand Color2 property to obtain the chosen color (if any). A color reference (COLORREF) is a 32-bit value used to specify an RGB color. It is used in some programming languages, such as C++.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
var boolResult=ColorChooser_object.ChooseColor2(["intInitColor"]);
```

`intInitColor` (optional) specifies the color initially selected, as a color reference.

Usage in VBScript

```
dim boolResult=ColorChooser_object.ChooseColor2 ["intInitColor"]
```

Example

```
// XMetaL Script Language JSCRIPT:
var obj = new ActiveXObject("SQExtras.ColorChooser" );
if (obj.ChooseColor2(51)) {
Application.Alert(obj.Color2);
}
```

FileDlg

The FileDlg interface enables you to display dialog boxes for opening and saving files.

The FileDlg interface enables you to display the following dialog boxes:

- Standard Windows open file dialog box
- Standard Windows save file dialog box
- Windows open file dialog box with image preview
- Windows save file dialog box with image preview

Properties

FileExt

Returns the file extension of the last file chosen using the specified FileDlg object. Returns null if no file was chosen.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var fExt = FileDlg_object.FileExt;
```

Usage in VBScript

```
dim fExt = FileDlg_object.FileExt
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FileDlg");
var chosen = Dlg.DisplayImageFileDlg(true, "Choose an image");
if (chosen) {
var chosenFile = Dlg.FileName;
```



```
var chosenFileExt = Dlg.FileExt;
if (chosenFileExt=="ps")
Application.Alert("Cannot display this file inline.");
Selection.ContainerAttribute("FileName") = chosenFile;
}
```

FileName
Returns the filename of the last file chosen using the specified FileDlg object. Returns null if no file was chosen.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var fName = FileDlg_object.FileName;
```

Usage in VBScript

```
dim fName = FileDlg_object.FileName
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FileDlg");
var chosen = Dlg.DisplayImageFileDlg(true, "Choose an image");
if (chosen) {
var chosenFile = Dlg.FileName;
Selection.ContainerAttribute("FileName") = chosenFile;
}
```

FileTitle
Returns the base name (the part to the left of the dot) of the last file chosen using the specified FileDlg object. Returns null if no file was chosen.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var fTitle = FileDlg_object.FileTitle;
```

Usage in VBScript

```
dim fTitle = FileDlg_object.FileTitle
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FileDlg");
var chosen = Dlg.DisplayImageFileDlg(true, "Choose an image");
if (chosen) {
var chosenFileTitle = Dlg.FileTitle;
// Use GIF no matter what file was chosen
Selection.ContainerAttribute("FileName") =
chosenFileTitle + ".gif";
}
```

FullPathName

Returns the full path to the last file chosen using the specified FileDlg object. Returns null if no file was chosen.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var fPath = FileDlg_object.FullPathName;
```

Usage in VBScript

```
dim fPath = FileDlg_object.FullPathName
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FileDlg");
var chosen = Dlg.DisplayImageFileDlg(true, "Choose an image");
if (chosen) {
var chosenFile = Dlg.FullPathName;
Selection.ContainerAttribute("FileName") = chosenFile;
}
```

ImageHeight

Returns the height in pixels of the of the last image file chosen using the specified FileDlg object. Returns null if no file was chosen.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var height = FileDlg_object.ImageHeight;
```

Usage in VBScript

```
dim height = FileDlg_object.ImageHeight
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FileDlg");
var chosen = Dlg.DisplayImageFileDlg(true, "Choose an image");
if (chosen) {
var chosenFileHeight = Dlg.ImageHeight;
Selection.ContainerAttribute("Height") = chosenFileHeight;
}
```

ImageWidth

Returns the width in pixels of the of the last image file chosen using the specified FileDlg object. Returns null if no file was chosen.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var width = FileDlg_object.ImageWidth;
```

Usage in VBScript

```
dim width = FileDlg_object.ImageWidth
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FileDlg");
var chosen = Dlg.DisplayImageFileDlg(true, "Choose an image");
if (chosen) {
var chosenFileWidth = Dlg.ImageWidth;
Selection.ContainerAttribute("Width") = chosenFileWidth;
}
```

Methods

DisplayFileDlg

Displays the standard Windows file open or file save dialog box. Returns True if the user clicks **OK**, and False if the user clicks **Cancel**.

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
var result = FileDlg_object.DisplayFileDlg(boolType, [strTitle], [strFilter], [strInitFldr],
 [strDefExt], [strInitFile]);
```

Use the `FullPathName` property to obtain the path of the chosen file (if any).

The type and appearance of the dialog box are controlled by the following parameters:

- `boolType`: `true` displays the Open dialog box; `false` displays the Save dialog box.
- `strTitle`: the dialog box title. The defaults are Open and Save As.
- `strFilter`: the file filter string. The default is `All Files (*.*)|*.*||`.
- `strInitFldr`: the folder initially displayed.
- `strDefExt`: the file extension that is assumed if none is specified.
- `strInitFile`: the default filename.

Usage in VBScript

```
dim result = FileDlg_object.DisplayFileDlg(boolType,
 [strTitle], [strFilter], [strInitFldr], [strDefExt], [strInitFile])
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FileDlg");
var chosen = Dlg.DisplayFileDlg(true, "Open Text File",
"Text Files (*.txt)|*.txt||");
if (chosen) {
var chosenFile = Dlg.FullPathName;
var txtFile = Application.FileToString(chosenFile);
}
```


DisplayImageFileDlg

Displays the standard Windows file open or file save dialog box with an image preview pane. Returns 1 if the user clicks **OK**, and 0 if the user clicks **Cancel**.

Applies to

XMetaL Author

Returns

False (0) or True (a number that is not 0)

Usage in JScript

```
var result = FileDlg_object.DisplayImageFileDlg(boolType, [strTitle], [strFilter],
[strInitFldr], [strDefExt],[strInitFile]);
```

Use the FullPathName property to obtain the path of the chosen file (if any).

The type and appearance of the dialog box are controlled by the following parameters:

- `boolType`: `true` displays the Open dialog box; `false` displays the Save dialog box.
- `strTitle`: the dialog box title. The defaults are Open and Save As.
- `strFilter`: the file filter string. The default is `All Files (*.*)|*.*||`
- `strInitFldr`: the folder initially displayed.
- `strDefExt`: the file extension that is assumed if none is specified.
- `strInitFile`: the default filename

It is recommended that, when developing code using DisplayImageFileDlg, you check for False (number 0) and not True.

Usage in VBScript

```
dim result = FileDlg_object.DisplayImageFileDlg(boolType, [strTitle], [strFilter],
[strInitFldr], [strDefExt], [strInitFile])
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FileDlg");
var chosen = Dlg.DisplayImageFileDlg(true, "Choose an image");
if (chosen) {
var chosenFile = Dlg.FullPathName;
Selection.ContainerAttribute("FileName") = chosenFile;
}
```

FolderDlg

The FolderDlg interface enables you to display the Standard Windows choose folder dialog box and the Standard Windows open file dialog box modified to choose folders.

Properties

FolderPath

Returns the full path to the folder last chosen using the specified FolderDlg object. Returns null if no folder was chosen.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
var strFldrPath = FolderDlg_object.FolderPath;
```

Usage in VBScript

```
dim strFldrPath = FolderDlg_object.FolderPath
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FolderDlg");
var chosen = Dlg.DisplayFolderDlg("Entity Folder");
if (chosen) {
var chosenFldr = Dlg.FolderPath;
var uniqueFile = Application.UniqueFileName(chosenFldr,"ent");
}
```

Methods

DisplayBrowseForFolderDlg

Displays the standard Windows choose folder dialog box. Returns True if the user clicks **OK**, and False if the user clicks **Cancel**. Use the FolderPath property to obtain the path of the chosen folder (if any).

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
var boolResult = FolderDlg_object.DisplayBrowseForFolderDlg([strTitle], [strFilter]);
```


The appearance of the dialog box is controlled by the following parameters:

- `strTitle`: the dialog box title. The default is Select Folder.
- `strInitFldr`: the folder initially displayed.

Usage in VBScript

```
dim boolResult = FileDlg_object.DisplayBrowseForFolderDlg([strTitle], [strFilter])
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FolderDlg");
var chosen = Dlg.DisplayBrowseForFolderDlg("Entity Folder");
if (chosen) {
var chosenFldr = Dlg.FolderPath;
var uniqueFile = Application.UniqueFileName(chosenFldr,"ent");
}
```

DisplayFolderDlg

Displays the standard Windows open file dialog box, modified to choose folders. Returns True if the user clicks **OK**, and False if the user clicks **Cancel**. Use the FolderPath property to obtain the path of the chosen folder (if any).

Applies to

XMetaL Author

Returns

Boolean

Usage in JScript

```
var boolResult = FolderDlg_object.DisplayFolderDlg([strTitle], [strFilter]);
```

The appearance of the dialog box is controlled by the following parameters:

- `strTitle`: the dialog box title. The default is Select Folder.
- `strInitFldr`: the folder initially displayed.

Usage in VBScript

```
dim boolResult = FileDlg_object.DisplayFolderDlg([strTitle], [strFilter])
```

Example

```
// XMetaL Script Language JSCRIPT:
var Dlg = new ActiveXObject("SQExtras.FolderDlg");
var chosen = Dlg.DisplayFolderDlg("Entity Folder");
if (chosen) {
var chosenFldr = Dlg.FolderPath;
var uniqueFile = Application.UniqueFileName(chosenFldr,"ent");
}
```

Methods

The Methods interface supports the GoToSleep method.

Methods

GoToSleep

Causes the script to pause for the specified number of milliseconds. The cursor changes to a ‘busy’ cursor.

Returns

No return value

Usage in JScript

```
Methods_object.GoToSleep(longDelay);
```

Usage in VBScript

```
Methods_object.GoToSleep longDelay
```

Example

```
// XMetaL Script Language JSCRIPT:
var obj=new ActiveXObject("SQExtras.Methods");
var msg="Pauses 5 seconds after you click [OK]."
Application.Alert(msg);
obj.GoToSleep(5000);
Application.Alert("Yawn... that was nice.");
```


Command Bar interfaces

Command bar interfaces are used for manipulating XMetaL menus and toolbars.

CommandBar

A CommandBar object represents an XMetaL toolbar or menu, and contains a collection of all its controls (individual buttons, menu items, or submenus). The collection is accessed through the Controls property.

Properties

BuiltIn

Returns True for a built-in toolbar, False for a custom tool bar.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
var myBoolean = CommandBar_object.BuiltIn;
```

Usage in VBScript

```
dim myBoolean = CommandBar_object.BuiltIn
```

Example

```
// XMetaL Script Language JSCRIPT:
// get the command bars collection
var cmdBars = Application.CommandBars;
cmdBars.Add("My Own Bar");
// get the "Standard" built-in toolbar
var cmdBar = cmdBars.item("Standard");
var customBar = cmdBars.item("My Own Bar");
Application.Alert(cmdBar.BuiltIn);// returns true
Application.Alert(customBar.BuiltIn); // returns false
```

Controls

Returns a collection of the controls (individual buttons, menu items, or submenus) on CommandBar.

Applies to

XMetaL Author

Access

Read only

Returns

CommandBarControls object

Usage in JScript

```
var myControls = CommandBar_object.Controls;
```

Usage in VBScript

```
dim myControls = CommandBar_object.Controls
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
// get the "Standard" built-in toolbar
var StndCmdBar = cmdBars.item("Standard");
// get the CommandBarControls object
// for the Standard toolbar
var stndcontrols = StndCmdBar.Controls;
```

Enabled

Returns or sets enable state of CommandBar. Setting `CommandBar.Enabled` to False disables all of the controls on the toolbar, while True allows the enable state of each control to be set individually.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = CommandBar_object.Enabled;
CommandBar_object.Enabled = boolState;
```

Usage in VBScript

```
dim boolState = CommandBar_object.Enabled
CommandBar_object.Enabled = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
cmdBars.Add("My Own Bar");
```



```
var cmdBar = cmdBars.item("My Own Bar");
cmdBar.Enabled = false;// disable the command bar
```

name
Returns or sets the name of the specified CommandBar object. This name is used in the XMetaL context (right-click) menu of toolbars, and as the palette title if the toolbar is displayed as a floating palette.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
var strName = CommandBar_object.name;
CommandBar_object.name = strName
```

Usage in VBScript

```
dim strName = CommandBar_object.name
CommandBar_object.name = strName
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
Application.Alert(cmdBar.name);
```

Position
Returns or sets the position of CommandBar based upon the assigned value.

Applies to

XMetaL Author

Access

Read/write

Returns

Integer

Usage in JScript

```
var intPosition = CommandBar_object.Position;
CommandBar_object.Position = intPosition;
```

The allowed values are:

- sqcbPosLeft (1): left
- sqcbPosTop (2): top
- sqcbPosRight (3): right
- sqcbPosBottom (4): bottom
- sqcbPosFloating (5): floating

Usage in VBScript

```
dim intPosition = CommandBar_object.Position
CommandBar_object.Position = intPosition
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
cmdBar.Position = 5; // make the standard toolbar float
```

ReadOnly

Returns or sets the readonly status of CommandBar.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = CommandBar_object.ReadOnly;
CommandBar_object.ReadOnly = boolState;
```

Usage in VBScript

```
dim boolState = CommandBar_object.ReadOnly
CommandBar_object.ReadOnly = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
cmdBar.ReadOnly = true;// prevent command bar modification
```

Visible

Returns or sets the visibility of CommandBar.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = CommandBar_object.Visible;
CommandBar_object.Visible = boolState;
```

Usage in VBScript

```
dim boolState = CommandBar_object.Visible
CommandBar_object.Visible = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
cmdBars.Add("My Own Bar");
var cmdBar = cmdBars.item("My Own Bar");
cmdBar.Visible = false;// hide the command bar
```

Methods

Delete

Deletes the CommandBar object. Do not use this method inside the On_Default_CommandBars_Complete event macro.

Applies to

XMetaL Author

Returns

No return value

Description

Users who experience memory problems with their systems can be advised to periodically hold down the Control key while XMetaL starts up until they are asked “Do you want to skip opening the last session's workspace?” They can then click Yes.

Ideally, however, you should adjust scripts to minimize deletion of custom toolbars and delete controls on the toolbar instead. Additionally, your customized user interface should be designed so that toolbar deletion is not required.

Note: Repeatedly deleting toolbars using scripting impairs the performance of XMetaL.

Usage in JScript

```
CommandBar_object.Delete();
```

Usage in VBScript

```
CommandBar_object.Delete
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
cmdBars.Add("My Own Bar");
Application.Alert("Toolbar \"My Own Bar\" created.");
var cmdBar = cmdBars.item("My Own Bar");
cmdBar.Delete();
Application.Alert("Toolbar \"My Own Bar\" deleted.");
```

FindControl

Finds and returns a CommandBarControl in the specified CommandBar, based on the arguments provided.

Applies to

XMetaL Author

Returns

CommandBarControl

Usage in JScript

```
CommandBar_object.FindControl(intControlType, longCmdId, [variantVisible],
[variantRecursive]);
```

`intControlType` can have the following values:

- sqcbcTypeButton (1): button
- sqcbcTypeEdit (2): text box
- sqcbcTypeDropDown (3): drop-down list
- sqcbcComboBox (4): combo box
- sqcbcTypePopup (5): popup

`longCmdId` is a command ID, as returned by `CommandBarControl.Id`.

If `boolVisible` is False, ‘invisible’ controls are searched. This is not recommended.

If `boolRecursive` is True, all submenus in the CommandBar are searched, recursively.

Usage in VBScript

```
CommandBar_object.FindControl(intControlType, longCmdId, [variantVisible],
[variantRecursive])
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
```



```
// get the "Standard" built-in toolbar
var StndCmdBar = cmdBars.item("Standard");
// get the CommandBarControls object
// for the Standard toolbar
var stndcontrols = StndCmdBar.Controls;
// get the id of the first item in standard
// tool bar
var id = stndcontrols.item(1).id;
// if the first item of the standard tool bar is a button,
// store it in the variable found
var found = StndCmdBar.FindControl(1, id);

if (found == null) {
Application.Alert("The first item in the standard tool bar is not a
button.");
} else {
Application.Alert("Found a button.");
}
```

Reset
Restores a built-in CommandBar to its startup configuration. This has no effect on custom CommandBar objects. Do not use this method inside the On_Default_CommandBars_Complete event macro.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
CommandBar_object.Reset();
```

Usage in VBScript

```
CommandBar_object.Reset
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
cmdBar.Reset();
```

CommandBarButton

A CommandBarButton object corresponds to a single toolbar button. CommandBarButton inherits the CommandBarControl interface. In effect, a CommandBarButton object is a CommandBarControl object for which `CommandBarControl.Type` is `sqcbcTypeButton (1)`. It has all of the properties and methods of the generic CommandBarControl interface, in addition to its own properties and methods.

Properties

BuiltInFace

Returns True if the icon/image of the specified control is its built-in face; this property can be set only to True, which resets the face to the built-in one.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolBuiltin = CommandBarButton_object.BuiltInFace;
CommandBarButton_object.BuiltInFace = boolBuiltin;
```

Usage in VBScript

```
dim boolBuiltin = CommandBarButton_object.BuiltInFace
CommandBarButton_object.BuiltInFace = boolBuiltin
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
// get the first button
var cmdBarButton = cmdBar.Controls.item(1);
// check to see if the button's icon/image
// is a built-in face
Application.Alert(cmdBarButton.BuiltInFace);
```

FaceId

Returns or sets the icon/image face ID for the specified control. These IDs refer to images in the icon sets for custom toolbar buttons. To see these images, choose **Tools** > **Macros**, and then click **Choose Image**. To obtain the face ID of a particular image in one of the icon sets, which you can then assign to a control, use Application.MakeFaceId. Do not hard-code a FaceId into a script, as this may produce unexpected results. Always use `Application.MakeFaceId`.

Applies to

XMetaL Author

Access

Read/write

Returns

Long

Usage in JScript

```
var intId = CommandBarButton_object.FaceId;
CommandBarButton_object.FaceId = intId;
```

Usage in VBScript

```
dim intId = CommandBarButton_object.FaceId
CommandBarButton_object.FaceId = intId
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste into a "Journalist" DTD document
// Create a new button and assign an icon to it
var cmdBars = Application.CommandBars;
var stdBar = cmdBars.item("Standard");
var stdBarCtrls = stdBar.Controls;
var newBtn = stdBarCtrls.Add(sqcbcTypeButton);
// Get the fifth button in the first row
var faceId = Application.MakeFaceId("General (Custom)",1,5);
newBtn.FaceId = faceId;
newBtn.OnAction="Insert Citation";
```

Methods

CopyFace

Copies the icon/image of the button to an internal clipboard.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
CommandBarButton_object.CopyFace();
```

Usage in VBScript

```
CommandBarButton_object.CopyFace
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
// get the first button
var cmdBarButton = cmdBar.Controls.item(1);
cmdBarButton.CopyFace();
```

PasteFace
Pastes the contents of the internal clipboard onto the specified command bar button control.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
CommandBarButton_object.PasteFace();
```

Usage in VBScript

```
CommandBarButton_object.PasteFace
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("My own bar");
// get the first button
var cmdBarButton = cmdBar.Controls.item(1);
// copy the graphic from the first button
cmdBarButton.CopyFace();
// get the second button
var cmdBarButton2 = cmdBar.Controls.item(2);
// paste clipboard contents on top of second button
cmdBarButton2.PasteFace();
```

CommandBarControl

A *control* is a single item in a toolbar or menu, for example, a toolbar button, menu item, or submenu. A control is represented by a CommandBarControl object. The collection of all the controls in a toolbar or menu (a CommandBar) is represented by a CommandBarControls object (that is, the object returned by `CommandBar.Controls`).

The CommandBarControl interface is a generic interface of common properties and methods shared by CommandBarButton and CommandBarPopup. In effect, a CommandBarButton is a CommandBarControl with `CommandBarControl.Type` equal to 1 (button); similarly, a CommandBarPopup has `CommandBarControl.Type` equal to 5 (menu).

Properties

BeginGroup
Determines if the control is preceded by a separatator. Returns True if the control is preceded by separator. You can set this property to True to add a separator. You cannot, however, add a separator as the first item in a toolbar. Even if this property is set to True for the first toolbar item, no separator is inserted.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolSeparator = CommandBarControl_object.BeginGroup;
CommandBarControl_object.BeginGroup = boolSeparator;
```

Usage in VBScript

```
dim boolSeparator = CommandBarControl_object.BeginGroup
CommandBarControl_object.BeginGroup = boolSeparator
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
// get the first button
var cmdBarControl = cmdBar.Controls.item(1);
// add a separator
cmdBarControl.BeginGroup = true;
Application.Alert(cmdBarControl.BeginGroup);
```

BuiltIn

Determines if the control is a built-in control. Returns True for a built-in control; False for a custom control.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
var myBoolean = CommandBarControl_object.BuiltIn;
```

Usage in VBScript

```
dim myBoolean = CommandBarControl_object.BuiltIn
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var newCmdBar = cmdBars.Add("MyCommandBar");
newCmdBar.Controls.Add(sqcbcTypeButton);// add a button
newCmdBar.Controls.Add(sqcbcTypePopup);// add a command bar pop up
var cmdBarControl = newCmdBar.Controls.item(1);
// returns false (0) because the control is not built-in
Application.Alert(cmdBarControl.BuiltIn);
// Get the Standard toolbar
var stdCmdBar = cmdBars.item("Standard");
// get the first button
var stdCmdBarControl = stdCmdBar.Controls.item(1);
// returns true(1) because the control is built-in
Application.Alert(stdCmdBarControl.BuiltIn);
```

DescriptionText

Returns the current description for the specified control, or sets the new one. The description is displayed in the status bar of the application when the user positions the mouse cursor over the control.

Applies to

XMetaL Author

Access

Read/write

Returns

Variant

Usage in JScript

```
var myText = CommandBarControl_object.DescriptionText;
CommandBarControl_object.DescriptionText = strText;
```

Usage in VBScript

```
dim myText = CommandBarControl_object.DescriptionText
CommandBarControl_object.DescriptionText = strText
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
// get the first button
var cmdBarControl = cmdBar.Controls.item(1);
Application.Alert(cmdBarControl.DescriptionText);
// assign some text
cmdBarControl.DescriptionText = "some descriptive text";
Application.Alert(cmdBarControl.DescriptionText);
```

Enabled

Returns the control's current enable state or sets the new enable state. For built-in controls, setting Enabled to True causes the application to determine its state, but setting it to False forces it to be disabled. Enabled applies only to user-defined controls.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = CommandBarControl_object.Enabled;
CommandBarControl_object.Enabled = boolState;
```

Usage in VBScript

```
dim boolState = CommandBarControl_object.Enabled
CommandBarControl_object.Enabled = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
// get the first button
var cmdBarControl = cmdBar.Controls.item(1);
Application.Alert(cmdBarControl.Enabled);
cmdBarControl.Enabled = false;// disable
Application.Alert(cmdBarControl.Enabled);
```

Height

Not implemented

Applies to

XMetaL Author

Access

Read/write

Returns

Long

Usage in JScript

n/a

Usage in VBScript

n/a

Id

Returns the command ID for built-in controls; a control's ID determines the built-in action for that control; the value of Id for all custom controls is 1.

Applies to

XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
var myInteger = CommandBarControl_object.Id;
```

Usage in VBScript

```
dim myInteger = CommandBarControl_object.Id
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
// get the first button
var cmdBarControl = cmdBar.Controls.item(1);
// display command bar control ID
Application.Alert(cmdBarControl.Id);
```

Left

Not implemented

Applies to

XMetaL Author

Access

Read/write

Returns

Long

Usage in JScript

n/a

Usage in VBScript

n/a

OnAction

Returns or sets the name of the XMetaL macro to be run when the user clicks or changes the value of the control.

Applies to

XMetaL Author

Access

Read/write

Returns

Variant

Usage in JScript

```
var strName = CommandBarControl_object.OnAction;
CommandBarControl_object.OnAction = strName;
```

Usage in VBScript

```
dim strName = CommandBarControl_object.OnAction
CommandBarControl_object.OnAction = strName
```

Example

Create a macro as follows:

```
<MACRO name="MyMacro" key="Ctrl+Shift+Q" lang="JScript">
Application.Alert("test");
</MACRO>
```

The CommandBarControl object below calls ‘MyMacro’.

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.Add("MyCommandBar");
cmdBar.Controls.Add();// add a button
cmdBar.Controls.Add(5);// add a command bar pop up
// get the button control
var cmdBarControl = cmdBar.Controls.item(1);
// assign the OnAction property to execute
// "MyMacro" when the button is clicked.
cmdBarControl.OnAction="MyMacro";
```

State

Returns or sets the button’s state.

Applies to

XMetaL Author

Access

Read/write

Returns

Integer

Usage in JScript

```
var intState = CommandBarButton.State;
CommandBarButton.State = intState;
```

The allowed values are:

- sqcbcStateUnset (0): unset
- sqcbcStateUp (1): up
- sqcbcStateDown (2): down

Usage in VBScript

```
dim intState = CommandBarButton.State
CommandBarButton.State = intState
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
// get the first button
var cmdBarButton = cmdBar.Controls.item(1);
Application.Alert(cmdBarButton.State);
```

TooltipText

Returns or sets the control's tooltip text.

Applies to

XMetaL Author

Access

Read/write

Returns

Variant

Usage in JScript

```
var strText = CommandBarControl_object.TooltipText;
CommandBarControl_object.TooltipText = strText;
```

Usage in VBScript

```
dim strText = CommandBarControl_object.TooltipText
CommandBarControl_object.TooltipText = strText
```


Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.Add("My Command Bar");
cmdBar.Controls.Add(sqcbcButton);// add a button
var cmdBarControl = cmdBar.Controls.item(1);
cmdBarControl.TooltipText = "My tool tip";
```

Top
Not implemented

Applies to

XMetaL Author

Access

Read/write

Returns

Long

Usage in JScript

n/a

Usage in VBScript

n/a

Type
Returns the type of control.

Applies to

XMetaL Author

Access

Read only

Returns

Integer

Usage in JScript

```
var myInteger = CommandBarControl_object.Type;
```

The allowed values are:

- sqcbcTypeButton (1): button
- sqcbcTypePopup (5): popup (menu)

Usage in VBScript

```
dim myInteger = CommandBarControl_object.Type
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
// get the first button
var cmdBarControl = cmdBar.Controls.item(1);
// display control type
Application.Alert(cmdBarControl.Type);
```

Visible

Returns True if the specified control is visible. Visible is True by default for newly created custom command bar controls. Hide a control by setting this property to False.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolState = CommandBarControl_object.Visible;
CommandBarControl_object.Visible = boolState;
```

Usage in VBScript

```
dim boolState = CommandBarControl_object.Visible
CommandBarControl_object.Visible = boolState
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
cmdBars.Add("My Command Bar");
var cmdBar = cmdBars.item("My Command Bar");
cmdBar.Controls.Add();//add button
// get the first button
var cmdBarControl = cmdBar.Controls.item(1);
cmdBarControl.Visible = false;
```

Width

Not implemented

Applies to

XMetaL Author

Access

Read/write

Returns

Long

Usage in JScript

n/a

Usage in VBScript

n/a

Methods

Copy

Not implemented.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

n/a

Usage in VBScript

n/a

Delete

Deletes the specified CommandBarControl from the collection it is contained in.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
CommandBarControl_object.Delete();
```

Usage in VBScript

```
CommandBarControl_object.Delete
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
// get the first button
var cmdBarControl = cmdBar.Controls.item(1);
cmdBarControl.Delete();// delete it from the command bar
Application.Alert("deleted command bar button");
cmdBar.Reset();// bring it back
```

Execute
Runs the macro or built-in command assigned to CommandBarControl (using OnAction).

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
CommandBarControl_object.Execute();
```

Usage in VBScript

```
CommandBarControl_object.Execute
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
// get the first button
var cmdBarControl = cmdBar.Controls.item(1);
// execute the built-in command
cmdBarControl.Execute();
```

Move
Not implemented.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

n/a

Usage in VBScript

n/a

Reset
Resets CommandBarControl to its original settings.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
CommandBarControl_object.Reset();
```

Usage in VBScript

```
CommandBarControl_object.Reset
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var stdBar = cmdBars.item("Standard");
var fmtBar = cmdBars.item("Formatting");
var fmtBarCtrls = fmtBar.Controls;
var oListBtn = fmtBarCtrls.item(2);
// Move the button to the "Standard" toolbar
oListBtn.Move(stdBar,6);
// Reset control bars to original settings
stdBar.Reset();
fmtBar.Reset();
```

CommandBarControls

The CommandBarControls object is a collection of all the controls for a particular XMetaL toolbar or menu. An individual CommandBarControl can be retrieved through this collection. The collection is accessed through the Controls property of the parent object (a CommandBarButton or CommandBarPopup).

Do not call any API that displays a dialog inside the On_Default_CommandBars_Complete event, including APIs such as Application.Alert, Application.Prompt. You should also not open a document inside this event including the use of Documents.Open or similar APIs.

Properties

count
Returns the number of CommandBarControl objects in this collection.

Applies to

XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
CommandBarControls_object.count;
```

Usage in VBScript

```
CommandBarControls_object.count
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Standard");
Application.Alert(cmdBar.Controls.count);
```

item

Returns the CommandBarControl corresponding to a specified index item.

Applies to

XMetaL Author

Access

Read only

Returns

CommandBarControl object

Usage in JScript

```
CommandBarControls_object.item(variantIndex);
```

Allowed values for `variantIndex` are 1 to the value represented by `count`, or a control name.

Usage in VBScript

```
CommandBarControls_object.item(variantIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
```



```
var cmdBar = cmdBars.item("Standard");
Application.Alert(cmdBar.Controls.item(1).TooltipText);
```

Methods

Add

Creates a new CommandBarControl derived object and adds it to this collection; returns a CommandBarButton or CommandBarPopup object.

Applies to

XMetaL Author

Returns

CommandBarControl object

Usage in JScript

```
CommandBarControls_object.Add([intControlType], [variantCmdId], [variantBefore],
[variantParam]);
```

The optional arguments are:

- `intControlType` specifies the type of CommandBarControl created. Allowed values are:
  - sqcbcTypeButton (1): button (the default)
  - sqcbcTypePopup (5): popup (menu)
- `variantCmdId` is an integer that specifies a built-in command to clone; if this argument is omitted, a blank custom control of the specified type is added to the command bar.
- `variantBefore` is a number that indicates the position of the new control in the collection; the new control is inserted before the control at this position. If this argument is omitted, the control is added at the end of the specified command bar.
- `variantParam`: associates a control with a string that can be looked up with the item property.

Usage in VBScript

```
CommandBarControls_object.Add([intControlType], [variantCmdId], [variantBefore],
[variantParam])
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.Add("MyCommandBar");
cmdBar.Controls.Add();// add a button
cmdBar.Controls.Add(5);// add a command bar pop up
```

CommandBarPopup

XMetaL menus, submenus, and menu items are all manipulated through the CommandBarPopup interface. You can drill down into the object through the Controls property.

CommandBarPopup inherits the CommandBarControl interface. In effect, a CommandBarPopup object is a CommandBarControl object for which `CommandBarControl.Type` is `sqcbcTypePopup (5)`. It has all of the properties and methods of the generic CommandBarControl interface, in addition to its own properties and methods.

At the top level, a CommandBarPopup represents a menu such as the **File** or **Edit** menu. Using the Controls property would return a CommandBarControls collection of all the submenus and menu items immediately under that menu.

The Controls property of a CommandBarPopup that represents a submenu returns a CommandBarControls collection that contains all the submenus and menu items in that submenu. The Controls property of a CommandBarPopup that represents a menu item would return an empty CommandBarControls collection: that is, the `CommandBarControls.count` property would have a value of zero. By using the `CommandBarControls.Add` method, a menu item can be changed into a submenu (that is, if you add a menu item to a popup, the popup changes from a menu item to a menu).

Properties

BuiltInFace

Returns True if the icon/image on a menu item's push button is its built-in face.This property can be set only to True, which resets the face to the built-in one.

Applies to

XMetaL Author

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolBuiltin = CommandBarPopup_object.BuiltInFace;
CommandBarPopup_object.BuiltInFace = true;
```

Usage in VBScript

```
dim boolBuiltin = CommandBarPopup_object.BuiltInFace
CommandBarPopup_object.BuiltInFace = True
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
```



```
var cmdBar = cmdBars.item("Menu bar");

// get the "Insert menu" (a CommandBarPopup object)
var cmdBarPopup = cmdBar.Controls.item("Insert");

// This is how to obtain a collection of
// all controls under the "Insert" menu item.
var commandBarControls = cmdBarPopup.Controls;

// Copy the "Element" push-button
commandBarControls.item(1).CopyFace();

// Paste the push-button onto the "Entity Reference" menu item
commandBarControls.item(2).PasteFace();

// Returns false, as the icon for the "Entity Reference"
// menu item has changed from its default
Application.Alert(commandBarControls.item(2).BuiltInFace);

// Restore the "Entity Reference" menu item to its original state
// (no icon)
commandBarControls.item(2).BuiltInFace = true;
```

Caption
Returns or sets the menu, submenu, or menu item caption text.

Applies to

XMetaL Author

Access

Read/write

Returns

String

Usage in JScript

```
vbl = CommandBarPopup_object.Caption;
CommandBarPopup_object.Caption = strCaption;
```

Usage in VBScript

```
vbl = CommandBarPopup_object.Caption
CommandBarPopup_object.Caption = strCaption
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.Add("MyCommandBar");
// add a command bar pop up
var cmdBarPopup = cmdBar.Controls.Add(sqcbcTypePopup);
// display the caption
Application.Alert(cmdBarPopup.Caption);
// change the caption
cmdBarPopup.Caption = "Change the Name";
```

Controls

Returns the collection of menu items in the menu or submenu.

Applies to

XMetaL Author

Access

Read only

Returns

CommandBarControls object

Usage in JScript

```
CommandBarPopup_object.Controls;
```

Usage in VBScript

```
CommandBarPopup_object.Controls
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Menu bar");
// get the "File" CommandBarPopup
var cmdBarPopup = cmdBar.Controls.item("File");
Application.Alert(cmdBarPopup.Caption);
// This is how to obtain a collection of
// all controls under the "File" menu item.
var commandBarControls = cmdBarPopup.Controls;
```

faceId

Returns or sets the icon/image face ID for the specified menu item's push button. These IDs refer to images in the icon sets for custom toolbar buttons. To see these images, choose **Tools** > **Macros**, and then click **Choose Image**. To obtain the face ID of a particular image in one of the icon sets, which you can then assign to a control, use `Application.MakeFaceId`. Do not hard-code a `faceId` into a script, as this may produce unexpected results. Always use `Application.MakeFaceId` to generate a `faceId`.

Applies to

XMetaL Author

Access

Read/write

Returns

Long

Usage in JScript

```
var intId = CommandBarPopup_object.faceId;
CommandBarPopup_object.faceId = intId;
```

Usage in VBScript

```
dim intId = CommandBarPopup_object.faceId
CommandBarPopup_object.faceId = intId
```

Example

```
// XMetaL Script Language JSCRIPT:
// Paste into a "Journalist" DTD document
// Add a "Citation" menu item to the "Insert" menu
// and add a custom push button

var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Menu bar");

// get the "Insert menu" (a CommandBarPopup object)
var cmdBarPopup = cmdBar.Controls.item("Insert");

// This is how to obtain a collection of
// all controls under the "Insert" menu item.
var commandBarControls = cmdBarPopup.Controls;

// add the "Citation" menu item with a push button
var newMenuItem = commandBarControls.Add(sqcbcTypePopup);
// Get the fifth button in the first row
var faceId = Application.MakeFaceId("General (Custom)",1,5);
newMenuItem.faceId = faceId;
newMenuItem.OnAction="Insert Citation";
newMenuItem.Caption ="Citation";
```

Methods

CopyFace

Copies the icon/image of the specified menu item’s push button to an internal clipboard.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
CommandBarPopup_object.CopyFace();
```

Usage in VBScript

```
CommandBarPopup_object.CopyFace
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Menu bar");

// get the "Insert menu" (a CommandBarPopup object)
var cmdBarPopup = cmdBar.Controls.item("Insert");

// This is how to obtain a collection of
// all controls under the "Insert" menu item.
var commandBarControls = cmdBarPopup.Controls;

// Copy the "Element" push-button
commandBarControls.item(1).CopyFace();

// Paste the push-button onto the "Entity Reference" menu item
commandBarControls.item(2).PasteFace();

// Returns false, as the icon for the "Entity Reference"
// menu item has changed from its default
Application.Alert(commandBarControls.item(2).BuiltInFace);

// Restore the "Entity Reference" menu item to its original state
// (no icon)
commandBarControls.item(2).BuiltInFace = true;
```

PasteFace

Pastes the contents of the internal clipboard onto the specified the specified menu item's push button.

Applies to

XMetaL Author

Returns

No return value

Usage in JScript

```
CommandBarPopup_object.PasteFace();
```

Usage in VBScript

```
CommandBarPopup_object.PasteFace
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
var cmdBar = cmdBars.item("Menu bar");

// get the "Insert menu" (a CommandBarPopup object)
var cmdBarPopup = cmdBar.Controls.item("Insert");

// This is how to obtain a collection of
// all controls under the "Insert" menu item.
var commandBarControls = cmdBarPopup.Controls;

// Copy the "Element" push-button
commandBarControls.item(1).CopyFace();
```



```
// Paste the push-button onto the "Entity Reference" menu item
commandBarControls.item(2).PasteFace();

// Returns false, as the icon for the "Entity Reference"
// menu item has changed from its default
Application.Alert(commandBarControls.item(2).BuiltInFace);

// Restore the "Entity Reference" menu item to its original state
// (no icon)
commandBarControls.item(2).BuiltInFace = true;
```

CommandBars

This is a collection of all CommandBar objects in XMetaL Author (toolbars and menus). A CommandBar object represents one particular toolbar, which can be either built-in or custom. There is only one CommandBars object in XMetaL Author. This object is obtained by using the `Application.CommandBars` property.

All CommandBar objects can be prevented from modification while running XMetaL Author by changing the XMetaL configuration file as follows:

```
enable_toolbar_customization = false
```

Properties

Count

Returns the number of menu bars and toolbars in the CommandBars collection.

Applies to

XMetaL Author

Access

Read only

Returns

Long

Usage in JScript

```
CommandBars.Count;
```

Usage in VBScript

```
CommandBars.Count
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.Alert("Command bars = " +
Application.CommandBars.Count);
```

item
Returns the CommandBar object specified within the CommandBars collection.

Applies to

XMetaL Author

Returns

CommandBar object

Usage in JScript

```
CommandBars_object.item(variantIndex);
```

`variantIndex` can be a number (1 to the value represented by `CommandBars.count`) or the name of the command bar.

Usage in VBScript

```
CommandBars_object.item(variantIndex)
```

Example

```
// XMetaL Script Language JSCRIPT:
Application.CommandBars.Add("My Toolbar");
//access the command bar
var cmdBar = Application.CommandBars.item("My Toolbar");
Application.Alert(cmdBar.name);
```

HasResults
Returns true if the **View** > **Results** submenu is present. This property is used in conjunction with `AddToResults`.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
var hasResults = CommandBars_object.HasResults;
```

Usage in VBScript

```
Dim hasResults = CommandBars_object.HasResults
```


Example

```
//XMetaL Script Language JSCRIPT:
Application.Alert(Application.CommandBars.HasResults);
```

HasSamples
Returns true if the **Help** > **Samples** submenu is present. For backward-compatibility reasons, there is an INI variable, `help_samples_visible`. When it is set to false, the **Help** > **Samples** submenu will be removed at startup. This property is used in conjunction with `AddToSamples`.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
var hasSamples = CommandBars_object.HasSamples;
```

Usage in VBScript

```
Dim hasSamples = CommandBars_object.HasSamples
```

Example

```
//XMetaL Script Language JSCRIPT:
Application.Alert(Application.CommandBars.HasSamples);
```

IsImportantContextPopup
Returns true if the context menu item represents an important or specific action based on the current document selection. Menu items such as spelling suggestions or track change acceptance or rejection qualify as important menu items since the selection is on a misspelt word or a tracked change. This API can only be called during the On_Context_Menu event macro.

Applies to

XMetaL Author

Access

Read only

Returns

Boolean

Usage in JScript

```
var isImportant = CommandBars_object.IsImportantContextPopup;
```

Usage in VBScript

```
Dim isImportant = CommandBars_object.IsImportantContextPopup
```

Example

```
//XMetaL Script Language JSCRIPT:
var popup = Application.ActiveContextMenu;
if (popup) {
  var ctrls = popup.Controls;
  var i = 1;

  // Find position of first non-important popup
  for ( ; i <= ctrls.Count; i++) {
    if (!Application.CommandBars.IsImportantContextPopup(ctrls.item(i))
{
      break;
    }
  }

  // Insert less important menu item after
  var lessImportantPopup = ctrls.Add(sqcbcTypePopup, -1, i);
  lessImportantPopup.DescriptionText = "Insert &Author";
  lessImportantPopup.OnAction = "Insert Author";
  lessImportantPopup.FaceId = -1;

  var journalist2 = Application.CommandBars("Journalist 2");
  if (journalist2) {
    ctrls = journalist2.Controls;
    if (ctrls.Item(3).OnAction == "Insert Author") {
      lessImportantPopup.FaceId = ctrls.Item(3).FaceId;
    }
  }
}
```

Name

Returns the toolbar base filename associated with this CommandBars collection (for example, journalist.tbr). If no documents are open, returns default.tbr.

Applies to

XMetaL Author

Access

Read only

Returns

String

Usage in JScript

```
CommandBars.Name;
```


Usage in VBScript

```
CommandBars.Name
```

Example

```
//XMetaL Script Language JSCRIPT:
Application.Alert(CommandBars.Name);
```

Methods

Add

Creates and adds to the collection a new CommandBar object with the given name and position.

Applies to

XMetaL Author

Returns

CommandBar object

Description

Usage in JScript

```
CommandBars_object.Add("strName", [intPosition]);
```

The allowed values for `intPosition` are:

- sqcbPosLeft (1): left
- sqcbPosTop (2): top
- sqcbPosRight (3): right
- sqcbPosBottom (4): bottom
- sqcbPosFloating (5): floating (the default)

Usage in VBScript

```
CommandBars.Add("strName", [intPosition])
```

Example

```
// XMetaL Script Language JSCRIPT:
// create a floating command bar
Application.CommandBars.Add("My Toolbar");
```

AddToResults

Creates and appends a menu item to the collection of **View** > **Results** submenu items only if a menu item with the same caption doesn't already exist. The method returns a new CommandBarControl object that represents the menu item.

Applies to

XMetaL Author

Returns

CommandBarControl object

Description

Usage in JScript

```
var commandBarControl = CommandBars_object.AddToResults("strCaption", "strOnAction",
[boolSeparator], [strDescription], [intFaceId]);
```

The parameters are:

- `strCaption` = menu item caption text
- `strOnAction` = macro name to invoke
- `boolSeparator` = insert separator before menu item
- `strDescription` = status line text with addition info
- `intFaceId` = icon id to display beside menu item text

Usage in VBScript

```
Application.CommandBars.AddToResults "Annotation comments", "List All Comments", true
```

Example

```
// XMetaL Script Language JSCRIPT:
if (Application.CommandBars.HasResults) {
  Application.CommandBars.AddToResults("Annotation comments", "List All
Comments", true);
}
```

AddToSamples

Creates and appends a menu item to the collection of **Help** > **Samples** submenu items only if a menu item with the same caption doesn't already exist. The method returns a new CommandBarControl object that represents the menu item.

Applies to

XMetaL Author

Returns

CommandBarControl object

Description

Usage in JScript

```
var commandBarControl = CommandBars_object.AddToSamples("strCaption", "strOnAction",
[boolSeparator], [strDescription], [intFaceId]);
```


The parameters are:

- `strCaption` = menu item caption text
- `strOnAction` = macro name to invoke
- `boolSeparator` = insert separator before menu item
- `strDescription` = status line text with addition info
- `intFaceId` = icon id to display beside menu item text

Usage in VBScript

```
Application.CommandBars.AddToSamples("Cameras In &Focus (Journalist)",
"__Journalist_RevealSample__")
```

Example

```
// XMetaL Script Language JSCRIPT:
// create a floating command bar
if (Application.CommandBars.HasSamples) {
  Application.CommandBars.AddToSamples("My Sample", "Show my sample
macro");
}
```

FindControl

Searches for and returns a CommandBarControl within the complete set of XMetaL Author command bars, based on the arguments provided.

Applies to

XMetaL Author

Returns

CommandBarControl object

Usage in JScript

```
CommandBars_object.FindControl(intControlType, longCmdId, [boolVisible], [boolRecursive]);
```

The allowed values for `intControlType` are:

- sqcbcTypeButton (1): button
- sqcbcTypeEdit (2): text box
- sqcbcTypeDropDown (3): drop-down list
- sqcbcComboBox (4): combo box
- sqcbcTypePopup (5): popup

`longCmdId` is a command ID, as returned by `CommandBarControl.Id`.

If `boolVisible` is False, invisible controls are searched. This is not recommended.

If `boolRecursive` is True, all submenus in the `CommandBar` are searched recursively.

Usage in VBScript

```
CommandBars_object.FindControl(intControlType, longCmdId, [boolVisible], [boolRecursive])
```

Example

```
// XMetaL Script Language JSCRIPT:
var cmdBars = Application.CommandBars;
// get the "Standard" built-in toolbar
var StndCmdBar = cmdBars.item("Standard");
// get the CommandBarControls object
// for the Standard toolbar
var stndcontrols = StndCmdBar.Controls;
// get the id of the first item in standard
// tool bar
var id = stndcontrols.item(1).id;
// if the first item of the standard tool bar is a button,
// store it in the variable found
var found = cmdBars.FindControl(1, id);

if (found == null) {
Application.Alert("The first item in the standard tool bar is not a
button.");
} else {
Application.Alert("Found a button.");
}
```


XFT interfaces

All form objects, including the form background and frame, have properties. Object properties are displayed in the Property Sheet. The scripting interface lists all properties and methods available for an object.

XFT properties

The following code samples include an object called ‘ObjTest’. This name must correspond to an object that already exists on the form.

Accelerator

The Windows accelerator key used to access the object (for example, Alt+A, Alt+1). Can be set at design or run time.

Applies to

- ComboBox
- EditBox
- ListBox
- MultiEditBox

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Accelerator= long;
```

Usage in VBScript

```
ObjTest.Accelerator = long
```

Example

```
// Set Accelerator of the given object to ALT + ! (or ALT + SHIFT + 1)
// Name of the object that contains the property/method is ObjTest.

//set accelerator to a ASCII value of "!"
var acl = '!';
ObjTest.Accelerator = acl.charCodeAt(0);

//Prompt the current accerlator status
var status = 'ObjTest.Accelerator = ' + ObjTest.Accelerator() + ' (String:
 ' + String.fromCharCode(ObjTest.Accelerator()) + ')';
Application.Alert(status);
```

AlignHorizontal

Sets the horizontal alignment for the text in the selected text box. Can be set to left, right, or center. This property can be set only at design time.

Applies to

Text

Access

Read-write

Returns

N/A

AlignVertical

Sets the vertical alignment for the text in the selected text box. Can be set to top, bottom, or center. This property can be set only at design time.

Applies to

Text

Access

Read-write

Returns

N/A

Alignment

Sets the text justification for the given control to left, right, or center.

Applies to

- Button
- Connector
- EditBox
- Frame
- MultiEditBox
- Text

Note: This property is not supported at run time for the following objects:

- Connector
- EditBox
- MultiEditBox

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Alignment = long;
```

Usage in VBScript

```
ObjTest.Alignment = long
```

Example

```
// Set Alignment to either left, center, or right.
// Name of the object that contains the property/method is ObjTest.

// Set the value of Alignment to: Left: 0 , Center: 6, Right: 2
ObjTest.Alignment = 0;
Application.Alert('Alignment is set to left');

ObjTest.Alignment = 6;
Application.Alert('Alignment is set to center');

ObjTest.Alignment = 2;
Application.Alert('Alignment is set to right');
```

AlignTextLeft

If set to 1, moves the text to the left side of the control, and the check button or radio button to the right side of the control. Although you can set this property at run time, it changes the appearance of the control only if run from the control's own OnInitialize event.

Applies to

- CheckButton
- RadioButton

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.AlignTextLeft = long;
```

Valid values are

- 0 (No)
- 1 (Yes)

Usage in VBScript

```
ObjTest.AlignTextLeft = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of AlignTextLeft to 0 (no) or 1 (yes).

ObjTest.AlignTextLeft = 1; //yes
Application.Alert('Text is set to be aligned on the left side of the
control');

ObjTest.AlignTextLeft = 0; //no
Application.Alert('Text is not set to be aligned on the left side of the
 control');
```

AnchorSnaps

Opens the Anchor Snaps dialog which contains a set of anchor snap tie properties for the selected control. This property is available at design time only, and only if you have selected **View** > **Snap Points**.

Applies to

- Bitmap
- Border
- Ellipse
- Freehand
- Polyline
- Text

Access

Read-write

Returns

N/A

Description

Objects can be permanently connected by line snapping. When they are moved, the lines (or polylines) are moved to maintain the connections.The endpoints of lines and polylines are ‘snap ties’ which are matched up with ‘anchor points’ in other objects.

By selecting this property at design time, you open the Anchor Snaps dialog which contains a set of anchor snap tie properties for the selected control. The fields that can be set using the AnchorSnaps property are indicated in the following table.

Table 8: Fields

Field	Description
Type	Used to select the location of the snap point (e.g., top, bottom, right, topLeft, etc.) to be added or deleted.
dX	The offset of the snap point along the x-axis (positive or negative integer). A positive integer will offset the snap to the right while a negative integer will offset the snap point to the left.

Field	Description
dY	The offset of the snap point along the y-axis (positive or negative integer). A positive integer will offset the snap to the bottom while a negative integer will offset the snap point to the top.
By pixel	When selected, snap point will be offset by pixels other wise by per cent. It is suggested that you offset 'by per cent' for objects that are sizable to ensure that the snap points remain in their appropriate positions.
Max. Snapped In	The number of connections that can be made to the snap point specified (i.e., the second of the two snap points used by the connector).
Max. Snapped Out	The number of connections that can be made from the snap point specified (i.e., the first of the two snap points used by the connector).
Tip	Tooltip text.
Use Grid Offset (dG)	Used to offset the connector to prevent overlapping. Field must be enabled.
Snap Point Color	Selects the color of the snap point and associated connector(s). Field must be enabled.

Angle1

Indicates the starting angle in degrees to begin drawing the arc. You can set this property only at design time. If you want to set angles at run time, use the StartAngle property.

Applies to

Arc

Access

Read-write

Returns

N/A

Description

An Angle1 of 0 indicates the point directly to the right of the centerpoint of the circle, this arc is part of. An Angle2 of 180 indicates the point directly to the left of the centerpoint. Use this property with Angle2 to determine the shape of the Arc. The range is -179 degrees to 180 degrees. Arcs are drawn counter-clockwise, starting from Angle1 and ending at Angle2.

Angle2

Indicates the ending angle in degrees to end drawing the arc. You can set this property only at design time. If you want to set angles at run time, use the EndAngle property.

Applies to

Arc

Access

Read-write

Returns

N/A

Description

An Angle2 of 0 indicates the point directly to the right of the centerpoint of the circle this arc is part of. An Angle2 of 180 indicates the point directly to the left of the centerpoint. Use this property with Angle1 to determine the shape of the Arc. The range is -179 degrees to 180 degrees. Arcs are drawn counter-clockwise, starting from Angle1 and ending at Angle2.

Arrowhead

Indicates which ends of the line to draw arrowheads on.

Applies to

- Connector
- Freehand
- Line
- Polyline

Note: This property is not implemented for the Connector object.

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Arrowhead = long;
```

You can set the following values:

- 0 = None
- 1 = Line end point
- 2 = Line start point
- 3 = Both ends of the line

Usage in VBScript

```
ObjTest.Arrowhead = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of Arrowhead to either 0 (No arrowhead,) 1, (End point,)
 2 (Starting point), or 3 (both)

ObjTest.Arrowhead = 0;
Application.Alert( 'Arrowhead is not attached');

ObjTest.Arrowhead = 1;
Application.Alert( 'Arrow on the end point');

ObjTest.Arrowhead = 2;
Application.Alert( 'Arrow on the starting point');

ObjTest.Arrowhead = 3;
Application.Alert( 'Arrow on the both edges');

ObjTest.ArrowheadHeight = 15;
Application.Alert( 'ObjTest.ArrowheadHeight is: ' +ObjTest.ArrowheadHeight
 + ' pixels.');

ObjTest.ArrowheadHeight = 0;
Application.Alert( 'ObjTest.ArrowheadHeight is: ' +ObjTest.ArrowheadHeight
 + ' pixels.');

ObjTest.ArrowheadHeight = 2;
Application.Alert( 'ObjTest.ArrowheadHeight is: ' +ObjTest.ArrowheadHeight
 + ' pixels.');
```

ArrowheadHeight

Sets the size of the arrowhead in points. The valid range is 2 - 32 points.

Applies to

- Connector
- Freehand
- Line
- Polyline

Note: This property is not implemented for the Connector object.

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.ArrowheadHeight = long;
```

Usage in VBScript

```
ObjTest.ArrowheadHeight = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of Arrowhead to either 0 (No arrowhead,) 1, (End point,)
 2 (Starting point), or 3 (both)

ObjTest.Arrowhead = 0;
Application.Alert( 'Arrowhead is not attached');

ObjTest.Arrowhead = 1;
Application.Alert( 'Arrow on the end point');

ObjTest.Arrowhead = 2;
Application.Alert( 'Arrow on the starting point');

ObjTest.Arrowhead = 3;
Application.Alert( 'Arrow on the both edges');

ObjTest.ArrowheadHeight = 15;
Application.Alert( 'ObjTest.ArrowheadHeight is: ' +ObjTest.ArrowheadHeight
 + ' pixels.');

ObjTest.ArrowheadHeight = 0;
Application.Alert( 'ObjTest.ArrowheadHeight is: ' +ObjTest.ArrowheadHeight
 + ' pixels.');

ObjTest.ArrowheadHeight = 2;
Application.Alert( 'ObjTest.ArrowheadHeight is: ' +ObjTest.ArrowheadHeight
 + ' pixels.');
```

AutoRecord

Not implemented.

Applies to

- TheView

BackColor

Indicates what color to use for the background of the selected control.

Applies to

- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand

- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

Long

Description

When setting this value at design time, values are set as three numbers in the range of 0-255 (i.e., `X.Y.Z`, where all numbers X, Y, and Z are in the range of 0 to 255). When setting this value at run time, a Long value representing these three numbers is used instead. When returning values in code, this property returns values as Long.

Setting this property to Nil indicates a transparent background.

Usage in JScript

```
ObjTest.BackColor = long;
```

Usage in VBScript

```
ObjTest.BackColor = long
```

Example

```
//Name of the object that contains the property/method is ObjTest.
//Set the value of BackColor to number representation of RGB color.
ObjTest.BackColor = 255; //solid red

//Retrieve the value of BackColor
Application.Alert('ObjTest.BackColor: ' +ObjTest.BackColor);
```

Bitmap

Indicates the folder path and filename of the bitmap to use as a background for the selected control. If you try to return this property at run time, a string is only returned if a bitmap was set at run time, and prior to returning this value. No string is returned if the Bitmap property was set at design time.

Applies to

- Bitmap
- Button
- CheckButton
- RadioButton

Note: For the CheckButton control, the OwnerDrawn property must be set before the bitmap is displayed.

Access

Read-write

Returns

String

Description

When setting this property at run time, indicate the filename and path as a string. When setting the Bitmap property at design time, there is a special interface:

Table 9: Interfaces

Name	Description
OK	Save the new property value.
Cancel	Abort (don't save) new property value.
Copy	Copies the specified bitmap to the clipboard.
Paste	Allows a bitmap to be pasted in from the clipboard. 'RLE' specifies whether the pasted bitmap is compressed when saved with the view.
Load	Loads a bitmap from a file. 'Link' saves only the directory path with the view. If unchecked, the entire bitmap file is saved in the view file.
Clear	Clears any bitmap specification.
Transparent	Specifies a color within the bitmap to be made transparent.

Usage in JScript

```
ObjTest.Bitmap = pathToBmp;
```

Usage in VBScript

```
ObjTest.Bitmap = pathToBmp
```

Example

```
// This property sets the border color of frame object.
Application.Alert('Current image: ' + ObjTest.Bitmap);

// Provide a path to bitmap image for this code to work properly.
var pathToBmp = 'C:\windows\Greenstone.bmp';

ObjTest.Bitmap = pathToBmp;
Application.Alert('Current image: ' + ObjTest.Bitmap);
```


BitmapOffState

Indicates whether or not the bitmap is enabled for the selected control.

Applies to

- CheckButton
- RadioButton

Note: For the CheckButton control, the OwnerDrawn property must be set before the bitmap is displayed.

Access

Read-write

Returns

At run time: Boolean value indicating whether or not the bitmap is enabled.

At design time: String representing the folder path and filename of the bitmap for the selected control.

Description

When setting this property at run time, indicate a boolean value. When setting the Bitmap property at design time, there is a special interface:

Table 10: Interfaces

Name	Description
OK	Save the new property value.
Cancel	Abort (don't save) new property value.
Copy	Copies the specified bitmap to the clipboard.
Paste	Allows a bitmap to be pasted in from the clipboard. 'RLE' specifies whether the pasted bitmap is compressed when saved with the view.
Load	Loads a bitmap from a file. 'Link' saves only the directory path with the view. If unchecked, the entire bitmap file is saved in the view file.
Clear	Clears any bitmap specification.
Transparent	Specifies a color within the bitmap to be made transparent.

Usage in JScript

```
ObjTest.BitmapOffState = bool;
```

Usage in VBScript

```
ObjTest.BitmapOffState = bool
```

Example

```
// During run time, this is a boolean value indicating whether bitmap is
 enabled.
Application.Alert('Is bitmap enabled? ' + ObjTest.BitmapOffState);
```

BorderColor

Sets the color of the border around the frame object. This property can be set only at design time; using it at run time will lead to unpredictable behavior. Use this property in conjunction with the BorderStyle and BorderWidth properties.

Applies to

- Frame

Access

Read-write

Returns

N/A

Usage in JScript

```
ObjTest.BorderColor = long;
```

Usage in VBScript

```
ObjTest.BorderColor = long
```

Example

```
// This property sets the border color of frame object.
// Name of the object that contains the property/method is ObjTest.
// Set the value of BorderColor to number representation of RGB color.
ObjTest.BorderColor = 255; //red

// Retrieve the value of BorderColor
Application.Alert('ObjTest.BorderColor: ' +ObjTest.BorderColor);
```

BorderDrawn

Indicates whether the border around the selected control is drawn. When setting this value at run time, use the control's OnInitialize event. Although accessible, this property is not implemented to be set at run time.

Applies to

- EditBox
- ListBox
- MultiEditBox

Access

Read-write

Returns

Boolean

Usage in JScript

```
ObjTest.BorderDrawn = long;
```

Usage in VBScript

```
ObjTest.BorderDrawn = long
```

Example

```
function TheView_OnInitialize() {

// This property tells whether to draw borders around or not.
// Name of the object that contains the property/method is ObjTest.
// Set the value of BorderColor to 0 (not drawn) or 1 (drawn).
// Default is 0.

ObjTest.BorderDrawn = 0;

}
```

BorderStyle

Indicates the border style of the selected control.

Applies to

- Border
- Frame
- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.BorderStyle = long;
```

Valid values are:

- 0 = Normal
- 1 = 3D
- 2 = Sunken
- 3 = Raised

Usage in VBScript

```
ObjTest.BorderStyle = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of BorderStyle to number in the range of 0 to 3.
ObjTest.BorderStyle = 0; //0 -Normal
Application.Alert('ObjTest.BorderStyle: ' +ObjTest.BorderStyle);

ObjTest.BorderStyle = 1; //1 - 3D
Application.Alert('ObjTest.BorderStyle: ' +ObjTest.BorderStyle);

ObjTest.BorderStyle = 2; //2 - Sunken
Application.Alert('ObjTest.BorderStyle: ' +ObjTest.BorderStyle);

ObjTest.BorderStyle = 3; //3 = Raised
Application.Alert('ObjTest.BorderStyle: ' +ObjTest.BorderStyle);
```

BorderWidth

Indicates the width of the border for the selected control in pixels. The range of this property is 0 (for no border) to 79 pixels.

Applies to

- Button
- Frame

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.BorderWidth = long;
```

Usage in VBScript

```
ObjTest.BorderWidth = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of BorderWidth to desired width.
ObjTest.BorderWidth = 15;
Application.Alert( 'ObjTest.BorderWidth: ' +ObjTest.BorderWidth);

ObjTest.BorderWidth = 0;
Application.Alert( 'ObjTest.BorderWidth: ' +ObjTest.BorderWidth);

ObjTest.BorderWidth = 1;
Application.Alert( 'ObjTest.BorderWidth: ' +ObjTest.BorderWidth);
```


Bottom

Indicates the number of pixels from the top of the form that the selected object's bottom is drawn. Although this property can be set for objects that are not visible, no change to the display of the form is made. Setting this property to a value that is less than that of the Top property renders the control invisible.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Bottom = long;
```

Usage in VBScript

```
ObjTest.Bottom = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Retrieve the value of Bottom
Application.Alert('Top: ' +ObjTest.Top + ' | ' + 'Bottom: '
+ObjTest.Bottom);

var height = 100;
ObjTest.Top = 10;
```

```
ObjTest.Bottom = ObjTest.Top + height;

// Retrieve the value of Bottom
Application.Alert('Top: ' +ObjTest.Top + ' | ' + 'Bottom: '
+ObjTest.Bottom);
```

ButtonShape

Changes the bottom border of buttons in order to create several tabs on a single form.

Applies to

- Button

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.ButtonShape = long;
```

When this property is set to 0 (Normal), the button has a normal border and appears as a standard Windows button.

When this property is set to 1 (Property Tab - Inactive), it appears as the top tab of a property sheet for a page that is not visible.

When the property is set to 2 (Property Tab - Active), it appears as the top tab of a property sheet for a page that is currently visible.

By using these properties dynamically at run time, and assigning layers to various controls for the form controls, you can create tabs for use on the form. A typical scenario would be to have several buttons, one set to 2 and the others set to 1 with a single Layer property shown for several controls. When the user clicks on one of the other buttons, its ButtonShape property is set to 2 and all others set to 1 and the layer changed to display a different set of controls. In this way, you can visually and functionally reproduce a set of tabbed pages.

Usage in VBScript

```
ObjTest.ButtonShape = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of buttonShape to 0 (normal), 1 (inactive tab), or 2
(active tab)
ObjTest.ButtonShape = 0;
Application.Alert('Button is now set to have a normal shape');

ObjTest.ButtonShape = 1;
Application.Alert('Button is now set to have an inactive tab shape');
```



```
ObjTest.ButtonShape = 2;
Application.Alert('Button is now set to have an active tab shape');
```

ButtonType

Indicates the type of button.

Applies to

- Button

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.ButtonType = long;
```

The default type is OnClick, which executes the script in the OnClick event. If this property is set to OK, it indicates that changes are to be recorded and the window is closed. If set to Cancel, changes are not recorded and the window is closed.

When setting or getting this property in code, the following values are used:

- 0 = OnClick
- 1 = OK
- 2 = Cancel

Usage in VBScript

```
ObjTest.ButtonType = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of buttonType to number in the range of 0 to 2.

ObjTest.ButtonType = 2; //Cancel button
Application.Alert('Button will behave as a cancel button.');
```

Case

Indicates the case to use when displaying text in a multi-edit box. If set to 0 (None), the case is ignored and both upper and lowercase are allowed. You can also indicate 1 (Lowercase) or 2 (Uppercase); in either instance, all text is converted to the indicated case. This property can be set and read only at design time.

Applies to

- MultiEditBox

Access

Read-write

Returns

N/A

CaseOrPassword

Indicates the case to use when displaying text in an edit box. If set to 0 (None), the case is ignored and both upper and lowercase are allowed. You can also indicate 1 (Lowercase) or 2 (Uppercase); in either instance, all text is converted to the indicated case. If this property is set to 3 (Password), asterisks are echoed back to the user; however, you can read the Text property in code. The case of the password text is preserved. This property can be set and read only at design time.

Applies to

- EditBox

Access

Read-write

Returns

N/A

Check

Indicates whether the check button is checked.

Applies to

- CheckButton

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Check = long;
```

Usage in VBScript

```
ObjTest.Check = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of check to checked (true), or unchecked (false).

ObjTest.Check = 1; //set the status to true
```



```
Application.Alert( 'ObjTest.check is currently checked');
ObjTest.Check = 0; //set the status to false
Application.Alert( 'ObjTest.check is currently not checked');
```

Code

Represents the object name.

Applies to

- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

String

Description

The property (ObjectCode) as displayed in the property sheet is represented in scripting as the Code property. If you set this property at run time, you can get the value back; however, you cannot use the name you set as an object name. See below:

```
' A quick VBScript example:
EditBox1.Code = "NewName"
MsgBox EditBox1.Code ' This returns "NewName"
MsgBox NewName.Code ' This causes an error
```

Usage in JScript

```
ObjTest.code = strName;
```

Usage in VBScript

```
ObjTest.code = strName
```

Example

```
// Name of the object that contains the property is ObjTest.
// You can set the value of code, but change will not affect {ObjectCode}
// in run-time.

// Display original object name.
Application.Alert('Object name is ' +ObjTest.code);
ObjTest.code = 'NewName';

Application.Alert('Object name is now ' +ObjTest.code);
```

Color

Indicates what color to use for the background of the view.

Applies to

- TheFrame
- TheView

Access

Read-write

Returns

Long

Usage in JScript

```
TheView.Color = long;
```

When setting this value at design-time, values are set as three numbers in the range of 0-255 (i.e., `X.Y.Z`, where all numbers are in the range of 0 to 255). When setting this value at run time, a Long value representing these three numbers is used instead. When returning values in code, this property returns values as Long.

Setting this property to Nil indicates a transparent background.

Usage in VBScript

```
TheView.Color = long
```

Example

```
// Set the value of color to number representation of RGB color.

Application.Alert('Original background color: ' +TheView.Color);
TheView.Color = 255; //solid red
Application.Alert('New background color: ' +TheView.Color);
```


ComboType

Indicates whether the user can enter values into the combo box control. If this property is set to 1 (Dropdown), the field is editable by the user. If it is set to 2 (Droplist), then the user can only view the items in the combo box control without editing them. This property is available only at design time.

Applies to

- ComboBox

Access

Read-write

Returns

N/A

CurrCol

Indicates which column the cursor is in for a multi-edit box. You can set this value only at run time. If you are setting a cursor position using this property and the CurrLine property, make sure you set the CurrLine property first, as setting that property resets this property to 0.

Applies to

- MultiEditBox

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.CurrCol = long;
```

Usage in VBScript

```
ObjTest.CurrCol = long
```

Example

```
// Name of the object that contains these properties is ObjTest.
// Write some text into the MultiEditBox before executing this script.

// Display current location of the cursor
Application.Alert('Cursor is at Line: ' +ObjTest.currLine + ', Coumn: '
+ObjTest.currCol);

ObjTest.currLine = 1; //make sure that MultiEditBox has two lines
ObjTest.currCol = 5; //make sure that second line has at least 5 characters

// Display new location of the cursor
```

```
Application.Alert('Cursor is at Line: ' +ObjTest.currLine + ', Coumn: '
+ObjTest.currCol);
```

CurrLine

Indicates which line the cursor is in for a multi-edit box. You can use this property with the CurrCol property. You can set this value at run time only.

Applies to

- MultiEditBox

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.currLine = long;
```

Usage in VBScript

```
ObjTest.currLine = long
```

Example

```
// Name of the object that contains these properties is ObjTest.
// Write some text into the MultiEditBox before executing this script.

// Display current location of the cursor
Application.Alert('Cursor is at Line: ' +ObjTest.currLine + ', Coumn: '
+ObjTest.currCol);

ObjTest.currLine = 1; //make sure that MultiEditBox has two lines
ObjTest.currCol = 5; //make sure that second line has at least 5 characters

// Display new location of the cursor
Application.Alert('Cursor is at Line: ' +ObjTest.currLine + ', Coumn: '
+ObjTest.currCol);
```

CursorPointer

Indicates the type of cursor pointer to display when the mouse pointer hovers over the selected control. You should read and set this value at design time only.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton

- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

N/A

DataField

This property can be used only if a DataSource object is placed on the form.

Applies to

- ComboBox
- EditBox
- ListBox
- Text

DataSource

This property can be used only if a DataSource object is placed on the form.

Applies to

- ComboBox
- EditBox
- ListBox
- Text

Default

When the Default property of a button is set to 1 (Yes), pressing enter on a form, regardless of which control has focus, emulates a user pointing to and clicking on that button. Depending on what the ButtonType property is set to, an OnClick, OK, or Cancel action is performed.

Applies to

- Button

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Default = long;
```

Usage in VBScript

```
ObjTest.Default = long
```

Example

```
// either 0 (not set as default) or 1 (set as default).
// Name of the object that contains the property/method is ObjTest.
// Set the button to be a default button.
ObjTest.Default = !ObjTest.Default; //toggle the status

// Retrieve the value of check
if (ObjTest.Default)
{Application.Alert( 'Button is a default button'); }
else
{Application.Alert( 'Button is not a default button'); }
```

Enable

When a control has its Enabled property set to 0 (No), the control is still visible on the form, but as a ‘grayed out’ control, and users cannot use it. When set to 1 (Yes), the control is enabled and functional. The default value is 1.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton

- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Enable = long;
```

Usage in VBScript

```
ObjTest.Enable = long
```

Example

```
// either 0 (disabled) or 1 (enabled).
// Name of the object that contains the property/method is ObjTest.

ObjTest.Enable = 0; //set the status to disabled
Application.Alert( 'Object is not enabled');

ObjTest.Enable = 1; //set the status to enabled
Application.Alert( 'Object is enabled');
```

EndAngle

Indicates the ending angle in degrees to stop drawing the arc.

Applies to

- Arc

Access

Read-write

Returns

Long

Description

An EndAngle of 0 indicates the point directly to the right of the centerpoint of the circle this arc is part of. An EndAngle of 180 indicates the point directly to the left of the centerpoint. Use this property with StartAngle to determine the shape of the arc. The range is -179 degrees to 180 degrees. Arcs are drawn counter-clockwise, starting from StartAngle and ending at EndAngle.

You can set and get this property only at run time. If you want to set angles at design time, use the Angle2 property.

Usage in JScript

```
ObjTest.EndAngle = long;
```

Usage in VBScript

```
ObjTest.EndAngle = long
```

Example

```
// Name of the arc that contains the property is ObjTest.
// Arc spans from startAngle to endAngle.

Application.Alert( 'Start: ' +ObjTest.startangle + ' | End: ' +
ObjTest.endAngle);

ObjTest.startAngle = 10;
ObjTest.endAngle = 20;

// Retrieve the value of endAngle
Application.Alert( 'Start: ' +ObjTest.startangle + ' | End: ' +
ObjTest.endAngle);
```

FlexHorizontal

Controls how the control behaves when the form is resized. Use this property in conjunction with the FlexVertical property. This property can be set only at design time.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

N/A

Description

The valid values are:

- 0 = None (No resizing - the default)
- 1 = Shift (Move the control keeping the space between it and the right side of the form constant)
- 2 = Expand (Resize the control so that the spaces between it and the left and right sides of the form are constant)
- 3 = Proportional (Resize the control, maintaining the same proportions to the window size)

FlexVertical

Controls how the control behaves when the form is resized. Use this property in conjunction with the FlexHorizontal property. This property can be set only at design time.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

N/A

Description

The valid values are:

- 0 = None (No resizing - the default)

- 1 = Shift (Move the control keeping the space between it and the bottom of the form constant)
- 2 = Expand (Resize the control so that the spaces between it and the top and bottom of the form are constant)
- 3 = Proportional (Resize the control, maintaining the same proportions to the window size)

Font

Sets the font style for the selected object.

Applies to

- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Note: Although available to all of the above properties, this property has no effect on the ComboBox and ListBox objects.

Access

Read-write

Returns

N/A

Description

This property has its own Font dialog to set the font and its attributes for the selected control. The fields on the dialog are indicated in the following table.

Table 11: Fields

Name	Description
Font	The name of the font that the text of the selected control should be displayed in.

Name	Description
Font style	The style of the font (bold, italic, etc.)
Size	The size, in points, of the font.

When setting the font in code, you must set the values in this order:

- longWidth
- longEscapement
- longOrientation
- longWeight
- longItalic (0 or -1)
- longUnderline (0 or -1)
- longStrikeOut (0 or -1)
- longCharSet
- longOutPrecision
- longClipPrecision
- longQuality
- longPitchAndFamily
- stringFontName

Usage in JScript

```
ObjTest.Font = (long, long, long, long, long, long, long, long, long, long, long, long,
long, string);
```

Usage in VBScript

```
ObjTest.Font = (long, long, long, long, long, long, long, long, long, long, long, long,
long, string)
```

Example

```
ObjTest.Font = '-13,0,0,0,400,0,0,0,0,3,3,1,49,Times New Roman';
Application.Alert('Font is now set to Times New Roman')

ObjTest.Font = '-20,0,0,0,700,0,0,0,0,3,3,1,49,Arial';
Application.Alert('Font is now set to Arial')

ObjTest.Font = '-13,0,0,0,400,0,0,0,0,3,3,1,49,Verdana';
Application.Alert('Font is now set to Verdana')
```

ForeColor

Indicates the color of the foreground of the control.

Applies to

- Arc
- Bitmap
- Border
- Button
- CheckButton

- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

Long

Description

When setting this value at design time, values are set as three numbers in the range of 0-255 (i.e., `X.Y.Z`, where all numbers are in the range of 0 to 255). When setting this value at run time, a Long value representing these three numbers is used instead. When returning values in code, this property returns values as Long.

Setting this property to Nil indicates a transparent background.

Usage in JScript

```
ObjTest.ForeColor = long;
```

Usage in VBScript

```
ObjTest.ForeColor = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of foreColor to numeric representation of RBG color.
// For example, Red is 255,and Blue is 16711680.

ObjTest.foreColor = 16711680; //Set to blue.

// Retrieve the value of foreColor
Application.Alert('foreground color is ' +ObjTest.foreColor);
```

Group

Represents the number of the group that the control is part of.

Applies to

- ActiveX
- Button
- CheckButton
- ComboBox
- EditBox
- ListBox
- MultiEditBox
- RadioButton

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Group = long;
```

When designing a form, you can select more than one control and group them together. In order to assign a group, you must set the Group property to 1 (Yes). If you assign a set of controls to a group, you can return the number of the group that a control is a member of in script. If -1 is returned, then the control is not part of any group. Group names can be assigned in the property sheet of the design view; however, the name is not returned using this property, only the group number. You can assign a control to a group at both design time and run time.

Usage in VBScript

```
ObjTest.Group = long
```

Example

```
// Names of the objects that contain the property are ObjTest and ObjTest2.
Application.Alert('Currently object is part of ' + ObjTest.Group);

//now make a new group
ObjTest.Group = 2;
ObjTest2.Group = 2;
//ObjTest and ObjTest2 will now be grouped together.
```

GroupID

Represents the number of the group the control is part of.

Applies to

- Arc
- Bitmap
- Border
- Button
- CheckButton

- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read only

Returns

Long

Usage in JScript

```
ObjTest.GroupID = long;
```

When designing a form, you can select more than one control and group them together. In order to assign a group, you must set the GroupID property to 1 (Yes). If you assign a set of controls to a group, you can return the number of the group that a control is a member of in script. If -1 is returned, then the control is not part of any group. Group names can be assigned in the property sheet of the design view; however, the name is not returned using this property, only the group number.

Usage in VBScript

```
ObjTest.GroupID = long
```

Example

```
// Names of the objects that contain the property are ObjTest and ObjTest2.
Application.Alert('Currently object is part of ' + ObjTest.GroupID);

//now make a new group
ObjTest.GroupID = 2;
ObjTest2.GroupID = 2;
//ObjTest and ObjTest2 will now be grouped together.
```

HatchStyle

Indicates the text line background to use for the selected control. This property can be used with the BackColor property to highly customize the appearance of controls.

Applies to

- Arc

- Bitmap
- Border
- Connector
- Ellipse
- Freehand
- Line
- Polyline
- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.HatchStyle = long;
```

You can set lines in a variety of styles, with the default for this property being no text background. Valid values are:

- 0 = None
- 1 = Horizontal
- 2 = Vertical
- 3 = Diagonal
- 4 = DiagRev
- 5 = Cross
- 6 = DiagCross

Usage in VBScript

```
ObjTest.HatchStyle = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of hatchStyle in the range from 0 to 6.
ObjTest.hatchStyle = 0;
Application.Alert( 'hatchStyle: None');

ObjTest.hatchStyle = 1;
Application.Alert( 'hatchStyle: Horizontal');

ObjTest.hatchStyle = 2;
Application.Alert( 'hatchStyle: Vertical');

ObjTest.hatchStyle = 3;
Application.Alert( 'hatchStyle: Diagonal');

ObjTest.hatchStyle = 4;
Application.Alert( 'hatchStyle: Reverse Diagonal');

ObjTest.hatchStyle = 5;
Application.Alert( 'hatchStyle: Cross');
```

```
ObjTest.hatchStyle = 6;
Application.Alert( 'hatchStyle: Cross Diagonal');
```

Height

Sets the height of the form at design time when used with the TheFrame and TheView objects. You can also modify the form height using the mouse, which updates the Height property automatically.

Applies to

- TheFrame
- TheView

Access

Read-write

Returns

Long

Usage in JScript

```
TheFrame.Height = long;
```

Usage in VBScript

```
TheFrame.Height = long
```

Example

```
// Execute this code multiple times to observe the incremental effect.
// This code increments the size of the form by the amount specified in
variables below.
var xWidth= 10;
var yWidth= 20;

TheFrame.Width += xWidth;
TheFrame.Height += yWidth;
```

HiliteColor

Indicates the color to use when highlighting the selected control. Although this property can be set and read at run time, there is no effect on the appearance of the control. When setting the value in script, use a Long value. When setting this property at design time, there is a color chooser interface.

Applies to

- Border
- Frame
- Hilite
- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.HiliteColor = long;
```

Usage in VBScript

```
ObjTest.HiliteColor = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of hiliteColor to a numeric representation of RGB color.
ObjTest.hiliteColor = 255; //red

// Retrieve the value of hiliteColor
Application.Alert('Current hilite color is: ' +ObjTest.hiliteColor);
```

HorizontalScroll

Sets the state of the horizontal scroll bar in list boxes and multi-edit boxes. Although accessible, this property is not implemented to be set at run time.

Applies to

- ListBox
- MultiEditBox

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.HorizontalScroll = 0;
```

If the property is set to 0 (None), text that is typed into the control will wrap to the beginning of the next line when it reaches the right side of the interior of the control. In this instance, no horizontal scroll bar is drawn, and the width of the text entered into the control is always the same or less than the width of the control interior.

If the property is set to 1 (Always), the horizontal scroll bar is always drawn, and the text entered by the user never wraps to the next line. Instead, the text continues to the right of the right-hand side of the control. In this instance, the user can use the scroll bar, which is always visible, to scroll across the entered text.

If the property is set to 2 (Automatic), the scroll bar is drawn at the time that the text extends past the right side of the interior of the control. Entered text does not wrap to the next line, instead it continues past the right side of the control. Once the user has entered text that is wider than the control, the scroll bar appears and can be used to scroll through the text.

Usage in VBScript

```
ObjTest.HorizontalScroll = 0
```

Example

```
// Set the Horizontal Scroll for object ObjTest
ObjTest.HorizontalScroll = 0; //0 - None
Application.Alert('Horizontal bar is disabled.');

ObjTest.HorizontalScroll = 1; //1 - Always
Application.Alert('Horizontal bar will always appear.');

ObjTest.HorizontalScroll = 2; //2 - Automatic
Application.Alert('Horizontal bar will appear as needed.');
```

Index

Indicates the index of the selected radio button control. When the user selects an item in the radio button control, its index can be returned. You can set this value at run time; however, attempts to set the number higher than the number of items in the list have no effect.

Applies to

- RadioButton

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Index = long;
```

Index numbers start at 0 for the first item in the control.

Usage in VBScript

```
ObjTest.Index = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of index to a value below the number of items present.
// Add at least three items to the radio button before executing this
code.
ObjTest.index = 2; //Third item is selected.
Application.Alert('Third item is selected');
```



```
ObjTest.index = 0; //First item is selected.
Application.Alert('First item is selected');

//NOTE: there is no method for adding items to radio button on the fly.
```

IsWedge

Determines whether to draw a line between the endpoints of an arc control. If set to 1 (Yes), a line is drawn from the starting point (set using Angle1 or StartAngle) to the center point of the circle, then from the center point of the circle to the ending point (set using Angle2 or EndAngle). This property can be set only at design time; however, changing the starting and ending points of the arc will change the shape of the lines drawn using the IsWedge property.

Applies to

- Arc

Access

Read-write

Returns

N/A

Layer

Indicates which layer the selected control belongs to.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

N/A

Description

When assigning layers to controls, the layers must have already been set up using the Layers Sheet dialog (available from the **Layout** -> **Layers** menu).

You can use the Layers Sheet dialog to add or delete view layers (except the default layer). An individual layer can be moved up or down in the list, but this has no effect on the drawing of the view (objects are always drawn according to their Z-order in the Object Sheet dialog). You can select all of the objects on a particular layer.

The current 'Insert Layer' determines which layer new objects are inserted into. An individual layer's view (visible/invisible) and lock (locked/unlocked) states can be set. You can click the check boxes in the appropriate areas to the left of the layer name to toggle these states. Double-clicking on the eye and lock pictures next to the check boxes toggles the view and lock states for all of the layers.

When reading or assigning the layer in script, use the LayerID property.

LayerID

Indicates which layer the selected control belongs to.

Applies to

- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

Long

Description

When assigning layers to controls, the layers must have already been set up using the Layers Sheet dialog (available from the **Layout** -> **Layers** menu).

You can use the Layers Sheet dialog to add or delete view layers (except the default layer). An individual layer can be moved up or down in the list, but this has no effect on the drawing of the view (objects are always drawn according to their Z-order in the Object Sheet dialog). You can select all of the objects on a particular layer.

The current 'Insert Layer' determines which layer new objects are inserted into. An individual layer's view (visible/invisible) and lock (locked/unlocked) states can be set. You can click the check boxes in the appropriate areas to the left of the layer name to toggle these states. Double-clicking on the eye and lock pictures next to the check boxes toggles the view and lock states for all of the layers.

When reading or assigning the layer in script, use the LayerID property.

Usage in JScript

```
ObjTest.layerID = long;
```

The value indicates an index number (range: 0 to number of layers - 1) of the layer the selected control is part of.

Usage in VBScript

```
ObjTest.layerID = long
```

Example

```
// Have at least 2 layers before executing this code
ObjTest.layerID = 1;
TheView.Layer(ObjTest.LayerName()) = false;
Application.Alert('Layer ' + ObjTest.LayerName() + ' is currently
invisible.');
```

Layout

Indicates the position of the bitmap within a Button or Bitmap control. The bitmap and text are laid out based on this value.

Applies to

- Bitmap
- Button

Access

Read only

Returns

Long

Usage in JScript

```
ObjTest.Layout = long;
```

Valid values are:

- 6 = Center
- 0 = Left
- 2 = Right
- 24 = Top
- 8 = Bottom

Usage in VBScript

```
ObjTest.Layout = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Insert image into the bitmap control ObjTest before executing this code
ObjTest.layout = 24; //24 - Top
Application.Alert( 'Object is at top. (layout: ' +ObjTest.layout + ')');

ObjTest.layout = 8; //8 - Bottom
Application.Alert( 'Object is at bottom. (layout: ' +ObjTest.layout +
')');

ObjTest.layout = 6; //6 - Center
Application.Alert( 'Object is at center. (layout: ' +ObjTest.layout +
')');

ObjTest.layout = 0; //0 - Left
Application.Alert( 'Object is at left. (layout: ' +ObjTest.layout + ')');

ObjTest.layout = 2; //2 - Right
Application.Alert( 'Object is at right. (layout: ' +ObjTest.layout + ')');
```

Left

Indicates the number of pixels from the left side of the form that the selected object's left side is drawn. Although this property can be set for objects which are not visible, no change to the display of the form is made. Setting this property to be greater than the Right property renders the selected control invisible.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line

- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Left = long;
```

Usage in VBScript

```
ObjTest.Left = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Retrieve the value of Bottom
Application.Alert('Left: ' +ObjTest.Left + ' | ' + 'Right: '
+ObjTest.Bottom);


var width = 100;
ObjTest.Left = 10;
ObjTest.Right = ObjTest.Left + width;


// Retrieve the value of Bottom
Application.Alert('Left: ' +ObjTest.Left + ' | ' + 'Right: '
+ObjTest.Bottom);
```

ListItems

Each ListItem and its associated value is set using a special dialog which is called when you click on the ListItem property in the design-time property sheet. You can read and set this property only at design time. This means that you cannot dynamically build a ComboBox list or RadioButton list at run time using scripting.

Applies to

- ComboBox
- RadioButton

Access

Read-write

Returns

N/A

Description

The fields on the List Choices dialog are as follows:

- OK = Save changes and exit the dialog
- Cancel = Exit the dialog without saving changes
- Add = Add a choice to the list of choices
- Delete = Delete the currently selected choice in the list
- Up = Move the selected choice up one position in the list
- Down = Move the selected choice down one position in the list
- Name = The name of the choice. This is the displayed text that the user sees
- Value = The value of the choice

MappingEnabled

Enables data mapping. This form property is used as a boolean value. When it is True, data is moved in and out of the form. When it is False, no data transfer to and from the XML document occurs. This property is usually set with the OnXftGetMappingEnabled script event.

Applies to

- TheView

Access

Read-write

Returns

Long, set to either True (-1) or False (0)

Usage in JScript

```
TheView.MappingEnabled = long;
```

Valid values are:

- True (1)
- False (0)

Usage in VBScript

```
TheView.MappingEnabled = long
```

Example

```
//When this property is set to false, changes made in the form will not
// be reported to the parent document, and vice versa.
function TheView_OnXftGetMappingEnabled() {
TheView.MappingEnabled = true;
}
```

MaximizeBox

Indicates whether the control (on the right side of the title bar) to maximize the form is enabled or not.

Applies to

- TheFrame
- TheView

Access

Read-write

Returns

Boolean

Usage in JScript

```
TheFrame.MaximizeBox = bool;
```

Usage in VBScript

```
TheFrame.MaximizeBox = bool
```

Example

```
// This code will adjust the setting of the title bar.
var showMinBox = true;
var showMaxBox = false;

TheFrame.Title = 'This is a sample title';
TheFrame.MaximizeBox = showMaxBox;
TheFrame.MinimizeBox = showMinBox;
```

MinimizeBox

Indicates whether the control (on the right side of the title bar) to minimize the form is enabled or not.

Applies to

- TheFrame
- TheView

Access

Read-write

Returns

Boolean

Usage in JScript

```
TheFrame.MinimizeBox = bool;
```

Usage in VBScript

```
TheFrame.MinimizeBox = bool
```

Example

```
// This code will adjust the setting of the title bar.
var showMinBox = true;
var showMaxBox = false;

TheFrame.Title = 'This is a sample title';
TheFrame.MaximizeBox = showMaxBox;
TheFrame.MinimizeBox = showMinBox;
```

NewDBImport

Displays the XMetaL Import Database dialog. Returns 0 if the user clicks **OK** to close the Import Database dialog, and 1 otherwise.

Returns

Long

Usage in JScript

```
DBImport_object.NewDBImport("strParamFile", "strOutputFile", [boolIsSGML]);
```

XMetaL saves the details of the query in `strParamFile`. The output (an HTML, CALS, or generic XML table) is saved in `strOutputFile`. If `boolIsSGML` is true, the table markup is generated in SGML (this affects only the COLSPEC element of CALS tables).

Usage in VBScript

```
DBImport_object.NewDBImport("strParamFile", "strOutputFile", [boolIsSGML])
```

Example

```
// XMetaL Script Language JSCRIPT:
// Show the dialog box
var dbi = new ActiveXObject("SoftQuad.DBImport");
var ok = dbi.NewDBImport("paramfile.txt", "outputfile.txt");
if (ok) {
//...code to insert the table in the document... }
```

NumDropped

Indicates the number of items to display in the drop-down list when the ComboBox opens. Can be set to any long value, 0 or larger. If it is set to 0, no items are displayed when the combo box gets focus. If it is set to a number larger than the number of items in the ComboBox list, all the items are displayed with no extra whitespace.

Applies to

- ComboBox

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.NumDropped = long;
```

Usage in VBScript

```
ObjTest.NumDropped = long
```

Example

```
// Name of the comboBox that contains the property/method is ObjTest.
ObjTest.NumDropped = 3; //Show 3 items when combobox list is dropped.
```

Object Code

The ‘friendly’ name that you can use to reference the corresponding control in XFT scripts. It identifies the object for convenient access. This property can be changed only at design time.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read only

Orientation

Sets text orientation. Text can be displayed either horizontally or vertically in a TextBox, depending on the value of the Orientation property. If this value is set to 0 (default), text is displayed horizontally. Setting this

value to 1 indicates vertical orientation. Some fonts can only be displayed horizontally (for example, MS Sans Serif).

Applies to

- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Orientation = long;
```

Usage in VBScript

```
ObjTest.Orientation = long
```

Example

```
//First, set the font to something that allows orientation
ObjTest.Font = "-20,0,0,0,700,0,0,0,0,3,3,1,49,Arial"; //Set to Arial.

//Then show the orientations
ObjTest.Orientation = 500;
Application.Alert('Orientation at ' + ObjTest.Orientation);

ObjTest.Orientation = 0;
Application.Alert('Orientation at ' + ObjTest.Orientation);

ObjTest.Orientation = -50;
Application.Alert('Orientation at ' + ObjTest.Orientation);
```

OwnerDrawn

Controls whether the CheckButton control is drawn using the bitmaps specified in the Bitmap and BitmapOffState properties. If set to 1 (Yes), then the bitmaps are used. If set to 0 (No), then the bitmaps are not used. This property can be set only at design time.

Applies to

- CheckButton

Access

Read-write

Returns

N/A

PenStyle

Sets the line style when objects are drawn. You can set this style independently for various controls.

Applies to

- Arc
- Bitmap
- Border
- Connector
- Ellipse
- Freehand
- Line
- Polyline
- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.PenStyle = long;
```

Valid values are:

- 0 = Solid
- 1 = Dash
- 2 = Dot
- 3 = Dash-Dot
- 4 = Dash-Dot-Dot

Usage in VBScript

```
ObjTest.PenStyle = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.

ObjTest.PenStyle = 0; //0 - Solid
Application.Alert( 'Drawn using solid pen. (PenStyle: ' +ObjTest.PenStyle
 + ')');

ObjTest.PenStyle = 1; //1 - Dash
Application.Alert( 'Drawn using dash pen. (PenStyle: ' +ObjTest.PenStyle
 + ')');

ObjTest.PenStyle = 2; //2 - Dot
Application.Alert( 'Drawn using Dot pen. (PenStyle: ' +ObjTest.PenStyle
+ ')');

ObjTest.PenStyle = 3; //3 - Dash-Dot
Application.Alert( 'Drawn using Dash-Dot pen. (PenStyle: '
+ObjTest.PenStyle + ')');

ObjTest.PenStyle = 4; //4 - Dash-Dot-Dot
```

```
Application.Alert( 'Drawn using Dash-Dot-Dot pen. (PenStyle: '
+ObjTest.PenStyle + ')');
```

PenWidth

Sets the line width. When objects are drawn, the width in pixels of the lines used can be set using the PenWidth property. You can set this width independently for various controls. Valid values are 0 to 100. Setting this property to 0 only erases the line for the Line object; for all other objects, the minimum width lines are drawn with is 1 pixel.

Applies to

- Arc
- Bitmap
- Border
- Connector
- Ellipse
- Freehand
- Line
- Polyline
- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.PenWidth = long;
```

Usage in VBScript

```
ObjTest.PenWidth = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.

ObjTest.PenWidth = 0; //0 - No width
Application.Alert( 'Pen width: ' +ObjTest.PenWidth);

ObjTest.PenWidth = 1; //1 - Normal pen
Application.Alert( 'Pen width: ' +ObjTest.PenWidth);

ObjTest.PenWidth = 10; //10 - wide pen.
Application.Alert( 'Pen width: ' +ObjTest.PenWidth);
```

PrintScale

Adjusts the scaling of the view when printing. The default value is 1.0. You can set this property dynamically at run time using the TheView object.

Applies to

- TheFrame
- TheView

Access

Read-write

Returns

Double

Usage in JScript

```
TheView.PrintScale = double;
```

Usage in VBScript

```
TheView.PrintScale = double
```

Example

```
// This code sets the magnification scale of actual window to print image.
TheView.PrintScale = 5.0; //Change to any arbitrary value as needed
```

ReadOnly

Controls whether users can edit text in an edit box or multi-edit box. Users can be prevented from editing text in an edit box or multi-edit box by setting the ReadOnly property to 1 (True). The value of 0, or read-write, is the default value.

Applies to

- EditBox
- MultiEditBox

Access

Read-write

Returns

Boolean

Usage in JScript

```
ObjTest.ReadOnly = bool;
```

Usage in VBScript

```
ObjTest.ReadOnly = bool
```

Example

```
// either 0 (disabled) or 1 (enabled).
// Name of the object that contains the property/method is ObjTest.
ObjTest.ReadOnly = !ObjTest.ReadOnly; //toggle the status

// Retrieve the value of check
if (ObjTest.ReadOnly)
{Application.Alert( 'Object is ReadOnly'); }
else
{Application.Alert( 'Object is not ReadOnly'); }
```

ReportLock

Controls whether the view can be updated by new data. If ReportLock is set to True, the view is a static report view, and is not updated by new data.

Applies to

- TheView

Access

Read-write

Returns

Long

Usage in JScript

```
TheView.ReportLock = long;
```

Usage in VBScript

```
TheView.ReportLock = long
```

Example

```
//Try updating data outside the form after this value is set to true.
TheView.ReportLock = true;
```

Right

Indicates the number of pixels from the left side of the form that the selected object's right side is drawn. Although this property can be set for objects which are not visible, no change to the display of the form is made. Setting this property to be smaller than the Left property renders the selected control invisible.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton

- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Right = long;
```

Usage in VBScript

```
ObjTest.Right = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Retrieve the value of Bottom
Application.Alert('Left: ' +ObjTest.Left + ' | ' + 'Right: '
+ObjTest.Bottom);


var width = 100;
ObjTest.Left = 10;
ObjTest.Right = ObjTest.Left + width;


// Retrieve the value of Bottom
Application.Alert('Left: ' +ObjTest.Left + ' | ' + 'Right: '
+ObjTest.Bottom);
```

RotateAngle

Sets the angle of rotation in degrees for the selected object. Grid points are used to draw the control, so angles that are not possible are adjusted to the nearest grid point on the form. Using this property on the Connector object may render the Connector object invisible.

Applies to

- Arc

- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Note: Although available to all the above objects, this property **has no effect** on the CheckButton, EditBox, Frame, Hilite, MultiEditBox, RadioButton, and Text objects.

Access

Read-write

Returns

Double

Usage in JScript

```
ObjTest.RotateAngle = double;
```

The range of -179 to 180.

Usage in VBScript

```
ObjTest.RotateAngle = double
```

Example

```
// Name of the object that contains the property/method is ObjTest.
ObjTest.RotateAngle = 45;
Application.Alert('Rotate ' + ObjTest.RotateAngle + ' degrees.');

ObjTest.RotateAngle = 90;
Application.Alert('Rotate ' + ObjTest.RotateAngle + ' degrees.');

ObjTest.RotateAngle = 360;
Application.Alert('Rotate ' + ObjTest.RotateAngle + ' degrees.');
```


Scrollbars

Controls whether or not the view is scrollable using the Scrollbars property. When set to 1 (Yes - the default), scrollbars appear as needed on the left and bottom of the view. Turn scrollbars off by setting this property to 0 (No). This property is implemented only for use at design time.

Applies to

- TheFrame
- TheView

Access

Read-write

Returns

Long

Usage in JScript

```
TheView.Scrollbars = long;
```

Usage in VBScript

```
TheView.Scrollbars = long
```

Example

```
// This code will adjust the setting of the scroll bars.
var showScrollOnNeed = true;
var sWidth = 900;
var sHeight = 900;

TheView.ScrollBars = showScrollOnNeed;
TheView.ScrollHeight = sHeight;
TheView.ScrollWidth = sWidth;
```

ScrollHeight

Controls how much to scroll, if scrollbars are needed. Set this value to the number of pixels in the height of the view. This property is implemented only for use at design time.

Applies to

- TheFrame
- TheView

Access

Read-write

Returns

Long

Usage in JScript

```
TheView.ScrollHeight = long;
```

Usage in VBScript

```
TheView.ScrollHeight = long
```

Example

```
// This code will adjust the setting of the scroll bars.
var showScrollOnNeed = true;
var sWidth = 900;
var sHeight = 900;

TheView.ScrollBars = showScrollOnNeed;
TheView.ScrollHeight = sHeight;
TheView.ScrollWidth = sWidth;
```

ScrollWidth

Controls how much to scroll, if scrollbars are needed. Set this value to the number of pixels in the width of the view. This property is implemented for use only at design time.

Applies to

- TheFrame
- TheView

Access

Read-write

Returns

Long

Usage in JScript

```
TheView.ScrollWidth = long;
```

Usage in VBScript

```
TheView.ScrollWidth = long
```

Example

```
// This code will adjust the setting of the scroll bars.
var showScrollOnNeed = true;
var sWidth = 900;
var sHeight = 900;

TheView.ScrollBars = showScrollOnNeed;
TheView.ScrollHeight = sHeight;
TheView.ScrollWidth = sWidth;
```


Selection

Indicates the index of the selection in combo boxes and list boxes. For items selected in combo boxes and list boxes, you can read and set the selected item. The number of the selected item corresponds to the index number for the item. If you attempt to set this number higher than the number of items in the combo box or list box, the operation fails without reporting an error.

Applies to

- ComboBox
- ListBox

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Selection = long;
```

Usage in VBScript

```
ObjTest.Selection = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of selection to a value below the number of items present.

ObjTest.AddString('Brandon Stewart');
ObjTest.AddString('Zebra Farm');
ObjTest.AddString('Abraham Simpson');

ObjTest.Selection = 2; //Third item is selected.

// Retrieve the selection.
Application.Alert('Selection is made on item ' + ( ObjTest.Selection + 1
 )+ '.');
```

ShadowStyle

Sets the position of the shadow for control. By giving controls a shadow, you can improve the appearance of the controls; they will have a 3D lighting effect.

Applies to

- Arc
- Bitmap
- Border
- Connector
- Ellipse
- Freehand
- Line

- Polyline
- Text

Note: Although available to all the above objects, this property affects only the Border, Ellipse, and ShadowStyle objects.

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.ShadowStyle = long;
```

The valid values are:

- 0 = None
- 1 = Top/Left
- 2 = Bottom/Right (This is the standard lighting effect for Windows.)
- 3 = Bottom/Left
- 4 = Top/Right

Usage in VBScript

```
ObjTest.ShadowStyle = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.

ObjTest.ShadowStyle = 0; //0 - None
Application.Alert('No shadow selected.');

ObjTest.ShadowStyle = 1; //1 - Top/Left
Application.Alert('Shadow on top-left side.');

ObjTest.ShadowStyle = 2; //2 - Bottom/Right
Application.Alert('Shadow on bottom-right side.');

ObjTest.ShadowStyle = 3; //3 - Bottom/Left
Application.Alert('Shadow on bottom-left side.');

ObjTest.ShadowStyle = 4; //4 - Top/Right
Application.Alert('Shadow on top-right side.');
```

Sort

Indicates whether or not to sort the lines of text added to a list box with the AddString method during run time, or to the lines of text added to a combo box during design time. For the ComboBox object, the Sort property is design-time only.

Applies to

- ComboBox
- ListBox

Access

Read-write

Returns

ListBox: Long value, either 0 (does not sort) or 2 (does sort).

ComboBox: No return value.

Usage in JScript

```
ObjTest.Sort = long;
```

Usage in VBScript

```
ObjTest.Sort = long
```

Example

```
// Add some items to ObjTest and then sort them
// This script must be put into the OnInitialize event
function OnInitialize {
ObjTest.AddString('Brandon Stewart');
ObjTest.AddString('Zebra Farm');
ObjTest.AddString('Abraham Simpson');
ObjTest.Sort = 2;
}
```

Source

Indicates the database table and fields for the Datasource object. Design time only.

Applies to

- DataSource

Access

Read-write

Returns

N/A

SpacerWidth

Indicates the number of pixels to appear between the buttons (tabs) at the top of a page. Can only be read or set at run time.

Applies to

- Button

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.SpacerWidth = long;
```

Usage in VBScript

```
ObjTest.SpacerWidth = long
```

Example

```
// Display then change the spacer width for a button
// with the name ObjTest
Application.Alert ('SpacerWidth = ' + ObjTest.SpacerWidth);
ObjTest.SpacerWidth = 2;
Application.Alert ('SpacerWidth now = ' + ObjTest.SpacerWidth);
```

StartAngle

Indicates the starting angle in degrees to start drawing the arc. You can only set and get this property at run time. If you want to set angles at design time, use the Angle1 property.

Applies to

- Arc

Access

Read-write

Returns

Long

Usage in JScript

```
 ObjTest.StartAngle = long;
```

A StartAngle of 0 indicates the point directly to the right of the centerpoint of the circle this arc is part of. A StartAngle of 180 indicates the point directly to the left of the centerpoint. Use this property with EndAngle to determine the shape of the arc. The range is -179 degrees to 180 degrees. Arcs are drawn counter-clockwise, starting from StartAngle and ending at EndAngle.

Usage in VBScript

```
ObjTest.StartAngle = long
```


Example

```
// Name of the arc that contains the property is ObjTest.
// Arc spans from startAngle to endAngle.

Application.Alert( 'Start: ' +ObjTest.startangle + ' | End: ' +
ObjTest.endAngle);

ObjTest.startAngle = 10;
ObjTest.endAngle = 20;

// Retrieve the value of endAngle
Application.Alert( 'Start: ' +ObjTest.startangle + ' | End: ' +
ObjTest.endAngle);
```

Tabstop

Indicates whether or not the selected control is a tab stop. You can set this to 0 (No) or 1 (Yes) at design time, or you can set this property to 0 at run time. You cannot enable a control's tab stop property at run time, only disable it.

Applies to

- ActiveX
- Button
- CheckButton
- ComboBox
- EditBox
- ListBox
- MultiEditBox
- RadioButton

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.TabStop = long;
```

Either 0 for disabled, or 65536 for enabled for the selected control.

Usage in VBScript

```
ObjTest.TabStop = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// You can only disable the object.
ObjTest.TabStop = 0; //Tabstop is disabled.
ObjTest.TabStop = 1; //Tabstop will still be disabled.
```

Tag

Stores user-defined data. You can set this value at design time or run time and it can contain anything allowable as a variant data type.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

Variant

Usage in JScript

```
ObjTest.Tag = variant;
```

Usage in VBScript

```
ObjTest.Tag = variant
```

Example

```
// Display then change the tag for an object
// with the name ObjTest
Application.Alert ('Tag = ' + ObjTest.Tag);
ObjTest.Tag = 'Here is some text';
Application.Alert ('Tag now = ' + ObjTest.Tag);
```


Text

In most cases, contains the string used to display the selected control.

Applies to

- Button
- CheckButton
- Connector
- EditBox
- Frame
- ListBox
- MultiEditBox
- Text

Note: This property does not apply to the ListBox object at run time.

Access

Read-write

Returns

String

Description

If you are setting the Text property for a multi-edit box, and want to add several lines of text, separate each line with a `Chr(10)` character. For example:

```
myMultiEditBox.Text = "First line" + CHR(10) + "Second line" + CHR(10) + "Third line"
```

Usage in JScript

```
ObjTest.Text = string;
```

Usage in VBScript

```
ObjTest.Text = string
```

Example

```
// Name of the object that contains the property/method is ObjTest.
Application.Alert('Text: ' + ObjTest.Text);

//Now, change the values of Text property
ObjTest.Text = 'First change.';
Application.Alert('Text: ' + ObjTest.Text);

ObjTest.Text = 'Another change.';
Application.Alert('Text: ' + ObjTest.Text);
```

Title

Indicates the text that appears in the frame's title bar. You can set it at design time or dynamically at run time.

Applies to

- TheFrame

Access

Read-write

Returns

String

Usage in JScript

```
TheFrame.Title = string;
```

Usage in VBScript

```
TheFrame.Title = string
```

Example

```
// This code will adjust the setting of the title bar.
var showMinBox = true;
var showMaxBox = false;

TheFrame.Title = 'This is a sample title';
TheFrame.MaximizeBox = showMaxBox;
TheFrame.MinimizeBox = showMinBox;
```

ToolTipText

Sets tooltip text. Can be set at design time or run time.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton

- Text

Access

Read-write

Returns

String

Usage in JScript

```
ObjTest.ToolTipText = string;
```

Usage in VBScript

```
ObjTest.ToolTipText = string
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of ToolTipText to an arbitrary value.
ObjTest.ToolTipText = 'This is a tool tip';

// Retrieve the value of ToolTipText
var msg = 'ToolTipText: ' +ObjTest.ToolTipText;
msg += '. Hover over the object to confirm';
Application.Alert(msg);
```

Top

Indicates the number of pixels from the top of the form that the selected object's bottom is drawn. Although this property can be set for objects that are not visible, no change to the display of the form is made. Setting this property to be smaller than the Top property renders the selected control invisible.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline

- RadioButton
- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Top = 100;
```

Usage in VBScript

```
ObjTest.Top = 100
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Retrieve the value of Bottom
Application.Alert('Top: ' +ObjTest.Top + ' | ' + 'Bottom: '
+ObjTest.Bottom);

var height = 100;
ObjTest.Top = 10;
ObjTest.Bottom = ObjTest.Top + height;

// Retrieve the value of Bottom
Application.Alert('Top: ' +ObjTest.Top + ' | ' + 'Bottom: '
+ObjTest.Bottom);
```

TransparentColor

Indicates the color within the Bitmap that is transparent. For example, if you have set green, RGB value of (0,255,0), as the transparent color, all areas of the bitmap that are green do not display as green, but as the color set in the BackColor property.

Applies to

- Bitmap

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.TransparentColor = long;
```


Usage in VBScript

```
ObjTest.TransparentColor = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of TransparentColor to a integer representation of RGB
color.
var rgbColor = 255; //Red - Change this value to select different color.

ObjTest.TransparentColor = rgbColor;
Application.Alert('Transparent Color is ' + ObjTest.TransparentColor);
```

TriState

Determines whether a check button can be in a tri-state. When the TriState property is set to 1 (Yes), the check button can be in a True state, a False state, or an Ambiguous (grayed-out) state. If this property is set to 0 (No), the check button can only be in a True or False state. This property can be set only at design time.

Applies to

- CheckButton

Access

Read only

Returns

N/A

UseColors

Determines whether the colors set in the BackColor and ForeColor properties are used.

Applies to

- ComboBox
- EditBox
- ListBox
- MultiEditBox

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.UseColors = long;
```

Possible values are:

- 0 = do not use the BackColor and ForeColor; Windows default colors are used

- 1 = use the BackColor and ForeColor

Usage in VBScript

```
ObjTest.UseColors = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Set the value of UseColors to either 0 (no) or 1 (yes).

ObjTest.ForeColor = 16711680; //Set to blue.
ObjTest.BackColor = 255; //solid red

ObjTest.UseColors = 1; //set to use colors
Application.Alert( 'Object is currently set to use colors.' );

ObjTest.UseColors = 0; //set to use colors
Application.Alert( 'Object is currently set not to use colors.' );
```

UseTabStops

Enables each line of a list box to be a tab stop. Returns either 0, for UseTabStops set to 0 (No), or 128, for UseTabStops set to 1 (Yes). This property is implemented for use at design time only.

Applies to

- ListBox

Access

Read-write

Returns

Long

Value

Returns the index of the selected radio button, check button, or selection in a combo box.

Applies to

- CheckButton
- ComboBox
- RadioButton

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Value = long;
```


Valid values are from 0 (the first item in the list) to the maximum number -1. Attempts to set the value to a number higher than the index number results in no action performed and no error reported. Setting the Value property to -1 for the combo box deselects all lines.

Usage in VBScript

```
ObjTest.Value = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Make sure at least three items are present in the object.
ObjTest.Value = 1;
Application.Alert('Second item is selected');

ObjTest.Value = 2;
Application.Alert('Third item is selected');

ObjTest.Value = -1;
Application.Alert('No item is selected');

ObjTest.Value = 0;
Application.Alert('First item is selected');
```

ValueID

Not implemented.

Applies to

- EditBox
- MultiEditBox

VerticalScroll

Sets the state of the vertical scroll bar in list boxes or multi-edit boxes. Although accessible, this property is not implemented to be set at run-time.

Applies to

- ListBox
- MultiEditBox

Access

Read-write

Returns

Long

Description

Usage in JScript

```
ObjTest.VerticalScroll = long;
```

If the property is set to 0 (None), text that extends below the bottom of the interior of the control is truncated to the maximum allowable set by the viewing area. In this instance, no vertical scroll bar is drawn, and the height of the text entered into the control is always the same or less than the height of the control interior.

If the property is set to 1 (Always), the vertical scroll bar is always drawn, and the text entered by the user either wraps to the next line, or a new line is created when the user presses Enter. The text continues past the bottom of the control. In this instance, the user can use the scroll bar, which is always visible, to scroll up and down through the entered text.

If the property is set to 2 (Automatic), the scroll bar is drawn at the time that the text extends past the bottom of the interior of the control. When entered text wraps to the next line or the user presses Enter enough times, it continues past the bottom of the control. Once the user has entered text that is 'taller' than the control, the scroll bar appears and can be used to scroll through up and down through the text

Usage in VBScript

```
ObjTest.VerticalScroll = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
ObjTest.VerticalScroll = 0; //0 - None
Application.Alert('Vertical bar is disabled.');

ObjTest.VerticalScroll = 1; //1 - Always
Application.Alert('Vertical bar will always appear.');

ObjTest.VerticalScroll = 2; //2 - Automatic
Application.Alert('Vertical bar will appear as needed.');
```

ViewLayers

Opens the Layers Sheet dialog to create and edit the view layers. This property is available only at design time.

Applies to

- TheFrame
- TheView

Access

Read-write

Returns

N/A

Definition of layers (there is one layer named Default by default)

Description

The Layers Sheet dialog allows the user to add or delete view layers (except the default layer). An individual layer can be moved up or down in the list, but this has no effect on the drawing of the view (objects are always drawn according to their Z-order in the Object Sheet dialog).The user can select all of the objects on a particular layer.

The current Insert Layer determines which layer new objects are inserted into. An individual layer's view (visible/invisible) and lock (locked/unlocked) states can be set. The user can click the check boxes in the areas to the left of the layer name to toggle these states. Double-clicking on the eye and lock pictures next to the check boxes toggles the view and lock states for all of the layers.

Visible

Sets a control to be visible or invisible.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.Visible = long;
```

When set to 0 (No), the control is not visible, and when set to 1 (Yes), the control is visible. When setting this property in script, use True to set the control to be visible, and False to set the control to be invisible.

Usage in VBScript

```
ObjTest.Visible = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
ObjTest.Visible = 0;
Application.Alert('Object is invisible.');

ObjTest.Visible = 1;
Application.Alert('Object is visible.');
```

WantKeyInput

Determines whether focus can be set on a list box. You can set this property at run time to be 0 (No); however, you cannot set this property to 1024 (Yes) at run time.

Applies to

- ListBox

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.WantKeyInput = long;
```

Valid values are

- 1024 (Yes)
- 0 (No)

Usage in VBScript

```
ObjTest.WantKeyInput = long
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// You can only disable the object.
ObjTest.WantKeyInput = 0; //KeyInput is disabled.
ObjTest.WantKeyInput = 1; //KeyInput will still be disabled.
```

WhatsThisHelp

Indicates whether to enable field-level context-sensitive help. If set to 1 (Yes), each object's help context ID value is used to display the associated help topic.

Applies to

- TheFrame

Access

Read-write

Returns

Long

Usage in JScript

```
TheFrame.WhatsThisHelp = long;
```

Valid values are

- 0 (No)
- 1 (Yes)

Usage in VBScript

```
TheFrame.WhatsThisHelp = long
```

Example

```
// Set the value below to either true or false to observe the change.
TheFrame.WhatsThisHelp = true; //this will enabled help.
```

Width

Sets the width of the form, in pixels. Use the Width property with the TheFrame and TheView objects to set the width of the form at design time. You can also modify the form width using the mouse, which updates the Width property automatically.

Applies to

- TheFrame
- TheView

Access

Read-write

Returns

Long

Usage in JScript

```
TheFrame.Width = long;
```

Usage in VBScript

```
TheFrame.Width = long
```

Example

```
// Execute this code multiple times to observe the incremental effect.
// This code increments the size of the form by the amount specified in
variables below.
var xWidth= 10;
var yWidth= 20;

TheFrame.Width += xWidth;
TheFrame.Height += yWidth;
```

X1

Indicates the distance in pixels from the left side of the form the starting point of the selected line. Use this property with X2, Y1, and Y2.

Applies to

- Line

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.x1=long;
```

Usage in VBScript

```
ObjTest.x1=long
```

Example

```
// Name of the line that contains the property/method is ObjTest.
// Following code will move the line from current coordinate by the amount
 specified below.
// change these values as needed
var cx1 = 10;
var cy1 = 10;
var cx2 = 50;
var cy2 = 100;

ObjTest.SetEndPoints(ObjTest.x1 + cx1, ObjTest.y1 + cy1, ObjTest.x2 + cx2,
 ObjTest.y2 + cy2);
```

X2

Indicates the distance in pixels from the left side of the form the ending point of the selected line. Use this property with X1, Y1, and Y2.

Applies to

- Line

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.x2=long;
```

Usage in VBScript

```
ObjTest.x2=long
```

Example

```
// Name of the line that contains the property/method is ObjTest.
// Following code will move the line from current coordinate by the amount
 specified below.
// change these values as needed
var cx1 = 10;
var cy1 = 10;
var cx2 = 50;
var cy2 = 100;

ObjTest.SetEndPoints(ObjTest.x1 + cx1, ObjTest.y1 + cy1, ObjTest.x2 + cx2,
 ObjTest.y2 + cy2);
```

XmlNodeRules

Not implemented.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

XmlValue

See the section XFT content mapping model for information on using this property.

Applies to

- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

XPath

See the section of the **Form-Creation Tutorials**, Create a form bound to XML content for information on using this property.

Applies to

- ActiveX
- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- DataSource
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton

- Text

Access

Read only

Returns

N/A

Y1

Indicates the distance in pixels from the top of the form the starting point of the selected line. Use this property with X1, X2, and Y2.

Applies to

- Line

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.y1=long;
```

Usage in VBScript

```
ObjTest.y1=long
```

Example

```
// Name of the line that contains the property/method is ObjTest.
// Following code will move the line from current coordinate by the amount
 specified below.
// change these values as needed
var cx1 = 10;
var cy1 = 10;
var cx2 = 50;
var cy2 = 100;

ObjTest.SetEndPoints(ObjTest.x1 + cx1, ObjTest.y1 + cy1, ObjTest.x2 + cx2,
 ObjTest.y2 + cy2);
```

Y2

Indicates the distance in pixels from the top of the form the ending point of the selected line. Use this property with X1, X2, and Y1.

Applies to

- Line

Access

Read-write

Returns

Long

Usage in JScript

```
ObjTest.y2=long;
```

Usage in VBScript

```
ObjTest.y2=long
```

Example

```
// Name of the line that contains the property/method is ObjTest.
// Following code will move the line from current coordinate by the amount
 specified below.
// change these values as needed
var cx1 = 10;
var cy1 = 10;
var cx2 = 50;
var cy2 = 100;

ObjTest.SetEndPoints(ObjTest.x1 + cx1, ObjTest.y1 + cy1, ObjTest.x2 + cx2,
 ObjTest.y2 + cy2);
```

XFT methods

The following code samples include an object called ‘ObjTest’. This name must correspond to an object that already exists on the form.

AddItem

Adds items to a combo box list dynamically at run time. The associated value can be returned in script, but is not visible to the user as part of the combo box. Items added to the combo box are appended to the end of the list.

Applies to

- ComboBox

Returns

No return value

Parameters

strItem - The string to add to the combo box

longValue - The value associated with the Item added to the combo box

Usage in JScript

```
ObjTest.AddItem(string, long);
```

Usage in VBScript

```
ObjTest.AddItem(string, long)
```

Example

```
// Name of the ComboBox that contains the property/method is ObjTest.
ObjTest.AddItem('Zebra', 3);
ObjTest.AddItem('Brad', 5);
ObjTest.AddItem('Abraham', 2);
```

AddString

Adds items to a combo box or list box list dynamically at run time. Added strings are appended to the end of the list.

Applies to

- ComboBox
- ListBox

Returns

Long

Parameters

strString - The string to add to the combo box or list box

Usage in JScript

```
ObjTest.AddString(string);
```

Usage in VBScript

```
ObjTest.AddString(string)
```

Example

```
// Name of the object that contains the property/method is ObjTest.
ObjTest.AddString('Brandon Stewart');
ObjTest.AddString('Zebra Farm');
ObjTest.AddString('Abraham Simpson');
ObjTest.Selection = 2; //Third item is selected.
```

Cancel

Cancels the view object. It has the functional equivalent of closing the form.

Applies to

- TheView

Returns

No return value

Parameters

None

Usage in JScript

```
TheView.Cancel();
```

Usage in VBScript

```
TheView.Cancel()
```

Example

```
//Warning: executing this code will close the document
Application.Alert('This code will cancel the form');
TheView.Cancel();
```

ClearDataAll

Not implemented.

Applies to

- TheFrame

Returns

No return value

Parameters

None

DeleteItem

Deletes items from a combo box list dynamically at run time. Remember that the index number will change for the items below the one you are deleting; therefore, be careful when removing multiple items from a list to remove them in ascending order from the bottom of the list to the top. DeleteItem fails if you try to use it to delete a currently selected item.

Applies to

- ComboBox

Returns

No return value

Parameters

longIndex - The index number of the item to delete from the combo box

Usage in JScript

```
ObjTest.DeleteItem(long);
```

Usage in VBScript

```
ObjTest.DeleteItem(long)
```

Example

```
// Name of the ComboBox that contains the property/method is ObjTest.
// Make sure there are at least 3 itesm in the givn object to test this
properly/

// Delete first three items
ObjTest.DeleteItem(0);
ObjTest.DeleteItem(0);
ObjTest.DeleteItem(0);
```

DeleteString

Deletes items from a combo box or list box list dynamically at run time. Remember that the Index number will change for the items below the one you are deleting; therefore, be careful when removing multiple strings from a list to remove them in ascending order from the bottom of the list to the top.

Applies to

- ComboBox
- ListBox

Returns

Long

Parameters

longIndex - The index number of the item to delete from the combo box or list box

Usage in JScript

```
ObjTest.DeleteString(long);
```

Usage in VBScript

```
ObjTest.DeleteString(long)
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Make sure there are at least 3 items in the given object to test this
 properly

// Delete first three items, delete all if there are less than three items
 present

var deleteCount = 3;
var textToDelete;
if (ObjTest.GetNumItems() < deleteCount) deleteCount =
ObjTest.GetNumItems();
```

```
Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');
Application.Alert(deleteCount + ' items to delete.');

for(i=0; i < deleteCount; i++){
textToDelete = ObjTest.GetText(0);
Application.Alert('Deleting: ' + textToDelete);
ObjTest.DeleteString(0);
}

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');

// Following code empties the list. If code above has emptied the list
already,
// no change will occur. However, if there were more then three items in
 the List,
// following code will erase them.
ObjTest.ResetContent();

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');
```

EditItem

Replaces the string of an item in a combo box with a new string. When using this method, indicate the new string using strItem and the index number of the item to replace it with using longIndex.

Applies to

- ComboBox

Returns

No return value.

Parameters

strItem - The new string for the item being edited

longIndex - The index number of the item being edited

Usage in JScript

```
ObjTest.EditItem(string, long);
```

Usage in VBScript

```
ObjTest.EditItem(string, long)
```

Example

```
// Name of the ComboBox that contains the property/method is ObjTest.

// Edit fourth, sixth, and third items, adding them if they are not found.
ObjTest.EditItem('Frank', 3);
ObjTest.EditItem('Edited', 5);
ObjTest.EditItem('Joseph', 2);
```


FormFind

Returns another form as an object. Once returning this form, you can work with it and its properties and methods. The parameter used to 'find' the form, strFormName, must be the 'friendly' name assigned in the form's (FormCode) property

Applies to

- TheView

Returns

A Form object

Parameters

strFormName - The name of the form object you want to find and work with.

Usage in JScript

```
var newForm = TheView.FormFind(string);
```

Usage in VBScript

```
newForm = TheView.FormFind(string)
```

Example

```
var formName = '123'; //Enter a valid form name here
var formStatus = TheView.IsFormOpen(formName);
Application.Alert('Is form open? ' + formStatus);
if (formStatus) {
var newForm = TheView.FormFind(formName);
Application.Alert(newForm.TheView.Title);
}
```

GetCount

Returns the number of items in the selected list box control.

Applies to

- ListBox

Returns

Long

Parameters

None

Usage in JScript

```
var itemCount = ObjTest.getCount();
```

Usage in VBScript

```
itemCount = ObjTest.getCount()
```

Example

```
// Name of the ListBox that contains the property/method is ObjTest.

// Make sure you have at least one item in the ListBox to execute the
following code
// This code cycles through the lines in ListBox and inserts new line at
 that line
Application.Alert('There are ' + ObjTest.getCount() + ' items in the
ListBox.');

Application.Alert('Currently item ' + (ObjTest.getCurSel() + 1) + ' is
selected in the ListBox.');

var nextLineSel = (ObjTest.getCurSel() + 1) % ObjTest.getCount();
ObjTest.SetCurSel(nextLineSel); //Select the next line, by cycling through
 lines.
Application.Alert('Content of line ' + nextLineSel + ': ' +
ObjTest.Value());
//Insert new line at that selection
ObjTest.InsertString(nextLineSel, 'This was inserted at line ' +
(nextLineSel + 1));
```

GetCurSel

Returns the index number of the item selected by the user in a list box control. If no item is selected in the list box, this method returns -1.

Applies to

- ListBox

Returns

Long

Parameters

None

Usage in JScript

```
var Sel = ObjTest.SetCurSel(long);
```

Usage in VBScript

```
Sel = ObjTest.SetCurSel(long)
```

Example

```
// Name of the ListBox that contains the property/method is ObjTest.

// Make sure you have at least one item in the ListBox to execute the
following code
// This code cycles through the lines in ListBox and inserts new line at
 that line
Application.Alert('There are ' + ObjTest.getCount() + ' items in the
```



```
ListBox.');

Application.Alert('Currently item ' + (ObjTest.getCurSel() + 1) + ' is
selected in the ListBox.');

var nextLineSel = (ObjTest.getCurSel() + 1) % ObjTest.getCount();
ObjTest.SetCurSel(nextLineSel); //Select the next line, by cycling through
 lines.
Application.Alert('Content of line ' + nextLineSel + ': ' +
ObjTest.Value());
//Insert new line at that selection
ObjTest.InsertString(nextLineSel, 'This was inserted at line ' +
(nextLineSel + 1));
```

GetDlgCtrlID

Retrieves the value of the Windows Dialog Control ID for a control.

Applies to

- Button
- CheckButton
- ComboBox
- EditBox
- ListBox
- MultiEditBox
- RadioButton

Returns

Long

Parameters

None

Usage in JScript

```
var controlID = ObjTest.GetDlgCtrlID();
```

Usage in VBScript

```
controlID = ObjTest.GetDlgCtrlID()
```

Example

```
// Name of the MultiEditBox that contains the property/method is ObjTest.
var controlID = ObjTest.GetDlgCtrlID();
Application.Alert('Conrol ID for this obj is ' + controlID + '.');
```

GetLineCount

Returns the number of lines of text in a multi-edit box. Even if there is no text in the multi-edit box, a value of 1 is returned.

Applies to

- MultiEditBox

Returns

Long

Parameters

None

Usage in JScript

```
var lineCount = ObjTest.GetLineCount();
```

Usage in VBScript

```
lineCount = ObjTest.GetLineCount()
```

Example

```
// Name of the MultiEditBox that contains the property/method is ObjTest.
var msg = 'Number of Lines: ' + ObjTest.GetLineCount();
msg += String.fromCharCode(10) + 'Content in first Line:' ;
msg += ObjTest.GetLineOfText(0);
Application.Alert(msg);
```

GetLineOfText

Returns a string that contains a line of text from a mutli-edit box. Use longIndex to indicate which line you want the string for. If you indicate a value for longIndex that does not correspond to a line in the mutli-edit box, an empty string is returned.

Applies to

- MultiEditBox

Returns

String

Parameters

longIndex - The index number of the line to retrieve a string of.

Usage in JScript

```
var textLine = ObjTest.GetLineOfText(long);
```

Usage in VBScript

```
textLine = ObjTest.GetLineOfText(long)
```


Example

```
// Name of the MultiEditBox that contains the property/method is ObjTest.
var msg = 'Number of Lines: ' + ObjTest.GetLineCount();
msg += String.fromCharCode(10) + 'Content in first Line:' ;
msg += ObjTest.GetLineOfText(0);
Application.Alert(msg);
```

GetNumItems

Returns the number of items in a combo box or list box.

Applies to

- ComboBox
- ListBox

Returns

Long

Parameters

None

Usage in JScript

```
var numberOfItems = ObjTest.GetNumItems();
```

Usage in VBScript

```
numberOfItems = ObjTest.GetNumItems()
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Make sure there are at least 3 itesm in the givn object to test this
properly/

// Delete first three items, delete all if there are less than three items
 present

var deleteCount = 3;
var textToDelete;
if (ObjTest.GetNumItems() < deleteCount) deleteCount =
ObjTest.GetNumItems();

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');
Application.Alert(deleteCount + ' items to delete.');

for(i=0; i < deleteCount; i++){
textToDelete = ObjTest.GetText(0);
Application.Alert('Deleting: ' + textToDelete);
ObjTest.DeleteString(0);
}

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');

// Following code empties the list. If code above has emptied the list
```

```
already,
// no change will occur. However, if there were more then three items in
 the List,
// following code will erase them.
ObjTest.ResetContent();

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');
```

GetPointX

Returns the x-coordinate of a Connector, Freehand, or Polyline object. Use this method in conjunction with the GetPointY method.

Applies to

- Connector
- Freehand
- Polyline

Returns

Long value, representing the number of pixels from the left side of the form

Parameters

longIndex - The index number of the line segment you want to return the X coordinate for

Description

The Connector, Freehand, and Polyline objects consist of several line segments. Each line segment has an x- and y-coordinate denoting its starting point. The ending points of a line segment are the starting points of the next segment.

When you create an object with multiple line segments, several points are used, and each point has an index number. Therefore, if you want to return the x- and y-coordinates of a single line segment in an object that contains several line segments, you must indicate, using the parameter longIndex, which segment you want the coordinates for. The last line segment in the object has an index number of 0. The second-to-last line segment has an index of 1, and so on.

Usage in JScript

```
var Pnt = ObjTest.GetPointX(long);
```

Usage in VBScript

```
Pnt = ObjTest.GetPointX(long)
```

Example

```
// Name of the Object that contains the property/method is ObjTest.

// This code moves the object ObjTest right and down by a specified amount.
var Xdistance = 10;
var Ydistance = 5;
```



```
for (i = 0; i < ObjTest.NumberOfPoints(); i++){
Application.Alert('Old point X:' + ObjTest.GetPointX(i) + ', Y:' +
ObjTest.GetPointY(i));
ObjTest.SetPoint(i, ObjTest.GetPointX(i) + Xdistance, ObjTest.GetPointY(i)
 + Ydistance);
Application.Alert('New point X:' + ObjTest.GetPointX(i) + ', Y:' +
ObjTest.GetPointY(i));
}
```

GetPointY

Returns the y-coordinate for a Connector, Freehand, or Polyline object. Use this method in conjunction with the GetPointX method.

Applies to

- Connector
- Freehand
- Polyline

Returns

Long value, representing the number of pixels from the top of the form

Parameters

longIndex - The index number of the line segment you want to return the Y coordinate for

Description

The Connector, Freehand, and Polyline objects consist of several line segments. Each line segment has an x- and y-coordinate denoting its starting point. The ending points of a line segment are the starting points of the next segment.

When you create an object with multiple line segments, several points are used, and each point has an index number. Therefore, if you want to return the x- and y-coordinates of a single line segment in an object that contains several line segments, you must indicate, using the parameter longIndex, which segment you want the coordinates for. The last line segment in the object has an index number of 0. The second-to-last line segment has an index of 1, and so on.

Usage in JScript

```
var Pnt = ObjTest.GetPointY(long);
```

Usage in VBScript

```
Pnt = ObjTest.GetPointY(long)
```

Example

```
// Name of the Object that contains the property/method is ObjTest.

// This code moves the object ObjTest right and down by a specified amount.
var Xdistance = 10;
var Ydistance = 5;
```

```
for (i = 0; i < ObjTest.NumberOfPoints(); i++){
Application.Alert('Old point X:' + ObjTest.GetPointX(i) + ', Y:' +
ObjTest.GetPointY(i));
ObjTest.SetPoint(i, ObjTest.GetPointX(i) + Xdistance, ObjTest.GetPointY(i)
 + Ydistance);
Application.Alert('New point X:' + ObjTest.GetPointX(i) + ', Y:' +
ObjTest.GetPointY(i));
}
```

GetSelectedText

Retrieves the selected text in a multi-edit box. If multiple lines are selected (text separated by a carriage return/line feed) all lines are returned with the line break character intact.

Applies to

- MultiEditBox

Returns

String

Parameters

None

Usage in JScript

```
var selected = ObjTest.GetSelectedText();
```

Usage in VBScript

```
selected = ObjTest.GetSelectedText()
```

Example

```
// Name of the Object that contains the property/method is ObjTest.
// Select a portion of text that you wish to extract before executing this
 code.
var selected = ObjTest.GetSelectedText();
Application.Alert('You have selected {' + selected + '}');
```

GetText

Retrieves a string containing the text of an item in a combo box or list box. The longIndex parameter indicates the index number of the item. If you specify an index that does not exist, a blank string is returned.

Applies to

- ComboBox
- ListBox

Returns

String

Parameters

longIndex - The index number of the item to retrieve the text of

Usage in JScript

```
var text = ObjTest.GetText(long);
```

Usage in VBScript

```
text = ObjTest.GetText(long)
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Make sure there are at least 3 itesm in the givn object to test this
properly/

// Delete first three items, delete all if there are less than three items
 present

var deleteCount = 3;
var textToDelete;
if (ObjTest.GetNumItems() < deleteCount) deleteCount =
ObjTest.GetNumItems();

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');
Application.Alert(deleteCount + ' items to delete.');

for(i=0; i < deleteCount; i++){
textToDelete = ObjTest.GetText(0);
Application.Alert('Deleting: ' + textToDelete);
ObjTest.DeleteString(0);
}

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');

// Following code empties the list. If code above has emptied the list
already,
// no change will occur. However, if there were more then three items in
 the List,
// following code will erase them.
ObjTest.ResetContent();

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');
```

GetWindowText

Retrieves the text displayed in an edit box or multi-edit box. If there are multiple lines (text separated by a carriage return/line feed) all lines are returned with the line break character intact.

Applies to

- EditBox
- MultiEditBox

Returns

String

Parameters

None

Usage in JScript

```
var Text = ObjTest.GetWindowText();
```

Usage in VBScript

```
Text = ObjTest.GetWindowText();
```

Example

```
// Name of the Object that contains the property/method is ObjTest.

// This code retrieves the text and set the modified version of it
var originalText = ObjTest.GetWindowText();
ObjTest.SetWindowText('{Modified: ' + originalText + ' }');
```

GUIEvent

Updates the current view.

Applies to

- TheView

Returns

No return value

Parameters

None

Usage in JScript

```
TheFrame.GUIEvent();
```

Usage in VBScript

```
TheFrame.GUIEvent()
```

Example

```
// Update the current view
TheView.GUIEvent();
```

GUIEventAll

Updates all views contained by TheFrame.

Applies to

- TheFrame

Returns

No return value

Parameters

None

Usage in JScript

```
TheFrame.GUIEventAll();
```

Usage in VBScript

```
TheFrame.GUIEventAll()
```

Example

```
// Update all views in TheFrame
TheFrame.GUIEventAll();
```

IncrValue

Changes the state of a checkbutton control.

Applies to

- CheckButton

Returns

No return value

Parameters

None

Description

The state that the control changes to depends on whether or not the TriState property has been set to 0 (No) or 1 (Yes).

With TriState set to 0, this method toggles between two states:

- Unchecked
- Checked

With TriState set to 1, this method cycles between three states:

- Unchecked
- Checked grayed
- Checked

Usage in JScript

```
ObjTest.IncrValue();
```

Usage in VBScript

```
ObjTest.IncrValue()
```

Example

```
// Name of the Object that contains the property/method is ObjTest.

// Execute this code multiple times to cycle through all available states
 of check-button.
ObjTest.IncrValue();
```

InsertString

Inserts new lines of text into a list box control. Use the longIndex parameter to indicate which line to insert the text to (0 represents the first line, 1 represents the second line and so on), and strText to indicate the text to insert into that line. The existing text for the line, and all subsequent lines will have their index number incremented by one. Specifying an index number for a line that does not exist results in no effect on the list box, with no returned error.

Applies to

- ListBox

Returns

Long

Parameters

longIndex - The index number representing the line to insert text to

strText - The text to insert

Usage in JScript

```
var Ins = ObjTest.InsertString(long, string);
```

Usage in VBScript

```
Ins = ObjTest.InsertString(long, string)
```

Example

```
// Name of the ListBox that contains the property/method is ObjTest.

// Make sure you have at least one item in the ListBox to execute the
following code
// This code cycles through the lines in ListBox and inserts new line at
 that line
Application.Alert('There are ' + ObjTest.getCount() + ' items in the
ListBox.');

Application.Alert('Currently item ' + (ObjTest.getCurSel() + 1) + ' is
selected in the ListBox.');

var nextLineSel = (ObjTest.getCurSel() + 1) % ObjTest.getCount();
ObjTest.SetCurSel(nextLineSel); //Select the next line, by cycling through
 lines.
Application.Alert('Content of line ' + nextLineSel + ': ' +
```



```
ObjTest.Value());
//Insert new line at that selection
ObjTest.InsertString(nextLineSel, 'This was inserted at line ' +
(nextLineSel + 1));
```

IsFormOpen

Queries whether a form is open.

Applies to

- TheView

Returns

Boolean

Parameters

strFormName - The name of the form being tested (set in the (FormCode) property).

Usage in JScript

```
var formStatus = TheView.IsFormOpen(string);
```

Usage in VBScript

```
var formStatus = TheView.IsFormOpen(string)
```

Example

```
// Check to see if a form is open and display its title
var formName = '123'; //Enter a valid form name here
var formStatus = TheView.IsFormOpen(formName);
Application.Alert('Is form open? ' + formStatus);
if (formStatus) {
var newForm = TheView.FormFind(formName);
Application.Alert(newForm.TheView.Title);
}
```

LayerName

Returns the text description of the layer created using the Layer Sheet dialog and set using the Layer property.

Applies to

- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- EditBox
- Ellipse
- Frame

- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Returns

String

Parameters

None

Usage in JScript

```
var lName = ObjTest.LayerName();
```

Usage in VBScript

```
lName = ObjTest.LayerName()
```

Example

```
// Name of the Object that contains the property/method is ObjTest.
// Following code will display the name of the layer that the corresponding
 object resides under.
var lName = ObjTest.LayerName();
Application.Alert('The object is currently under ' + lName + ' layer.');
```

Move

Moves a control dynamically at run time. The parameters longChangeX and longChangeY indicate the number of pixels to move the control, and both parameters can be negative or positive. Coordinates are calculated based on the number of pixels from the left side of the form and the top of the form, which means that negative values indicate a move to the left (for longChangeX) or up (for longChangeY).

Applies to

- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite

- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Returns

No return value

Parameters

longChangeX - The number of pixels to move the control’s x-coordinates

longChangeY - The number of pixels to move the control's y-coordinates

Usage in JScript

```
ObjTest.Move(long, long);
```

Usage in VBScript

```
ObjTest.Move(long, long)
```

Example

```
// Name of the Object that contains the property/method is ObjTest.
// change these values as needed
var xDistance = 10;
var yDistance = 20;

ObjTest.Move(xDistance, yDistance);
```

NumberOfPoints

Returns the number of points used to draw a Connector, Freehand, or Polyline object. Use this method in conjunction with the GetPointX and GetPointY methods.

Applies to

- Connector
- Freehand
- Polyline

Returns

Long

Parameters

None

Description

The three objects, Connector, Freehand, and Polyline, are all made up of several line segments. Each line segment has an x- and y-coordinate denoting its starting point. The ending points of a line segment are the starting points of the next segment.

When you create an object with multiple line segments, several points are used to create the object. Use the method NumberOfPoints to determine the number of points used to draw the object.

Usage in JScript

```
var numPoints = ObjTest.NumberOfPoints();
```

Usage in VBScript

```
numPoints = ObjTest.NumberOfPoints()
```

Example

```
// Name of the Object that contains the property/method is ObjTest.

// This code moves the object ObjTest right and down by a specified amount.
var Xdistance = 10;
var Ydistance = 5;

for (i = 0; i < ObjTest.NumberOfPoints(); i++){
Application.Alert('Old point X:' + ObjTest.GetPointX(i) + ', Y:' +
ObjTest.GetPointY(i));
ObjTest.SetPoint(i, ObjTest.GetPointX(i) + Xdistance, ObjTest.GetPointY(i)
 + Ydistance);
Application.Alert('New point X:' + ObjTest.GetPointX(i) + ', Y:' +
ObjTest.GetPointY(i));
}
```

OK

Performs the same action as if a user had clicked the OK button on the form. It is the same as an ButtonType of OK.

Applies to

- TheView

Returns

No return value

Parameters

None

Usage in JScript

```
TheView.OK();
```

Usage in VBScript

```
TheView.OK()
```


Example

```
//Warning: executing this code will close the document
Application.Alert('This code will OK the form');
TheView.OK();
```

ResetContent

Clears all items from a combo box or list box.

Applies to

- ComboBox
- ListBox

Returns

No return value

Parameters

None

Usage in JScript

```
ObjTest.ResetContent();
```

Usage in VBScript

```
ObjTest.ResetContent()
```

Example

```
// Name of the object that contains the property/method is ObjTest.
// Make sure there are at least 3 itesm in the givn object to test this
properly/

// Delete first three items, delete all if there are less than three items
 present

var deleteCount = 3;
var textToDelete;
if (ObjTest.GetNumItems() < deleteCount) deleteCount =
ObjTest.GetNumItems();

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');
Application.Alert(deleteCount + ' items to delete.');

for(i=0; i < deleteCount; i++){
textToDelete = ObjTest.GetText(0);
Application.Alert('Deleting: ' + textToDelete);
ObjTest.DeleteString(0);
}

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');

// Following code empties the list. If code above has emptied the list
already,
// no change will occur. However, if there were more then three items in
```

```
 the List,
// following code will erase them.

ObjTest.ResetContent();

Application.Alert('There are ' + ObjTest.GetNumItems() + ' items left in
 the object.');
```

RunChange

Calls the OnChange event for the specified control. This method does not perform any changes to the state or content of the control. Instead, it runs the event for when the control is changed.

Applies to

- CheckButton
- ComboBox
- EditBox
- ListBox
- MultiEditBox
- RadioButton

Returns

No return value

Parameters

None

Usage in JScript

```
ObjTest.RunChange();
```

Usage in VBScript

```
ObjTest.RunChange()
```

Example

```
// Name of the Object that contains the property/method is ObjTest.
// Following will trigger the event method called onChange() in the
corresponding object.
ObjTest.RunChange();
```

RunClick

Calls the OnClick event for the specified control.

Applies to

- Arc
- Bitmap
- Border
- Button

- Connector
- Ellipse
- Freehand
- Line
- Polyline
- Text

Returns

No return value

Parameters

None

Usage in JScript

```
ObjTest.RunClick();
```

Usage in VBScript

```
ObjTest.RunClick()
```

Example

```
// Name of the Object that contains the property/method is ObjTest.
// Folowing will trigger the event method called onClick() in the
corresponding object.
ObjTest.RunClick();
```

RunInitialize

Calls the OnInitialize event for the specified control.

Applies to

- Arc
- Bitmap
- Border
- Button
- CheckButton
- ComboBox
- Connector
- EditBox
- Ellipse
- Frame
- Freehand
- Hilite
- Line
- ListBox
- MultiEditBox
- Polyline
- RadioButton
- Text

Returns

No return value

Parameters

None

Usage in JScript

```
ObjTest.RunInitialize();
```

Usage in VBScript

```
ObjTest.RunInitialize()
```

Example

```
// Name of the Object that contains the property/method is ObjTest.
// Following will trigger the event method called onInitialize() in the
corresponding object.
ObjTest.RunInitialize();
```

RunTerminate

Calls the OnTerminate event without closing the form or the view. Normally, the OnTerminate event fires only when exiting the view; however, you can use this method to fire the event at any time.

Applies to

- TheView

Returns

No return value

Parameters

None

Usage in JScript

```
ObjTest.RunTerminate();
```

Usage in VBScript

```
ObjTest.RunTerminate()
```

Example

```
// Following will trigger the event method called onTerminate() in the
view object.
TheView.RunTerminate();
```


SetCurSel

Sets the selected line using the SetCurSel method. The parameter longIndex indicates which line to select (0 is the first line, 1 is the second, and so on). If you attempt to set the selection to a line larger than the index number, no line is selected and no error is returned.

Applies to

- ListBox

Returns

Long

Parameters

longIndex - The index number of the line to set the selection to

Usage in JScript

```
var longSel = ObjTest.SetCurSel(long);
```

Usage in VBScript

```
longSel = ObjTest.SetCurSel(long)
```

Example

```
// Name of the ListBox that contains the property/method is ObjTest.

// Make sure you have at least one item in the ListBox to execute the
following code
// This code cycles through the lines in ListBox and inserts new line at
 that line
Application.Alert('There are ' + ObjTest.getCount() + ' items in the
ListBox.');

Application.Alert('Currently item ' + (ObjTest.getCurSel() + 1) + ' is
selected in the ListBox.');

var nextLineSel = (ObjTest.getCurSel() + 1) % ObjTest.getCount();
ObjTest.SetCurSel(nextLineSel); //Select the next line, by cycling through
 lines.
Application.Alert('Content of line ' + nextLineSel + ': ' +
ObjTest.Value());
//Insert new line at that selection
ObjTest.InsertString(nextLineSel, 'This was inserted at line ' +
(nextLineSel + 1));
```

SetEndPoints

Redraws a line on the form. A straight line is drawn from the point described by the parameters longX1 and longY1 to the point described by the parameters longX2 and longY2. Arrowhead properties are not affected by this method; however, they can be set by modifying the properties Arrowhead and ArrowheadHeight. You can also modify the line type using the PenStyle property.

Applies to

- Line

Returns

No return value

Parameters

longX1 - The X coordinate in pixels of the starting point of the line

longX2 - The Y coordinate in pixels of the starting point of the line

longY1 - The X coordinate in pixels of the ending point of the line

longY2 - The Y coordinate in pixels of the ending point of the line

Usage in JScript

```
ObjTest.SetEndPoints(long, long, long, long);
```

Usage in VBScript

```
ObjTest.SetEndPoints(long, long, long, long)
```

Example

```
// Name of the line that contains the property/method is ObjTest.
// Following code will move the line from current coordinate by the amount
 specified below.
// change these values as needed
var cx1 = 10;
var cy1 = 10;
var cx2 = 50;
var cy2 = 100;

ObjTest.SetEndPoints(ObjTest.x1 + cx1,ObjTest.y1 + cy1,ObjTest.x2 +
cx2,ObjTest.y2 + cy2);
```

SetFocus

Places the cursor into the selected control and makes the control active. This is similar to the user clicking on or tabbing to the control.

Applies to

- Button
- CheckButton
- ComboBox
- EditBox
- ListBox
- MultiEditBox
- RadioButton

Returns

No return value

Parameters

None

Usage in JScript

```
ObjTest.SetFocus();
```

Usage in VBScript

```
ObjTest.SetFocus()
```

Example

```
// Name of the Object that contains the property/method is ObjTest.
// Folowing will set the focus to the corresponding object.
// Observe the focuse after executing this code.
ObjTest.SetFocus();
```

SetPoint

Applies to

- Connector
- Freehand
- Polyline

Returns

No return value

Parameters

longIndex - The index number of the point to set the coordinates for

longX - The number of pixels from the left side of the form to set the coordinates to

longY - The number of pixels from the top of the form to set the coordinates to

Description

The three objects, Connector, Freehand, and Polyline, are all made up of several line segments. Each line segment has an X and Y coordinate denoting its starting point. The ending points of a line segment are the starting points of the next segment.

When you create an object with multiple line segments, several points are used, and each has an index number. If you want to change the X and Y coordinates of a single line segment in an object that contains several line segments, you must indicate, using the parameter longIndex, which segment you want the coordinates for. The last line segment in the object has an index number of 0. The second-to-last line segment has an index of 1, and so on.

When setting the point for the start of the line segment, use the parameters longX to indicate the number of pixels from the left side of the form and longY to indicate the number of pixels from the right side of the form.

Usage in JScript

```
ObjTest.SetPoint(long, long, long);
```

Usage in VBScript

```
ObjTest.SetPoint(long, long, long)
```

Example

```
// Name of the Object that contains the property/method is ObjTest.

// This code moves the object ObjTest right and down by a specified amount.
var Xdistance = 10;
var Ydistance = 5;

for (i = 0; i < ObjTest.NumberOfPoints(); i++){
Application.Alert('Old point X:' + ObjTest.GetPointX(i) + ', Y:' +
ObjTest.GetPointY(i));
ObjTest.SetPoint(i, ObjTest.GetPointX(i) + Xdistance, ObjTest.GetPointY(i)
 + Ydistance);
Application.Alert('New point X:' + ObjTest.GetPointX(i) + ', Y:' +
ObjTest.GetPointY(i));
}
```

SetWindowText

Sets the displayed text in an edit box or multi-edit box.

Applies to

- EditBox
- MultiEditBox

Returns

No return value

Parameters

strText - Text to set the Window Text to become

Description

If you are setting the Text property for a MultiEditBox, and want to add several lines of text, separate each line with a `Chr(10)` character, for example:

```
myMultiEditBox.SetWindowText "First line" + CHR(10) + "Second line" + CHR(10) + "Third
line"
```

Usage in JScript

```
ObjTest.SetWindowText(string);
```

Usage in VBScript

```
ObjTest.SetWindowText(string)
```

Example

```
// Name of the Object that contains the property/method is ObjTest.

// This code retrieves the text and set the modified version of it
```



```
var originalText = ObjTest.GetWindowText();
ObjTest.SetWindowText('{Modified: ' + originalText + ' }');
```

TextValue

Retrieves the text string displayed in a combo box control. If the text field of the combo box is empty, a blank string is returned.

Applies to

- ComboBox

Returns

String

Parameters

None

Usage in JScript

```
var displayedText = ObjTest.TextValue();
```

Usage in VBScript

```
displayedText = ObjTest.TextValue()
```

Example

```
// Name of the comboBox that contains the property/method is ObjTest.
// This code will retrieve the text selected from the comboBox.
Application.Alert('You have selected: ' + ObjTest.TextValue());
```

Value

Retrieves the text string of the item highlighted in a list box control. If no item is highlighted, or there is no text in the highlighted line, a blank string is returned.

Applies to

- ListBox

Returns

String

Parameters

None

Usage in JScript

```
var strVal = ObjTest.Value();
```

Usage in VBScript

```
strVal = ObjTest.Value()
```

Example

```
// Name of the ListBox that contains the property/method is ObjTest.

// Make sure you have at least one item in the ListBox to execute the
following code
// This code cycles through the lines in ListBox and inserts new line at
 that line
Application.Alert('There are ' + ObjTest.getCount() + ' items in the
ListBox.');

Application.Alert('Currently item ' + (ObjTest.getCurSel() + 1) + ' is
selected in the ListBox.');

var nextLineSel = (ObjTest.getCurSel() + 1) % ObjTest.getCount();
ObjTest.SetCurSel(nextLineSel); //Select the next line, by cycling through
 lines.
Application.Alert('Content of line ' + nextLineSel + ': ' +
ObjTest.Value());
//Insert new line at that selection
ObjTest.InsertString(nextLineSel, 'This was inserted at line ' +
(nextLineSel + 1));
```

XFT form objects

The following is a list of XFT form objects.

ActiveX

ActiveX windows control. Allows any well behaved ActiveX control to be inserted at run time, for its properties to be set up, and for event handlers to be connected.

Description

APIs for embedded ActiveX controls are provided from the third (created) tab of the property sheet pane. Click the new tab to view the properties and methods available. For information on how to use the ActiveX control's API, you must consult that control's documentation.

For an example of an embedded control in an XFT form, consult the sample file `...\Author\Forms\PubDate.xft`, which embeds the MSCAL.Calendar ActiveX control.

Properties

- Bottom
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Group
- Layer
- Left
- Right
- Tabstop

- Tag
- ToolTipText
- Top
- Visible
- XmlNodeRules
- XPath

Arc

Static elliptical arc object.Angle1, Angle2 and IsWedge properties can be edited from the property sheet only. The arc is measured in degrees going counter-clockwise.

Properties

- Angle1
- Angle2
- BackColor
- Bottom
- Code
- CursorPointer
- Enable
- EndAngle
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- GroupID
- HatchStyle
- IsWedge
- Layer
- LayerID
- Left
- PenStyle
- PenWidth
- Right
- rotateangle
- ShadowStyle
- Tag
- ToolTipText
- Top
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- LayerName
- Move
- RunClick
- RunInitialize

Events

- OnUpdate
- OnUpdateResize

Bitmap

Static bitmap object. When configuring the Bitmap object, the Bitmap property has a special interface.

Properties

- AnchorSnaps
- BackColor
- Bitmap
- Bottom
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- GroupID
- HatchStyle
- Layer
- LayerID
- Layout
- Left
- PenStyle
- PenWidth
- Right
- RotateAngle
- ShadowStyle
- Tag
- ToolTipText
- Top
- TransparentColor
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- LayerName
- Move
- RunClick
- RunInitialize

Events

- OnUpdate
- OnUpdateResize

Border

Static line object, with four lines total, making up a rectangle.

Properties

- AnchorSnaps
- BackColor
- BorderStyle
- Bottom
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- GroupID
- HatchStyle
- HiliteColor
- Layer
- LayerID
- Left
- PenStyle
- PenWidth
- Right
- RotateAngle
- ShadowStyle
- Tag
- ToolTipText
- Top
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- LayerName
- Move
- RunClick
- RunInitialize

Events

- OnUpdate
- OnUpdateResize

Button

Standard push button Windows control.

Properties

- Alignment
- BackColor
- Bitmap
- BorderWidth
- Bottom
- ButtonShape
- ButtonType
- Code
- CursorPointer
- Default
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- Group
- GroupID
- Layer
- LayerID
- Layout
- Left
- Right
- RotateAngle
- SpacerWidth
- Tabstop
- Tag
- Text
- ToolTipText
- Top
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- GetDlgCtrlID
- LayerName
- Move
- RunClick
- RunInitialize
- SetFocus

Events

- OnUpdate
- OnUpdateResize
- OnUpdateStyle

CheckButton

Standard check box button Windows control.

Properties

- AlignTextLeft
- BackColor
- Bitmap
- BitmapOffState
- Bottom
- Check
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- Group
- GroupID
- Layer
- LayerID
- Left
- OwnerDrawn
- Right
- RotateAngle
- Tabstop
- Tag
- Text
- ToolTipText
- Top
- TriState
- Value
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- GetDlgCtrlID
- IncrValue
- LayerName
- Move
- RunChange
- RunInitialize
- SetFocus

Events

- OnUpdate

- OnUpdateResize
- OnUpdateStyle

ComboBox

Standard combo box Windows control.

Properties

- Accelerator
- BackColor
- Bottom
- Code
- ComboType
- CursorPointer
- DataField
- DataSource
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- Group
- GroupID
- Layer
- LayerID
- Left
- ListItems
- NumDropped
- Right
- RotateAngle
- Selection
- Sort
- Tabstop
- Tag
- ToolTipText
- Top
- UseColors
- Value
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- AddItem
- AddString
- DeleteItem
- DeleteString
- EditItem
- GetDlgCtrlID

- GetNumItems
- GetText
- LayerName
- Move
- ResetContent
- RunChange
- RunInitialize
- SetFocus
- TextValue

Events

- OnUpdate
- OnUpdateResize
- OnUpdateStyle

Connector

Can be used only to make a connection between two anchor snaps (snap points). It is not possible to use a connector unless both ends are anchored to a snap point (the connector will auto-delete itself if both ends are not snapped). In order for the connector to be active, you must view the snap points on your form: click **View** -> **Snap points**.

Properties

- Alignment
- Arrowhead
- ArrowheadHeight
- BackColor
- Bottom
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- GroupID
- HatchStyle
- Layer
- LayerID
- Left
- PenStyle
- PenWidth
- Right
- RotateAngle
- ShadowStyle
- Tag
- Text
- ToolTipText
- Top
- Visible

- XmlNodeRules
- XmlValue
- XPath

Methods

- GetPointX
- GetPointY
- LayerName
- Move
- NumberOfPoints
- RunClick
- RunInitialize
- SetPoint

Events

- OnUpdate
- OnUpdateResize

DataSource

The data source object must be added to any form that uses external data sources. When dragged onto the form, it is visible during design time, but invisible at run time. In order to have a way to move to from row to row in the data source, you must add at least one drop down or list box to the form.

Properties

- BackColor
- Bottom
- Code
- CursorPointer
- Enable
- Font
- ForeColor
- GroupID
- LayerID
- Left
- Right
- RotateAngle
- Source
- Tag
- ToolTipText
- Top
- XmlNodeRules
- XmlValue
- XPath

Events

- OnUpdate
- OnUpdateResize

EditBox

Edit Windows control.Supports all types of hyper-variables.

Properties

- Accelerator
- Alignment
- BackColor
- BorderDrawn
- Bottom
- CaseOrPassword
- Code
- CursorPointer
- DataField
- DataSource
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- Group
- GroupID
- Layer
- LayerID
- Left
- ReadOnly
- Right
- RotateAngle
- Tabstop
- Tag
- Text
- ToolTipText
- Top
- UseColors
- ValueID
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- GetDlgCtrlID
- GetWindowText
- LayerName
- Move
- RunChange
- RunInitialize
- SetFocus
- SetWindowText

Events

- OnUpdate
- OnUpdateResize
- OnUpdateStyle

Ellipse

Static ellipse object.

Properties

- AnchorSnaps
- BackColor
- Bottom
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- GroupID
- HatchStyle
- Layer
- LayerID
- Left
- PenStyle
- PenWidth
- Right
- RotateAngle
- ShadowStyle
- Tag
- ToolTipText
- Top
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- LayerName
- Move
- RunClick
- RunInitialize

Events

- OnUpdate
- OnUpdateResize

Frame

2D/3D static frame object.

Properties

- Alignment
- BackColor
- BorderColor
- BorderStyle
- BorderWidth
- Bottom
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- GroupID
- HiliteColor
- Layer
- LayerID
- Left
- Right
- RotateAngle
- Tag
- Text
- ToolTipText
- Top
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- LayerName
- Move
- RunInitialize

Events

- OnUpdate
- OnUpdateResize

Freehand

The freehand object can be used to create curved lines or freehand polyline shapes.

Properties

- AnchorSnaps

- Arrowhead
- ArrowheadHeight
- BackColor
- Bottom
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- GroupID
- HatchStyle
- Layer
- LayerID
- Left
- PenStyle
- PenWidth
- Right
- RotateAngle
- ShadowStyle
- Tag
- ToolTipText
- Top
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- GetPointX
- GetPointY
- LayerName
- Move
- NumberOfPoints
- RunClick
- RunInitialize
- SetPoint

Events

- OnUpdate
- OnUpdateResize

Hilite

Draws a semi-transparent yellow, magenta, or cyan rectangle to work as a highlighter when placed on top of other objects. For more colors, group two highlight objects of different colors (for example, to create a green highlight object, group yellow and cyan together.)

Properties

- BackColor
- Bottom
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- GroupID
- HiliteColor
- Layer
- LayerID
- Left
- Right
- RotateAngle
- Tag
- ToolTipText
- Top
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- LayerName
- Move
- RunInitialize

Events

- OnUpdate
- OnUpdateResize

Line

Static line object.

Properties

- Arrowhead
- ArrowheadHeight
- BackColor
- Bottom
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font

- ForeColor
- GroupID
- HatchStyle
- Layer
- LayerID
- Left
- PenStyle
- PenWidth
- Right
- RotateAngle
- ShadowStyle
- Tag
- ToolTipText
- Top
- Visible
- X1
- X2
- XmlNodeRules
- XmlValue
- XPath
- Y1
- Y2

Methods

- LayerName
- Move
- RunClick
- RunInitialize
- SetEndPoints

Events

- OnUpdate
- OnUpdateResize

ListBox

List box Windows control.

Description

The list box hyper-control currently supports only a single selection. It operates in two modes:

1. Copy selection from ‘List ID’ into ‘Value ID’
2. Move the selection

When adding text at design time, you must enter a special editing mode by clicking on the control, then clicking on the text area a second time. To add another line, create a carriage-return/line-feed by pressing Ctrl+Enter.

Properties

- Accelerator

- BackColor
- BorderDrawn
- Bottom
- Code
- CursorPointer
- DataField
- DataSource
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- Group
- GroupID
- HorizontalScroll
- Layer
- LayerID
- Left
- Right
- RotateAngle
- Selection
- Sort
- Tabstop
- Tag
- Text
- ToolTipText
- Top
- UseColors
- UseTabStops
- VerticalScroll
- Visible
- WantKeyInput
- XmlNodeRules
- XmlValue
- XPath

Methods

- AddString
- DeleteString
- GetCount
- GetCurSel
- GetDlgCtrlID
- GetNumItems
- GetText
- InsertString
- LayerName
- Move
- ResetContent
- RunChange

- RunInitialize
- SetCurSel
- SetFocus
- Value

Events

- OnUpdate
- OnUpdateResize
- OnUpdateStyle

MultiEditBox

Similar to the Edit Windows control, but also supports multi-line editing and additional properties.

Properties

- Accelerator
- Alignment
- BackColor
- BorderDrawn
- Bottom
- Case
- Code
- CurrCol
- CurrLine
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- Group
- GroupID
- HorizontalScroll
- Layer
- LayerID
- Left
- ReadOnly
- Right
- RotateAngle
- Tabstop
- Tag
- Text
- ToolTipText
- Top
- UseColors
- ValueID
- VerticalScroll
- Visible
- XmlNodeRules
- XmlValue

- XPath

Methods

- GetDlgCtrlID
- GetLineCount
- GetLineOfText
- GetSelectedText
- GetWindowText
- LayerName
- Move
- RunChange
- RunInitialize
- SetFocus
- SetWindowText

Events

- OnUpdate
- OnUpdateResize
- OnUpdateStyle

Polyline

Static polyline object.

Properties

- AnchorSnaps
- Arrowhead
- ArrowheadHeight
- BackColor
- Bottom
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- GroupID
- HatchStyle
- Layer
- LayerID
- Left
- PenStyle
- PenWidth
- Right
- RotateAngle
- ShadowStyle
- Tag
- ToolTipText

- Top
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- GetPointX
- GetPointY
- LayerName
- Move
- NumberOfPoints
- RunClick
- RunInitialize
- SetPoint

Events

- OnUpdate
- OnUpdateResize

RadioButton

Specifies a single or a set of radio buttons.

Description

The ‘Choices List’ property has a special interface. This list of choices dialog box maintains choices and their associated values. The hyper-variable specified by ‘Value ID’ stores the associated value.

This control works differently than in Visual Basic or other graphical languages. You must assign the values at the same time as you create the OptionButton. You can use the default, without adding any additional buttons (also different from Visual Basic or other graphical languages). To add buttons to an OptionButton group, use the ListItems property.

Properties

- AlignTextLeft
- BackColor
- Bitmap
- BitmapOffState
- Bottom
- Code
- CursorPointer
- Enable
- FlexHorizontal
- FlexVertical
- Font
- ForeColor
- Group
- GroupID
- Index

- Layer
- LayerID
- Left
- ListItems
- Right
- RotateAngle
- Tabstop
- Tag
- ToolTipText
- Top
- Value
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- GetDlgCtrlID
- LayerName
- Move
- RunChange
- RunInitialize
- SetFocus

Events

- OnUpdate
- OnUpdateResize
- OnUpdateStyle

Text

Static text object. It is derived from the border object, so it is has all the properties the border object has, plus some additional properties.

Properties

- AlignHorizontal
- AlignVertical
- Alignment
- AnchorSnaps
- BackColor
- BorderStyle
- Bottom
- Code
- CursorPointer
- DataField
- DataSource
- Enable
- FlexHorizontal
- FlexVertical

- Font
- ForeColor
- GroupID
- HatchStyle
- HiliteColor
- Layer
- LayerID
- Left
- Orientation
- PenStyle
- PenWidth
- Right
- RotateAngle
- ShadowStyle
- Tag
- Text
- ToolTipText
- Top
- Visible
- XmlNodeRules
- XmlValue
- XPath

Methods

- LayerName
- Move
- RunClick
- RunInitialize

Events

- OnUpdate
- OnUpdateResize

TheFrame

The title/caption bar of a modal or modeless form. Although it can be embedded in XMetaL Author, it has no functionality in that platform.In terms of the Windows API, it is a frame-window. This object is responsible for the appearance of the form. Its methods and properties are related to the style of the border and how the form appears at initialization.

Properties

- BorderStyle
- Code
- Color
- CursorPointer
- Font
- Height
- MaximizeBox
- MinimizeBox

- PrintScale
- ScrollBars
- ScrollHeight
- ScrollWidth
- Tag
- Title
- ToolTipText
- ViewLayers
- WhatsThisHelp
- Width

Methods

- ClearDataAll
- GUIEventAll

Events

- OnUpdate
- OnUpdateResize

TheView

The parent or container of all the controls on the form. It s a child window of TheFrame. This object is responsible for the functionality of a form as an entire view object. Its methods, events, and properties are related to the interchange of data from the document and controls within form, how data appears in the document and related events (OnUpdate, OnResize, etc.), and controlling the entire form.

Description

This object is

Properties

- AutoRecord
- BorderStyle
- Code
- Color
- CursorPointer
- Font
- Height
- Layer
- MappingEnabled
- MaximizeBox
- MinimizeBox
- PrintScale
- ReportLock
- ScrollBars
- ScrollHeight
- ScrollWidth
- Tag
- Title
- ToolTipText

- ViewLayers
- WhatsThisHelp
- Width

Methods

- Cancel
- FormFind
- GuiEvent
- IsFormOpen
- OK
- RunClick
- RunInitialize
- RunTerminate

Events

- OnUpdate
- OnUpdateResize
- OnXFTPutDocToForm
- OnXFTPutFormToDoc
- OpenDoc
- Record

UserForm

An interface to the XMetaL-specific helper functions to perform XML data mapping. This object unites the functionality of the TheFrame/TheView objects, plus it adds extra functionality for the connection a form with XMetaL. This allows forcing XMetaL to update/save data to and from a form, and gathering data from the related XML document.

Methods

- CheckAllNodesExist
- GetNodeByXPath
- ReadFromXMLDoc
- SetAutoMap

XMetaL XMAX interfaces

Properties

This sections lists all properties in the XMetaL XMAX interfaces.

AlwaysUndoClearAfterSave

Clears the undo stack whenever `Document.Save` is called. Initially, this value is set to True.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
XMAX.AlwaysUndoClearAfterSave = false;
```

Usage in VBScript

```
XMAX.AlwaysUndoClearAfterSave = false
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
 web page.

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500"></OBJECT>
<SCRIPT>
   function allowUndoAfterSave() {
   XMAX.AlwaysUndoClearAfterSave = false;
   }
</SCRIPT>
<FORM><INPUT TYPE="FILE" NAME="FILENAME" SIZE="50">
   <INPUT TYPE="BUTTON" VALUE="Allow undo after save"
ONCLICK="allowUndoAfterSave()">
</FORM>
```

AutoLayoutForCALSTable

Toggles AutoLayoutForCALSTable display on/off. When set to true, the min/max values are automatically calculated and updated. Can be toggled back to false so that the min/max values are not updated. The default setting is ‘false’.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolAutoLayoutForCALSTable = myXMControl.AutoLayoutForCALSTable;

myXMControl.AutoLayoutForCALSTable = false;
```

Usage in VBScript

```
dim boolAutoLayoutForCALSTable = myXMControl.AutoLayoutForCALSTable

myXMControl.AutoLayoutForCALSTable = false
```

Example

```
<!-- The following Object tag sets AutoLayoutForCALSTable view to false
for the next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="AutoLayoutForCALSTable" Value="false">
</OBJECT>
```

ChangeColorByAuthor

Determines whether the revision marks change color by the author of the revisions. If the returned value is true, the number of colors of revision marks matches the number of authors of revisions. If false, InsertionColor specifies the insertion mark color and DeletionColor specifies the deletion mark color. This property is normally set at design time.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var myBoolean = myXMControl.ChangeColorByAuthor;myXMControl.ChangeColorByAuthor
= myBoolean;
```


Usage in VBScript

```
dim myBoolean = myXMControl.ChangeColorByAuthormyXMControl.ChangeColorByAuthor
= myBoolean
```

Example

```
// JSCRIPT example:
if (myXMControl.ChangeColorByAuthor) {
myXMControl.Document.Host.Alert("Revision marks change color by author.");
} else {
myXMControl.Document.Host.Alert("The color of insertion marks is specified
 by InsertionColor.");
}
```

DefaultWTDictionaryId

Specifies the default language to use for spell checking the XML document. More information is available on our forum *here*.

Applies to

XMetaL XMAX

Access

Read/write

Returns

String

Usage in JScript

```
XMAX.DefaultWTDictionaryId = "EN-US";
```

Usage in VBScript

```
XMAX.DefaultWTDictionaryId = "EN-US"
```

Example

```
<!-- The following Object tag sets DefaultWTDictionaryId view to false
for the next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="DefaultWTDictionaryId" Value="EN-US">
</OBJECT>
```

DeletionColor

If ChangeColorByAuthor is set to false, this property determines the color of a user's deletion marks while revision tracking is turned on. DeletionColor is usually a design-time property. You can supply a 32-bit integer value to `DeletionColor`. The default color is red. Applies only when revision marking is turned on.

Applies to

XMetaL XMAX

Access

Read/write

Returns

OLE_COLOR

Usage in JScript

```
var myOLEColor = myXMControl.DeletionColor;

myXMControl.DeletionColor = myOLEColor;
```

Usage in VBScript

```
dim myOLEColor = myXMControl.DeletionColor

myXMControl.DeletionColor = myOLEColor
```

Example

```
// JSCRIPT example:
myXMControl.Document.Host.Alert("The deletion color of revision marks is
 set to " + myXMControl.DeletionColor);
```

DeletionStyle

Specifies the display style for a deletion. Applies only when revision marking is turned on. This property is normally set at design time.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Integer

The possible return values are:

- delStrikethrough (0): Display deleted text with a strikethrough effect.
- delHidden (1): Hide deleted text.

The default is delStrikethrough (0).

Usage in JScript

```
var myInteger = myXMControl.DeletionStyle;

myXMControl.DeletionStyle = myInteger;
```

Usage in VBScript

```
dim myInteger = myXMControl.DeletionStyle
```



```
myXMControl.DeletionStyle = myInteger
```

Example

```
// JSCRIPT example:
if (myXMControl.DeletionStyle == 0) {
myXMControl.Document.Host.Alert("When revision marking is on, strikethrough
 style applies to deleted text.");
} else {
myXMControl.Document.Host.Alert("When revision marking is on, deleted text
 is hidden.");
}
```

Document

Provides access to XMetaL's document object. It is the primary document interface, and the entry point into the DOM interfaces.

Applies to

XMetaL XMAX

Access

Read only

Returns

Document object

Usage in JScript

```
var document = myXMControl.Document;
```

Usage in VBScript

```
dim document = myXMControl.Document
```

Example

```
// a JSCRIPT example:
// Displays the source text of the document

// pre: myXMControl has previously been defined as an XMetaL XMAX object

myXMControl.LoadFromFile(myFile);
if (myXMControl.Document != null) {
ActiveDocument.Host.Alert(myXMControl.Document.xml);
}
```

DrawGridOnBorderlessTable

Toggles DrawGridOnBorderlessTable display on/off. When set to true, a border is drawn around tables. Can be toggled back to false so table borders are not drawn. The default setting is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolDrawGridOnBorderlessTable = myXMControl.DrawGridOnBorderlessTable;

myXMControl.DrawGridOnBorderlessTable = false;
```

Usage in VBScript

```
dim boolDrawGridOnBorderlessTable = myXMControl.DrawGridOnBorderlessTable

myXMControl.DrawGridOnBorderlessTable = false
```

Example

```
<!-- The following Object tag sets DrawGridOnBorderlessTable view to false
 for the next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="DrawGridOnBorderlessTable" Value="false">
</OBJECT>
```

EmptyElementImagePath

Specifies if the GIF to display in Tags On and Normal view mode that denotes an empty element. This property is normally set at design time. The default value is 'empty.gif' in the XMAX installation folder.

Applies to

XMetaL XMAX

Access

Read/write

Returns

String

Usage in JScript

```
myXMControl.EmptyElementImagePath = "empty.gif";

var emptyGIFpath = myXMControl.EmptyElementImagePath;
```

Usage in VBScript

```
myXMControl.EmptyElementImagePath = 'empty.gif'

dim emptyGIFpath = myXMControl.EmptyElementImagePath
```


Example

```
<!-- The following Object tag sets EmptyElementImagePath for the next
document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="EmptyElementImagePath" Value="empty.gif">
</OBJECT>
```

EnableBidiAuthoring

Enable right-to-left authoring and the bi-directional formatter. This value is set to False by default.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
XMAX.EnableBidiAuthoring = true;
```

Usage in VBScript

```
XMAX.EnableBidiAuthoring = True
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
 web page.

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500"></OBJECT>
<SCRIPT>
   function enableBidiAuthoring() {
      XMAX.EnableBidiAuthoring = true;
   }
</SCRIPT>
<FORM><INPUT TYPE="FILE" NAME="FILENAME" SIZE="50">
   <INPUT TYPE="BUTTON" VALUE="Enable Bi-directional Authoring"
ONCLICK="enableBidiAuthoring()">
</FORM>
```

EnablePlainTextView

Toggles Plain Text view on/off. When set to true, makes Plain Text view visible on the next LoadFromFile or similar API call. Can be toggled back to false to make PlainText view invisible. Must not be called while XMAX has a document loaded. The default setting is ‘false’.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
boolEnablePlainTextView = myXMControl.EnablePlainTextView;

myXMControl.EnablePlainTextView = true;
```

Usage in VBScript

```
boolEnablePlainTextView = myXMControl.EnablePlainTextView

myXMControl.EnablePlainTextView = true
```

Example

```
<!-- The following Object tag sets EnablePlainText view to true for the
next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="EnablePlainTextView" Value="true">
</OBJECT>
```

EnableStructureView

This property toggles Structure View splitter control visibility on/off. When set to 'true', Structure View splitter control is visible on the next LoadFromFile or similar API call. it can be toggled back to 'false' to make Structure View invisible. it must not be called while XMAX has a document loaded. The default setting is 'false'.

Applies To

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
boolEnableStructureView = myXMControl.EnableStructureView;
myXMControl.EnableStructureView = true;
```


Usage in VBScript

```
boolEnableStructureView = myXMControl.EnableStructureView
myXMControl.EnableStructureView = True
```

Example

```
<!-- The following Object tag sets EnableStructureView view to true for the next document
 load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX" HEIGHT="500"
WIDTH="500">
   <PARAM Name="EnableStructureView" Value="true">
</OBJECT>
```

Handle

Returns the window handle of XMetaL XMAX. You can use this property to work with XMetaL XMAX at a Win32 API level.

Applies to

XMetaL XMAX

Access

Read only

Returns

Long

Usage in JScript

```
var myLongInt = myXMControl.Handle;
```

Usage in VBScript

```
dim myLongInt = myXMControl.Handle
```

Example

```
' A Visual Basic example (not VBScript)
' Sets the screen focus to XMetaL XMAX

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA"
(ByVal hwnd As IntPtr, ByVal wMsg As Int32, ByVal wParam As Int32, ByVal
 lParam As IntPtr) As Int32
Const WM_SETFOCUS = 7

Private Sub SetFocusToXMControl()
SendMessage(myXMControl.Handle, WM_SETFOCUS, Me.Handle.ToInt32,
IntPtr.Zero)
End Sub
```

InsertionColor

If ChangeColorByAuthor is set to false, this property determines the color of a user's insertion marks while revision tracking is turned on. DeletionColor is usually a design-time property. You can supply a 32-bit integer value to `DeletionColor`. The default color is blue. Applies only when revision marking is turned on.

Applies to

XMetaL XMAX

Access

Read/write

Returns

OLE_COLOR

Usage in JScript

```
var myOLEColor = myXMControl.InsertionColor;
myXMControl.InsertionColor = myOLEColor;
```

Usage in VBScript

```
dim myOLEColor = myXMControl.InsertionColor
myXMControl.InsertionColor = myOLEColor
```

Example

```
// JSCRIPT example:
myXMControl.Document.Host.Alert("The insertion color of revision marks is
 set to " + myXMControl.InsertionColor);
```

InsertionStyle

Specifies the display style for an insertion. Applies only when revision marking is turned on. This property is normally set at design time.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Integer

The possible return values are:

- insNormal (0): No special style applies to inserted text
- insUnderline (1): Underline inserted text
- insBold (2): Bold inserted text
- insItalic (3): Italicize inserted text

The default value for DeletionStyle is insUnderline (1).

Usage in JScript

```
var myInteger = myXMControl.InsertionStyle;

myXMControl.InsertionStyle = myInteger;
```

Usage in VBScript

```
dim myInteger = myXMControl.InsertionStyle

myXMControl.InsertionStyle = myInteger
```

Example

```
// JSCRIPT example:
if (myXMControl.InsertionStyle == 0) {
myXMControl.Document.Host.Alert("When revision marking is on, no special
 style applies to inserted text.");
} else if (myXMControl.InsertionStyle == 1) {
myXMControl.Document.Host.Alert("When revision marking is on, inserted
text is underlined.");
} else if (myXMControl.InsertionStyle == 2) {
myXMControl.Document.Host.Alert("When revision marking is on, inserted
text is bolded.");
} else {
myXMControl.Document.Host.Alert("When revision marking is on, inserted
text is italicized.");
}
```

IsSpellCheckerInstalled

If the spell checker (Writing Tools) is installed on the system and available to XMetaL XMAX, this property returns true; otherwise it returns false. This is especially useful to Web page designers who need to know whether or not to install the writing tools.

Applies to

XMetaL XMAX

Access

Read-only

Returns

Boolean

Installing the writing tools

An installer, writingtools10.exe, is provided.

The standalone WritingTools installer for XMetaL XMAX has several command-line options:

Run silently:

```
writingtools10.exe /s /v/qn
```

Install to a specific path:

```
writingtools10.exe /v"INSTALLDIR=<pathname>"
```

Where <pathname> is the path to the destination folder, e.g., `C:\WritingToolsInstall`.

Run silently and install to a specific path:

```
writingtools10.exe /s /v"/qn INSTALLDIR=<pathname>"
```

Where `<pathname>` is the path to the destination folder, e.g., `C:\WritingToolsInstall`.

Usage in JScript

```
var myBool = myXMControl.IsSpellCheckerInstalled;
```

Usage in VBScript

```
dim myBool = myXMControl.IsSpellCheckerInstalled
```

Example

```
//Jscript example
function isSpellCheckerInstalled()
{
if (myXMControl.IsSpellCheckerInstalled)
alert("Spell Checker Installed!");
else
alert("Spell Checker not available");
}
```

LicenseID

The host name or IP address of the XMetaL XMAX License Server. This property must be set at design time whenever the XMetaL XMAX License Server is used to provide licenses for XMetaL XMAX. Do not set the LicenseID property if XMetaL XMAX License Server is not used.

Applies to

XMetaL XMAX

Access

Read/write

Returns

String

Example

```
<!-- use this code in a web page intended for use in Internet Explorer -
->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myXMControl" HEIGHT="500" WIDTH="500">
<PARAM NAME="LicenseID" VALUE="myLicenseServerName">
</OBJECT>
```

MiniContextVisible

Sets or returns the visibility of the mini-context bar (box below the editor window displaying the hierarchy for the current element). This property must be set before loading a document into XMetaL XMAX. By default, this property is set to true. Two separate running instances of XMetaL XMAX can have different values. This property can be set at design time, but it can also be set programmatically.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
boolMiniContextVisible = myXMControl.MiniContextVisible;

myXMControl.MiniContextVisible = false;
```

Usage in VBScript

```
boolMiniContextVisible = myXMControl.MiniContextVisible

myXMControl.MiniContextVisible = false
```

Example

```
// JSCRIPT example:
// Hide the mini-context bar when a sample document is loaded
// Assumes myXMControl is a properly initialized XMetaL XMAX object

myXMControl.MiniContextVisible = false;
var doc = myXMControl.LoadFromFile("C:\\sample.xml");
// At this point, the document view will not display the mini-context bar
```

MissingImageImagePath

Specifies if the GIF to display in Tags On and Normal view mode that denotes a broken image element file reference. This property is normally set at design time. The default value is "noimg.gif" in the XMAX installation folder.

Applies to

XMetaL XMAX

Access

Read/write

Returns

String

Usage in JScript

```
myXMControl.MissingImageImagePath = "noimg.gif";

var noimgPath = myXMControl.MissingImageImagePath;
```

Usage in VBScript

```
myXMControl.MissingImageImagePath = 'noimg.gif'

dim noimgPath = myXMControl.MissingImageImagePath
```

Example

```
<!-- The following Object tag sets MissingImageImagePath for the next
document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="MissingImageImagePath" Value="noimg.gif">
</OBJECT>
```

NormalizeImagesURLsOnSave

Specifies if document's image file references should be normalized when the document is saved. This property is normally set at design time. The default value is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.NormalizeImagesURLsOnSave = false;

var saveDecl = myXMControl.NormalizeImagesURLsOnSave;
```

Usage in VBScript

```
myXMControl.NormalizeImagesURLsOnSave = False

dim saveDecl = myXMControl.NormalizeImagesURLsOnSave
```

Example

```
<!-- The following Object tag sets NormalizeImagesURLsOnSave for the next
 document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="NormalizeImagesURLsOnSave" Value="false">
</OBJECT>
```

NormalizeXMLBaseURLsOnSave

Specifies if document's xml:base attribute value should be normalized when the document is saved. This property is normally set at design time. The default value is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.NormalizeXMLBaseURLsOnSave = false;
```

```
var saveDecl = myXMControl.NormalizeXMLBaseURLsOnSave;
```

Usage in VBScript

```
myXMControl.NormalizeXMLBaseURLsOnSave = False
```

```
dim saveDecl = myXMControl.NormalizeXMLBaseURLsOnSave
```

Example

```
<!-- The following Object tag sets NormalizeXMLBaseURLsOnSave for the next
 document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="NormalizeXMLBaseURLsOnSave" Value="false">
</OBJECT>
```

NormalizeXIncludeURLsOnSave

Specifies if document's xinclude href attribute values should be normalized when the document is saved. This property is normally set at design time. The default value is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.NormalizeXIncludeURLsOnSave = false;
```

```
var saveDecl = myXMControl.NormalizeXIncludeURLsOnSave;
```

Usage in VBScript

```
myXMControl.NormalizeXIncludeURLsOnSave = False
```

```
dim saveDecl = myXMControl.NormalizeXIncludeURLsOnSave
```

Example

```
<!-- The following Object tag sets NormalizeXIncludeURLsOnSave for the
next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="NormalizeXIncludeURLsOnSave" Value="false">
</OBJECT>
```

NormalizeSystemIdURLsOnSave

Specifies if doctype declaration's system identifier value should be normalized when the document is saved. This property is normally set at design time. The default value is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.NormalizeSystemIdURLsOnSave = false;
```

```
var saveDecl = myXMControl.NormalizeSystemIdURLsOnSave;
```

Usage in VBScript

```
myXMControl.NormalizeSystemIdURLsOnSave = False
```

```
dim saveDecl = myXMControl.NormalizeSystemIdURLsOnSave
```

Example

```
<!-- The following Object tag sets NormalizeSystemIdURLsOnSave for the
next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="NormalizeSystemIdURLsOnSave" Value="false">
</OBJECT>
```

ParserLaxEmptyContent

Boolean method that controls parsing and validation for empty content.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Description

When the property is set to true, the validator allows comments in an element which has the empty content model.

Usage in JScript

```
XMControl.ParserLaxEmptyContent = true;
```

Usage in VBScript

```
XMControl.ParserLaxEmptyContent = True
```

PTVAutoIndent

Specifies if plain-text view's auto indent mode is enabled. Applies only to plain-text view mode editing. This property is normally set at design time. The default value is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.PTVAutoIndent = false;
```

```
var autoIndent = myXMControl.PTVAutoIndent;
```

Usage in VBScript

```
myXMControl.PTVAutoIndent = False
```

```
dim autoIndent = myXMControl.PTVAutoIndent
```

Example

```
<!-- The following Object tag sets PTVAutoIndent for the next document
load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="PTVAutoIndent" Value="false">
</OBJECT>
```

PTVExpandTabsOnSave

Specifies if plain-text view's expand-tabs-on-save mode is enabled. Applies only to plain-text view mode editing. This property is normally set at design time. The default value is 'false'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.PTVExpandTabsOnSave = true;
```

```
var expandTabsOnSave = myXMControl.PTVExpandTabsOnSave;
```

Usage in VBScript

```
myXMControl.PTVExpandTabsOnSave = True
```

```
dim expandTabsOnSave = myXMControl.PTVExpandTabsOnSave
```

Example

```
<!-- The following Object tag sets PTVExpandTabsOnSave for the next
document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="PTVExpandTabsOnSave" Value="true">
</OBJECT>
```

PTVFontName

Specifies plain-text view's font family to use for displaying XML text. Applies only to plain-text view mode editing. This property is normally set at design time. The default value is 'Courier New'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

String

Usage in JScript

```
myXMControl.PTVFontName = "Courier";
```



```
var fontName = myXMControl.PTVFontName;
```

Usage in VBScript

```
myXMControl.PTVFontName = "Courier"

dim fontName = myXMControl.PTVFontName
```

Example

```
<!-- The following Object tag sets PTVFontName for the next document load
 -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="PTVFontName" Value="Courier">
</OBJECT>
```

PTVFontSize

Specifies plain-text view's font size to use for displaying XML text. Applies only to plain-text view mode editing. This property is normally set at design time. The default value is '12'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Long

Usage in JScript

```
myXMControl.PTVFontSize = "Courier";

var fontSize = myXMControl.PTVFontSize;
```

Usage in VBScript

```
myXMControl.PTVFontSize = "Courier"

dim fontSize = myXMControl.PTVFontSize
```

Example

```
<!-- The following Object tag sets PTVFontSize for the next document load
 -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="PTVFontSize" Value="16">
</OBJECT>
```

PTVNoWrapInTags

Specifies plain-text view's line wrapping inside tags mode. Applies only to plain-text view mode editing. This property is normally set at design time. The default value is 'false'. A value of 'true' only has effect when PTVWrapLines is set to '2'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.PTVNoWrapInTags = true;
```

```
var noWrapInTags = myXMControl.PTVNoWrapInTags;
```

Usage in VBScript

```
myXMControl.PTVNoWrapInTags = True
```

```
dim noWrapInTags = myXMControl.PTVNoWrapInTags
```

Example

```
<!-- The following Object tag sets PTVNoWrapInTags for the next document
 load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="PTVWrapLines"    Value="2">
   <PARAM Name="PTVNoWrapInTags" Value="true"></OBJECT>
```

PTVShowLineNumbering

Specifies if plain-text view's line numbering mode is enabled. Applies only to plain-text view mode editing. This property is normally set at design time. The default value is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.PTVShowLineNumbering = false;

var showLineNumbering = myXMControl.PTVShowLineNumbering;
```

Usage in VBScript

```
myXMControl.PTVShowLineNumbering = False

dim showLineNumbering = myXMControl.PTVShowLineNumbering
```

Example

```
<!-- The following Object tag sets PTVShowLineNumbering for the next
document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="PTVShowLineNumbering" Value="false">
</OBJECT>
```

PTVShowTabs

Specifies if plain-text view's show tabs mode is enabled, otherwise a single space is shown for each tab character. Applies only to plain-text view mode editing. This property is normally set at design time. The default value is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.PTVShowTabs = false;

var showTabs = myXMControl.PTVShowTabs;
```

Usage in VBScript

```
myXMControl.PTVShowTabs = False

dim showTabs = myXMControl.PTVShowTabs
```

Example

```
<!-- The following Object tag sets PTVShowTabs for the next document load
 -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="PTVShowTabs" Value="false">
</OBJECT>
```

PTVTabSize

Specifies plain-text view's tab size which can be from 1 to 16. Applies only to plain-text view mode editing. This property is normally set at design time. The default value is '3'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Long

Usage in JScript

```
myXMControl.PTVTabSize = 4;
```

```
var tabSize = myXMControl.PTVTabSize;
```

Usage in VBScript

```
myXMControl.PTVTabSize = 4
```

```
dim tabSize = myXMControl.PTVTabSize
```

Example

```
<!-- The following Object tag sets PTVTabSize for the next document load
 -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="PTVTabSize" Value="4">
</OBJECT>
```

PTVWrapLines

Specifies plain-text view's wrap lines mode. Applies only to plain-text view mode editing. This property is normally set at design time. The value can be 0 (Off), 1 (Break within words), or 2 (Break between words). The default value is '0'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Long

Usage in JScript

```
myXMControl.PTVWrapLines = 2;

var wrapLines = myXMControl.PTVWrapLines;
```

Usage in VBScript

```
myXMControl.PTVWrapLines = 2

dim wrapLines = myXMControl.PTVWrapLines
```

Example

```
<!-- The following Object tag sets PTVWrapLines for the next document load
 -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="PTVWrapLines" Value="2">
</OBJECT>
```

ReducedTagsMode

Toggles ReducedTagsMode display on/off. When set to true, makes Tags On view icons display without any text inside the tag icon. Can be toggled back to false to make Tags On view icons display text inside each icon again. The default setting is ‘false’.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolReducedTagsMode = myXMControl.ReducedTagsMode;

myXMControl.ReducedTagsMode = true;
```

Usage in VBScript

```
dim boolReducedTagsMode = myXMControl.ReducedTagsMode

myXMControl.ReducedTagsMode = true
```

Example

```
<!-- The following Object tag sets ReducedTagsMode view to true for the
next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="ReducedTagsMode" Value="true">
</OBJECT>
```

SaveWithDoctypeDecl

Specifies if document's doctype declaration is saved. This property is normally set at design time. The default value is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.SaveWithDoctypeDecl = false;

var saveDecl = myXMControl.SaveWithDoctypeDecl;
```

Usage in VBScript

```
myXMControl.SaveWithDoctypeDecl = False

dim saveDecl = myXMControl.SaveWithDoctypeDecl
```

Example

```
<!-- The following Object tag sets SaveWithDoctypeDecl for the next
document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="SaveWithDoctypeDecl" Value="false">
</OBJECT>
```

SaveWithSGMLDecl

Specifies if document's SGML declaration is saved. This property is normally set at design time. The default value is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.SaveWithSGMLDecl = false;
```



```
var saveDecl = myXMControl.SaveWithSGMLDecl;
```

Usage in VBScript

```
myXMControl.SaveWithSGMLDecl = False

dim saveDecl = myXMControl.SaveWithSGMLDecl
```

Example

```
<!-- The following Object tag sets SaveWithSGMLDecl for the next document
 load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="SaveWithSGMLDecl" Value="false">
</OBJECT>
```

Selection

The current selection in the XMetaL XMAX document. Identical to the Selection global variable.

Applies to

XMetaL XMAX

Access

Read only

Returns

Selection object

Usage in JScript

```
var selection = myXMControl.Selection;
```

Usage in VBScript

```
dim selection = myXMControl.Selection
```

Example

```
// JSCRIPT example
// Displays selected text

// pre: myXMControl has previously been defined as an XMetaL XMAX object

if (myXMControl.Selection != null){
myXMControl.Document.Host.Alert(myXMControl.Selection.Text);
}
```

ShowComments

Toggles ShowComments display on/off. When set to false, XML comments are not displayed in Tags On or Normal view. Can be toggled back to true to make XML comments reappear. The default setting is ‘true’.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolShowComments = myXMControl.ShowComments;
```

```
myXMControl.ShowComments = false;
```

Usage in VBScript

```
dim boolShowComments = myXMControl.ShowComments
```

```
myXMControl.ShowComments = false
```

Example

```
<!-- The following Object tag sets ShowComments view to false for the next
 document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="ShowComments" Value="false">
</OBJECT>
```

ShowInlineImages

Toggles ShowInlineImages display on/off. When set to false, inline images are not displayed and that permits editing of alt-text or similar child markup normally hidden by the image. Can be toggled back to true to make inline images reappear. The default setting is ‘true’.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolShowInlineImages = myXMControl.ShowInlineImages;
```

```
myXMControl.ShowInlineImages = false;
```


Usage in VBScript

```
dim boolShowInlineImages = myXMControl.ShowInlineImages

myXMControl.ShowInlineImages = false
```

Example

```
<!-- The following Object tag sets ShowInlineImages view to false for the
 next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="ShowInlineImages" Value="false">
</OBJECT>
```

ShowTagTips

Toggles ShowTagTips display on/off. When set to false, tag icon tooltips are not displayed in Tags On view. Can be toggled back to true to make tag icon tooltips reappear. The default setting is ‘true’.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolShowTagTips = myXMControl.ShowTagTips;

myXMControl.ShowTagTips = false;
```

Usage in VBScript

```
dim boolShowTagTips = myXMControl.ShowTagTips

myXMControl.ShowTagTips = false
```

Example

```
<!-- The following Object tag sets ShowTagTips view to false for the next
 document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="ShowTagTips" Value="false">
</OBJECT>
```

SVWidth

This property sets the initial width in pixels of the Structure View panel. The setting is only in effect the first time the panel is made visible via the _Document.StructureViewVisible API. The default width is 220 pixels.

Applies To

XMetaL XMAX

Access

Read/write

Returns

Long

Usage in JScript

```
var intSVWidth = myXMControl.SVWidth;
myXMControl.SVWidth = intSVWidth;
```

Usage in VBScript

```
intSVWidth = myXMControl.SVWidth
myXMControl.SVWidth = intSVWidth
```

Example

```
<!-- The following Object tag sets EnableStructureView view to true for the next document
 load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX" HEIGHT="500"
WIDTH="500">
   <PARAM Name="SVWidth" Value="100">
</OBJECT>
```

TagFontName

The TagFontName property is used to specify the font family that XMAX uses for the tag icon text. The default tag icon font name is "Arial". Applies only when in Tags On view mode.

Applies to

XMetaL XMAX

Access

Read/write

Returns

String

Usage in JScript

```
var myTagFontName = myXMControl.TagFontName;

myXMControl.TagFontName = myTagFontName;
```

Usage in VBScript

```
dim myTagFontName = myXMControl.TagFontName

myXMControl.TagFontName = myTagFontName
```


Example

```
// JSCRIPT example:
myXMControl.Document.Host.Alert("The tag font name is set to " +
myXMControl.TagFontName);
```

TagFontSize

The TagFontSize property is used to specify the font size that XMAX uses for tag icon text. The default tag icon font size is 8. Applies only when in Tags On view mode.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Long

Usage in JScript

```
var myTagFontSize = myXMControl.TagFontSize;

myXMControl.TagFontSize = myTagFontSize + 2;
```

Usage in VBScript

```
dim myTagFontSize = myXMControl.TagFontSize

myXMControl.TagFontSize = myTagFontSize + 2
```

Example

```
// JSCRIPT example:
myXMControl.Document.Host.Alert("The tag font size is set to " +
myXMControl.TagFontSize);
```

TagColor

The TagColor property is used to specify the color that XMAX uses for tag icons. The default tag icon color is 000000. Applies only when in Tags On view mode.

Applies to

XMetaL XMAX

Access

Read/write

Returns

OLE_COLOR

Usage in JScript

```
var myTagColor = myXMControl.TagColor;

myXMControl.TagColor = myTagColor;
```

Usage in VBScript

```
dim myTagColor = myXMControl.TagColor

myXMControl.TagColor = myTagColor
```

Example

```
// JSCRIPT example:
myXMControl.Document.Host.Alert("The tag icon color is set to " +
myXMControl.TagColor);
```

TagBkgdColor

The TagBkgdColor property is used to specify the background color that XMAX uses for tag icons. The default background tag icon color is CDCDCD. Applies only when in Tags On view mode.

Applies to

XMetaL XMAX

Access

Read/write

Returns

OLE_COLOR

Usage in JScript

```
var myTagBkgdColor = myXMControl.TagBkgdColor;

myXMControl.TagBkgdColor = myTagBkgdColor;
```

Usage in VBScript

```
dim myTagBkgdColor = myXMControl.TagBkgdColor

myXMControl.TagBkgdColor = myTagBkgdColor
```

Example

```
// JSCRIPT example:
myXMControl.Document.Host.Alert("The tag icon background color is set to
 " + myXMControl.TagBkgdColor);
```

TagsOnViewGraphicalTables

Defines the initial value for Document.TagsOnGraphicalTables property upon loading a XML document. When set to false, raw table markup is displayed in Tags On view for all tables upon initially opening a XML document.

Can be toggled back to true to make tables reappear as formatted rows and columns upon opening the next XML document. The default setting is ‘true’.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolTagsOnGraphicalTables = myXMControl.TagsOnViewGraphicalTables;

myXMControl.TagsOnViewGraphicalTables = false;
```

Usage in VBScript

```
dim boolTagsOnGraphicalTables = myXMControl.TagsOnViewGraphicalTables

myXMControl.TagsOnViewGraphicalTables = false
```

Example

```
<!-- The following Object tag sets TagsOnViewGraphicalTables view to false
 for the next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="TagsOnViewGraphicalTables" Value="false">
</OBJECT>
```

TagTipsWithFixedAttrs

Toggles TagTipsWithFixedAttrs display on/off. When set to false, fixed attribute values are not displayed in the tag icon tooltip text. Can be toggled back to true to make fixed attribute values also appear in the tag icon tooltip text. The default setting is ‘true’.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
var boolTagTipsWithFixedAttrs = myXMControl.TagTipsWithFixedAttrs;

myXMControl.TagTipsWithFixedAttrs = false;
```

Usage in VBScript

```
dim boolTagTipsWithFixedAttrs = myXMControl.TagTipsWithFixedAttrs

myXMControl.TagTipsWithFixedAttrs = false
```

Example

```
<!-- The following Object tag sets TagTipsWithFixedAttrs view to false
for the next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="TagTipsWithFixedAttrs" Value="false">
</OBJECT>
```

UserName

Specifies the user name value to assign to tracked changes. Applies only when revision marking is turned on. This property is normally set at design time. The default value is the current Windows user login name.

Applies to

XMetaL XMAX

Access

Read/write

Returns

String

Usage in JScript

```
myXMControl.UserName = "Bob Smith";

var userName = myXMControl.UserName;
```

Usage in VBScript

```
myXMControl.UserName = "Bob Smith"

dim userName = myXMControl.UserName
```

Example

```
<!-- The following Object tag sets UserName for the next document load -
->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="UserName" Value="Bob Smith">
</OBJECT>
```

ValidateBeforeExport

Enable or disable XML validation during document save or export operations. This value is set to True by default.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
XMAX.ValidateBeforeExport = false;
```

Usage in VBScript

```
XMAX.ValidateBeforeExport = False
```

Example

```
<!-- The following Object tag sets ValidateBeforeExport view to false for
 the next document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="ValidateBeforeExport" Value="false">
</OBJECT>
```

Xac

An absolute file path to the XAC corresponding to the current instance of XMetaL XMAX. This property is intended for use when initializing XMetaL XMAX in a browser with the Xml property.

Applies to

XMetaL XMAX

Access

Read/write

Returns

String

Description

The file path can be specified in the following ways:

- `C:\my.xac`
- `\\my_network_computer\my.xac`
- `http://www.mydomainname.com/my.xac`
- `https://www.mydomainname.com/my.xac`

Note: Loading a XAC using HTTPS works only if the file is on the same HTTPS server.

Usage in JScript

```
myXMControl.Xac = "C:\\my.xac";

var myXac = myXMControl.Xac;
```

Usage in VBScript

```
myXMControl.Xac = "C:\my.xac"

dim myXac = myXMControl.Xac
```

Example

```
// JSCRIPT example
// Create a new XML document, and display the XML source

// pre: myXMControl has previously been defined as an XMetaL XMAX object

myXMControl.Xac = "C:\\journalist.xac";
var xmlDec = '<?xml version="1.0"?>';
var contents1 = "<Article><Title>New Document</Title>";
var contents2 = "<Para>Contents</Para></Article>";
var docstr = xmlDec + docType + contents1 + contents2;

myXMControl.Xml = docstr;
myXMControl.Document.Host.Alert(myXMControl.Xml);
```

Xml

Provides access to the entire document as a string. You can use this property to initialize XMetaL XMAX with a string before a Document object is constructed. You must always use an absolute path in the document type declaration when specifying the DTD or schema.

Applies to

XMetaL XMAX

Access

Read/write

Returns

String

Usage in JScript

```
myXMControl.Xml = "<someXMLdata />";

var myString = myXMControl.Xml;
```

Usage in VBScript

```
myXMControl.Xml = "<someXMLdata />"

dim myString = myXMControl.Xml
```


Example

```
// JSCRIPT example
// Create a new XML document, and display the XML source

// pre: myXMControl has previously been defined as an XMetaL XMAX object

var xmlDec = '<?xml version="1.0"?>';
var docType = '<!DOCTYPE Article SYSTEM "C:\\Program Files\\XMetaL
5.0\\Author\\Rules\\journalist.dtd">';
var contents1 = "<Article><Title>New Document</Title>";
var contents2 = "<Para>Contents</Para></Article>";
var docstr = xmlDec + docType + contents1 + contents2;

myXMControl.Xml = docstr;
myXMControl.Document.Host.Alert(myXMControl.Xml);
```

XmlOpenAsWellFormed

Specifies if document is allowed to be opened as well-formed if there are parsing errors. Otherwise, the document will not be opened. This property is normally set at design time. The default value is 'true'.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
myXMControl.XmlOpenAsWellFormed = false;

var saveDecl = myXMControl.XmlOpenAsWellFormed;
```

Usage in VBScript

```
myXMControl.XmlOpenAsWellFormed = False

dim saveDecl = myXMControl.XmlOpenAsWellFormed
```

Example

```
<!-- The following Object tag sets XmlOpenAsWellFormed for the next
document load -->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500">
   <PARAM Name="XmlOpenAsWellFormed" Value="false">
</OBJECT>
```

xmlWithCT

Recovers XML source with change tracking processing instructions preserved.

Applies to

XMetaL Author

Access

Read only

Returns

String value corresponding to the entire document with change tracking processing instructions preserved

Usage in JScript

```
var myString = myXMControl.XmlWithCT;
```

Usage in VBScript

```
dim myString = myXMControl.XmlWithCT
```

Example

```
// a JSCRIPT example:
// retrieve the entire document and all changes
// made with Change Tracking

// pre: myXMControl has previously been defined as an XMetaL XMAX object

var doc = myXMControl.xmlWithCT;
ActiveDocument.Host.Alert(doc);
```

Methods

This sections lists all methods in the XMetaL XMAX interfaces.

LoadFromFile

Opens the specified file.

Applies to

XMetaL XMAX

Returns

Document object

Usage in JScript

```
var myDocument = myXMControl.LoadFromFile(strFileName, [strXacURI],
[intViewType]);
```

`strFileName` must be a string containing the absolute path. Some examples of a valid file path are `C:\myxmldoc.xml`, `\\my_network_computer\myxmldoc.xml`, and `http://mydomain.tld/myxmldoc.xml` (WebDAV only).

strXacURI is an optional string that specifies the absolute file path to a XAC. The file path can be specified in the following ways:

- C:\my.xac
- \\my_network_computer\my.xac
- http://www.mydomainname.com/my.xac (Includes WebDAV)
- https://www.mydomainname.com/my.xac (Includes WebDAV)

Note: Loading a XAC using HTTPS works only if the file is on the same HTTPS server.

If strXacURI is specified, and strFileName contains a document type declaration with a system identifier, the system identifier must match the rules file contained in the XAC. If the system identifier specifies a DTD or schema, the XAC at location strXacURI must contain a DTD or schema of the same name (.dtd/.xsd or .rlx/.rld). You must use an absolute path in the document type declaration to locate the DTD or schema.

If strXacURI is not specified, you can still load a document customization in other ways.

If you specify intViewType, the document opens in the corresponding view. By default, the document opens in Normal view. When specifying the optional integer intViewType, use these values:

- -1 (Unspecified)
- 0 (Normal view)
- 1 (Tags On view)

You can also use the SQDocViewType enumerated constant to specify the view type.

Note: strFileName must refer to a well-formed XML file for LoadFromFile to return a Document object. Otherwise, LoadFromFile will return null. To avoid runtime errors in your program, check that LoadFromFile returned an object before trying to use the return value.

Usage in VBScript

```
dim myDocument = myXMControl.LoadFromFile(strFileName, [strXacURI], [intViewType])
```

Example

```
// JSCRIPT example
// Displays the source text of the document

// pre: myXMControl has previously been defined as an XMetaL XMAX object
// and myFile has previously been defined as a file name

myXMControl.LoadFromFile(myFile);
if (myXMControl.Document != null) {
myXMControl.Document.Host.Alert(myXMControl.Document.Xml);
}
```

LoadFromString

Opens a new file as XML.

Applies to

XMetaL XMAX

Returns

Document object

Opening as partial documents

If you set the optional boolean value `boolWFEdit` to true, XMetaL XMAX loads the document as a well-formed document (a partial document). If a DTD or schema cannot be found, a message is displayed and XMetaL continues loading the document as well-formed.

Usage in JScript

```
var myDoc = myXMControl.LoadFromString(strString, [strXacURI], [intViewType]);
```

`strString` contains the contents of the XML file.

`strXacURI` is an optional string that specifies the absolute file path to a XAC. The file path can be specified in any of the following ways:

- `C:\my.xac`
- `\\my_network_computer\my.xac`
- `http://www.mydomainname.com/my.xac` (Includes WebDAV)
- `https://www.mydomainname.com/my.xac`(Includes WebDAV)

Note: Loading a XAC using HTTPS works only if the file is on the same HTTPS server.

If `strXacURI` is specified, and `strFileName` contains a document type declaration with a system identifier, the system identifier must match the rules file contained in the XAC. If the system identifier specifies a DTD or schema, the XAC at location `strXacURI` must contain a DTD or schema of the same name (`.dtd/.xsd` or `.rlx/.rld`). You must use an absolute path in the document type declaration to locate the DTD or schema.

If `strXacURI` is not specified, you can still load a document customization in other ways.

If you specify `intViewType`, the document opens in the corresponding view. By default, the document opens in Normal view. When specifying the optional integer `intViewType`, use these values:

- -1 (Unspecified)
- 0 (Normal view)
- 1 (Tags On view)

You can also use the SQDocViewType enumerated constant to specify the view type.

Note: `strFileName` must refer to a well-formed XML file for `LoadFromFile` to return a Document object. Otherwise, `LoadFromFile` will return null. To avoid runtime errors in your program, check that `LoadFromFile` returned an object before trying to use the return value.

Usage in VBScript

```
dim myDoc = myXMControl.LoadFromString(strString, [strXacURI], [intViewType])
```


Example

```
// sample JSCRIPT code intended for Internet Explorer webpages
// surround this function with <SCRIPT> </SCRIPT>
// you can call this function from a button on a form

function testLoadFromString(){
var xmlDec = '<?xml version="1.0"?>';
var docType = '<!DOCTYPE Article SYSTEM "C:\\Program Files\\XMetaL
5.0\\Author\\Rules\\journalist.dtd">';
var contents1 = "<Article><Title>New Document</Title>";
var contents2 = "<Para>Contents</Para></Article>";
var docstr = xmlDec + docType + contents1 + contents2;

myXMControl.LoadFromString(docstr);
}
```

LoadFromStringAsSGML

Opens a new file as SGML.

Applies to

XMetaL XMAX

Returns

Document object

Usage in JScript

`var myDoc = myXMControl.LoadFromStringAsSGML(strString, [strXacURI], [intViewType]);`

`strString` contains the contents of the SGML file. It must contain valid SGML for `LoadFromStringAsSGML` to return a Document object. Otherwise, `LoadFromStringAsSGML` will return null. To avoid runtime errors in your program, check that `LoadFromStringAsSGML` returned an object before trying to use the return value.

`strXacURI` is an optional string that specifies the absolute file path to a XAC. The file path can be specified in any of the following ways:

- `C:\my.xac`
- `\\my_network_computer\my.xac`
- `http://www.mydomainname.com/my.xac` (Includes WebDAV)
- `https://www.mydomainname.com/my.xac`(Includes WebDAV)

Note: Loading a XAC using HTTPS works only if the file is on the same HTTPS server.

If `strXacURI` is specified, and `strFileName` contains a document type declaration with a system identifier, the system identifier must match the rules file contained in the XAC. If the system identifier specifies a DTD, the XAC at location `strXacURI` must contain a DTD of the same name (`.dtd` or `.rls`). You must use an absolute path in the document type declaration to locate the DTD or schema.

If `strXacURI` is not specified, you can still load a document customization in other ways.

If you specify `intViewType`, the document opens in the corresponding view. By default, the document opens in Normal view. When specifying the optional integer `intViewType`, use these values:

- -1 (Unspecified)
- 0 (Normal view)
- 1 (Tags On view)

You can also use the SQDocViewType enumerated constant to specify the view type.

Usage in VBScript

```
dim myDoc = myXMControl.LoadFromStringAsSGML(strString, [strXacURI],
[intViewType])
```

Example

```
// sample JSCRIPT code intended for Internet Explorer webpages
// you can call this function from a button on a form

<!-- Somewhere on the web page -->
<!-- assume there is a Journalist XAC at http://www/journalist.xac-->
<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="EE"
HEIGHT="500" WIDTH="500">
</OBJECT>

<SCRIPT>
function loadFromStringAsSGML()
{
var docType = '<!DOCTYPE Article SYSTEM "journalist.dtd">';
var contents1 = "<Article><Title>New Document</Title>";
var docstr = docType + contents1;
EE.LoadFromStringAsSGML(docstr, "http://www/journalist.xac");
}
</SCRIPT>

<FORM><INPUT TYPE="FILE" NAME="FILENAME" SIZE="50">
<INPUT TYPE="BUTTON" VALUE="Load From String as SGML"
ONCLICK="loadFromStringAsSGML()">
</FORM>
```

MapXmlLangToWTDictionaryId

Maps and xml:lang attribute value to a specific WritingTools dictionary. More information is on our forum *here*.

Applies to

XMetaL XMAX

Access

Read/write

Returns

Boolean

Usage in JScript

```
XMAX.MapXmlLangToWTDictionaryId("en-us", "US");
```


Usage in VBScript

```
XMAX.MapXmlLangToWTDictionaryId "en-us", "US"
```

Example

```
// XMetaL Script Language JSCRIPT:
// This code is intended for use with XMetaL XMAX in an Internet Explorer
 web page.

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46" ID="XMAX"
HEIGHT="500" WIDTH="500"></OBJECT>
<SCRIPT>
   function useCanadianEnglish() {
     XMAX.MapXmlLangToWTDictionaryId("en-us", "CE");
   }
</SCRIPT>
<FORM>
   <INPUT TYPE="BUTTON" VALUE="Use Canadian English"
ONCLICK="useCanadianEnglish()">
</FORM>
```

ShowAbout

Displays the About box.

Applies to

XMetaL XMAX

Returns

No return value

Usage in JScript

```
myXMControl.ShowAbout();
```

Usage in VBScript

```
myXMControl.ShowAbout
```

Example

```
// JSCRIPT example:
// Show the about box
// Assumes myXMControl is a properly initialized XMetaL XMAX object

myXMControl.ShowAbout();
```

ShowSpellChecker

Shows the spell checker dialog and starts the spell-checking process. Before showing the spell checker, it is a good idea to use the IsSpellCheckerInstalled property to determine whether or not the spell checker is installed (and whether to install it).

Applies to

XMetaL XMAX

Returns

No return value

Description

The spell checking process has the following limitations:

- There can be only one spell checker modeless dialog at any time, and it is always attached to the frame window that the spell checker dialog is first displayed from.

  For example, if the container that displays the spell checker is an instance of Internet Explorer, and new Internet Explorer windows are shown via **File** > **New Window** (that is, all of the Internet Explorer windows are running in the same process), only one of those Internet Explorer windows is able to show the spell checker dialog. However, if new Internet Explorer windows are launched via the system Start menu or via the Quick Launch Bar, then those Internet Explorer windows are able to show separate spell checker dialogs because they work in different processes.
- If a user pauses the spell checking process without activating an XMetaLControl main window (for example, by clicking on other places in the Internet Explorer window), the spell checker dialog may not switch to Resume mode. If the selection is changed by a script and then the **Replace** button on the spell checker is clicked, the spell checker attempts to replace the current selection, causing a system crash if the current selection can not be replaced.

Usage in JScript

```
myXMControl.ShowSpellChecker();
```

Usage in VBScript

```
myXMControl.ShowSpellChecker()
```

Example

```
//Jscript example
function showSpellChecker()
{
myXMControl.ShowSpellChecker();
}
```

Events

This sections lists all events in the XMetaL XMAX interfaces.

OnCheckAttributeValue

Fires whenever XMetaL XMAX encounters an attribute value during validation. Fires before the On_Check_Attribute_Value event macro.

Applies to

XMetaL XMAX

Usage in JScript

```
function myXMControl::OnCheckAttributeValue(checkData) {  //handle event  }
```

The `checkData` parameter is a readable and writable CheckData object, and it contains information related to the attributes and elements being validated.

Usage in VBScript

```
Sub myXMControl_OnCheckAttributeValue(checkData) 'handle event  End Sub
```

Example

```
// JSCRIPT example
// validates the URL of an InlineGraphic or Graphic

// pre: myXMControl has previously been defined as an XMetaL XMAX object

function myXMControl::OnCheckAttributeValue(checkData) {
var elem = checkData.Element;
var name = elem.tagName;
var attrName = checkData.AttributeName;

if((name == "InlineGraphic" || name == "Graphic") && attrName == "FileRef")
 {
var value = new String(checkData.Value);
var msg = null;

if(str.indexOf(" ")>=0) {
msg = "URL cannot have whitespace:"
} else if (str.indexOf("\\")>= 0) {
msg = "URL cannot have backslash:"
}

if(msg) {
msg += checkData.AttributeName;
msg += "='";
msg += value;
msg += "'";
checkData.ValidationMsg = msg;
}
}
}
```

OnCheckElementSimpleContent

Fires whenever XMetaL XMAX encounters an element with simple content (an element that allows only PCDATA). Fires before the On_Check_Element_SimpleContent event macro.

Applies to

XMetaL XMAX

Usage in JScript

```
function myXMControl::OnCheckElementSimpleContent(checkData) { //handle event }
```

The `checkData` parameter is a readable and writable CheckData object, and it contains information related to the attributes and elements being validated.

Usage in VBScript

```
Sub myXMControl_OnCheckElementSimpleContent(checkData) 'handle event End Sub
```

Example

```
// JSCRIPT example
// validates an email address (which contains only PCDATA)

// pre: myXMControl has previously been defined as an XMetaL XMAX object

function myXMControl::OnCheckElementSimpleContent(checkData) {
var elem = checkData.Element;
var name = elem.tagName;

// Check value..... if(name == "Email") {
var value = checkData.Value;

if (value != null) {
var re = new RegExp("^[^@ ]+@[^@ ]+$");

if (re.exec(value) == null) {
var msg = "Invalid email address value:"
msg += "'";
msg += value;
msg += "'";
checkData.ValidationMsg = msg;
}
}
}
}
```

OnClick

Fires after a mouse button click.

Applies to

XMetaL XMAX

Usage in JScript

```
function myXMControl::OnClick(longMouseButton, intNumberOfClicks, longX, longY)
```

Usage in VBScript

```
Sub myXMControl_OnClick(longMouseButton, intNumberOfClicks, longX, longY)
```

The `longMouseButton` parameter is a read-only long integer that represents the mouse button clicked. This parameter returns one of the following values:

- 1 (Left mouse button clicked)
- 2 (Right mouse button clicked)
- 3 (Middle mouse button clicked)

`intNumberOfClicks` is a read-only integer that represents the number of clicks of the mouse. Valid values are 1 and 2. The amount of time allowed to perform a double-click is set in the Mouse settings of the Windows Control Panel. If you want to perform special handling when the mouse is double-clicked, do not display any message boxes in `OnClick` when `intNumberOfClicks=1`. Displaying message boxes prevents `OnClick` from firing with `intNumberOfClicks=2`.

`longX` is a read-only long integer that represents the X-coordinate of the current cursor.

longY is a read only long integer that represents the Y-coordinate of the current cursor. The X and Y coordinates are relative to the XMetaL XMAX editor window. The top left corner of the editor window is the origin (0,0).

When the left mouse button is clicked, the On_Click event macro is fired after OnClick fires. When the left mouse button is double-clicked, OnClick fires twice. The On_Click event macro is fired after OnClick fires with a value of 1 for intNumberOfClicks, and the On_Double_Click event macro is fired after OnClick fires with a value of 2 for intNumberOfClicks. When the right mouse button is clicked, the OnContextMenu event is fired after OnClick fires.

Example

```
' VBScript example:
' Intended for use in an Internet Explorer web page
' Open a URL in a new window
Sub myXMControl_OnClick(longMouseButton, intNumberOfClicks, longX, longY)
If ((mouseButton = 1) AND (clicks = 1) AND
myXMControl.Selection.ContainerNode.hasAttribute("URL")) Then
window.open(myXMControl.Selection.ContainerAttribute("URL"))
End If
End Sub
```

OnContextMenu

Fires when the right mouse button is clicked, when Shift+F10 is pressed, or when the Application key is pressed. If the right mouse button is clicked, this event fires immediately after the OnClick event fires. Replaces the On_Context_Menu event macro.

Applies to

XMetaL XMAX

Usage in JScript

```
function myXMControl::OnContextMenu(longX, longY, boolHandled) {// handle event}
```

Usage in VBScript

```
Sub myXMControl_OnContextMenu(longX, longY, boolHandled) ' handle eventEnd Sub
```

The longX parameter is a read-only long integer that represents the X-coordinate of the current cursor. The longY parameter is a read-only long integer that represents the Y-coordinate of the current cursor.

The boolHandled parameter is a read/write BoolRefVar that indicates whether or not XMetaL XMAX should stop its default behavior. If boolHandled.value is set to true, this indicates that the OnContextMenu event was handled by the container application, and XMetaL XMAX should stop its default behavior. If boolHandled.value is set to false, XMetaL XMAX will perform its default behavior.

Example

```
' VBScript example
' Show custom context menu. Please refer to the HTML sample for
' the implementation of context (popup) menu using DHTML.

Sub myXMControl_OnContextMenu(longX, longY, boolHandled)
showContextMenu 1, X, Y
boolHandled.value = True
End Sub
```

OnDragOver

Fires when an object is dragged over XMetaL XMAX. This event fires only when you use the left mouse button to drag anobject over XMetaL XMAX.

Applies to

XMetaL XMAX

Description

The `dropDataObject` parameter is a read-only IDataObject pointer. It is identical to the Document.DropDataObject property. `dropDataObject` is a pointer to the data object that is currently being dragged.

`longKeyState` is a read-only parameter that represents the state of the keyboard and mouse modifier keys. `longKeyState` returns the sum of the keys and buttons pressed. The values are as follows:

Key or button pressed	Hexadecimal representation	Decimal representation
Ctrl	0x0008	8
Shift	0x0004	4
Alt	0x0020	32
Left mouse button	0x0001	1
Right mouse button	0x0002	2
Middle mouse button	0x0010	16

For example, if you hold down the Ctrl key while dragging an object with the left mouse button held down, `longKeyState` returns a decimal value of 9.

The parameters `longX` and `longY` represent the current coordinates of the cursor. These parameters are read-only long integers.

The `longDropEffect` parameter is a read/write IntegerRefVar that represents the suggested drop effect. The following values are valid:

Drop effect	Value	Description
None	0	Drop not allowed. **Note:** The OnDrop event is not fired if `longDropEffect` = 0.
Copy	1	Drop results in a copy. The data in the drag source remains unchanged, and the drop point contains a copy of the data in the drag source. **Note:** You must hold down the Ctrl key to perform a copy.

Drop effect	Value	Description
Move	2	Drop results in a move. The data in the drag source is moved to the drop point. The drag source is then deleted. **Note:** If you cannot delete the drag source, you will not be allowed to perform a move.
Link	4	Drop results in a link. The drag source remains unchanged, and the drop point contains a link to the data in the drag source.

The most predictable drop effect is None (0). Whenever you specify a drop effect of none, XMetaL XMAX disallows the drop of the object. Copy, Move, or Link drop effects depend on other factors, such as whether the original object can be deleted or linked, and whether the Ctrl key is held down. Copy, Move, and Link drop effects should be used sparingly.

Note: The drop effect specified in the OnDrop event takes precedence over the drop effect specified in `OnDragOver`. Keep drop effects consistent between these two events.

The `boolWasHandled` parameter is a read/write BoolRefVar. It represents whether the containing application for XMetaL XMAX handled the `OnDragOver` event.

Note: You must set `boolWasHandled.value` to true to override the default XMetaL XMAX drop effect. If`boolWasHandled.value` is false, changes made to `longDropEffect.value` are discarded.

Usage in JScript

```
function myXMControl::OnDragOver(dropDataObject, longKeyState, longX, longY,
longDropEffect, boolWasHandled) {}

//handle event
```

Usage in VBScript

```
Sub myXMControl_OnDragOver(dropDataObject, longKeyState, longX, longY,
longDropEffect, boolWasHandled) ' handle eventEnd Sub
```

Example

```
' VBScript example
' Use a third party object "DLLName.MyObject" to process
' dropDataObject and set the drop effect

Sub myXMControl_OnDragOver(dropDataObject, longKeyState, longX, longY,
longDropEffect, boolWasHandled)
Dim customObject
Set customObject = CreateObject("DLLName.MyObject")
customObject.process(dropDataObject) ' Manipulate dropDataObject

If (customObject.isMove = True)
longDropEffect.value = 2 ' Move = 2
boolWasHandled.value = True
Else If (customObject.isCopy = True)
longDropEffect.value = 1 ' Copy = 1
boolWasHandled.value = True
```

```
Else
' Default action
boolWasHandled.value = False
End If
End Sub
```

OnDrop

Fires only when you use the left mouse button to drop an object on XMetaL XMAX. The dropped object can be internal or external.

Applies to

XMetaL XMAX

Description

The `dropDataObject` parameter is a read-only IDataObject pointer. It is identical to the Document.DropDataObject property. This parameter is a pointer to the data object that is currently being dragged.

The `longKeyState` parameter is a read-only parameter that represents the state of the keyboard and mouse modifier keys. This parameter returns the sum of the keys and buttons pressed. The values are as follows:

Key or button pressed	Hexadecimal representation	Decimal representation
Ctrl	0x0008	8
Shift	0x0004	4
Alt	0x0020	32
Left mouse button	0x0001	1
Right mouse button	0x0002	2
Middle mouse button	0x0010	16

For example, if you hold down the Ctrl key while dropping an object with the left mouse button, `longKeyState` will return a decimal value of 8 (since you have to release the left mouse button to drop an object). If you hold down Ctrl and Shift while dropping an object with the left mouse button, longKeyState will return a decimal value of 12.

The parameters `longX` and `longY` are represent the current X and Y coordinates of the cursor. These parameters are read-only long integers.

The `longDropEffect` parameter is a read/write IntegerRefVar that represents the suggested drop effect. The following values are valid:

Drop effect	longDropEffect.value	Description of drop effect
None	0	Drop not allowed.

Drop effect	longDropEffect.value	Description of drop effect
Copy	1	Drop results in a copy. The data in the drag source remains unchanged, and the drop point contains a copy of the data in the drag source. **Note:** You must hold down the Ctrl key to perform a copy.
Move	2	Drop results in a move. The data in the drag source is moved to the drop point. The drag source is then deleted. **Note:** If you cannot delete the drag source, you will not be allowed to perform a move.
Link	4	Drop results in a link. The drag source remains unchanged, and the drop point contains a link to the data in the drag source.

The most predictable drop effect is None (0). Whenever you specify a drop effect of none, XMetaL XMAX disallows the drop of the object. Copy, Move, or Link drop effects depend on other factors, such as whether the original object can be deleted or linked, and whether the Ctrl key is held down. Copy, Move, and Link drop effects should be used sparingly.

Note: The drop effect specified in the `OnDrop` event takes precedence over the drop effect specified in OnDragOver. Keep drop effects consistent between these two events.

The `rangeDropPoint` parameter corresponds to the Range at which the object is dropped. It returns the same object as Document.DropPoint.

The `boolWasHandled` parameter is a read/write BoolRefVar. It represents whether or not the containing application for XMetaL XMAX handled the `OnDrop` event.

Note: You must set `boolWasHandled.value` to true to override the default XMetaL XMAX drop effect. If`boolWasHandled.value` is false, changes made to `longDropEffect.value` are discarded.

Usage in JScript

```
function myXMControl::OnDrop(dropDataObject, longKeyState, longX, longY,
longDropEffect, rangeDropPoint, boolWasHandled)
```

Usage in VBScript

```
Sub myXMControl_OnDrop(dropDataObject, longKeyState, longX, longY, longDropEffect,
rangeDropPoint, boolWasHandled)
```

Example

```
' VBScript example
' Use a third party object "DLLName.MyObject" to process
' dropDataObject and drop the data

Sub myXMControl_OnDrop(dropDataObject, longKeyState, longX, longY,
longDropEffect, rangeDropPoint, boolWasHandled)
```

```
Dim customObject
Set customObject = CreateObject("DLLName.MyObject")
customObject.process(dropDataObject)

If (customObject.success = True)
' Perform Drop Data
boolWasHandled.value = True
Else
' Display Error Message
boolWasHandled.value = False
End If
End Sub
```

OnDocumentLoadComplete

Gives you access to the document after it opens, but before it appears in a document window. Fires immediately after a document is opened or created, that is, after LoadFromFile or LoadFromString executes. Fires before the On_Document_Open_Complete event macro.

Applies to

XMetaL XMAX

Usage in JScript

```
function myXMControl::OnDocumentLoadComplete() { //handle event }
```

Usage in VBScript

```
Sub myXMControl_OnDocumentLoadComplete 'handle event End Sub
```

Example

```
' VBScript example
' Highlight the first element in the document before display

' pre: myXMControl has previously been defined as an XMetaL XMAX object

Sub myXMControl_OnDocumentLoadComplete
Dim sel
Set sel = myXMControl.Selection
sel.ContainerStyle = "color:black; background-color:yellow"
End Sub
```

OnDTDOpenComplete

Fires after the document's rules are loaded, but before the document is read in. The `domDocumentType` parameter is a readable and writable DOMDocumentType object, and it corresponds to the rules file for the document being opened. The object can be used to temporarily add new elements and attributes to the rules. This event replaces the On_DTD_Open_Complete event macro.

Applies to

XMetaL XMAX

Usage in JScript

```
function myXMControl::OnDTDOpenComplete(docType) { //handle event }
```


Usage in VBScript

```
Sub myXMControl_OnDTDOpenComplete(docType) 'handle event End Sub
```

Example

```
' VBScript example
' Add a new element "Annotation" and associated attributes to the rules
temporarily
' pre: myXMControl has previously been defined as an XMetaL XMAX object

Sub myXMControl_OnDTDOpenComplete(docType)
Dim rootElem
rootElem = docType.name

If rootElem = "Article" And Not docType.hasElementType("Annotation") Then
'add Annotation element
docType.addElement "Annotation", "Annotation", True, False
'add attributes (UserName, Time, Initials, Comment)
docType.addAttribute "UserName", "", 0, 0
docType.addAttribute "Time", "", 0, 0
docType.addAttribute "Initials", "", 0, 0
docType.addAttribute "Comment", "", 0, 0

'add Annotation element to the Article element's inclusion list
If docType.hasElementType("Annotation") And
docType.hasElementType("Article") Then
docType.addElementToInclusions "Annotation", "Article"
End If
End If
End Sub
```

OnMessage

Fired whenever XMetaL XMAX generates a message. Provides a means for containing applications to apply their own look and feel to the messages displayed to the user. If the containing application does not handle this event, XMetaL XMAX displays its default message box.

Applies to

XMetaL XMAX

intMessageBoxType values

The first group of values (0 to 5) describes the number and type of buttons displayed in the dialog box; the second group (16, 32, 48, 64) describes the icon style; the third group (0,256,512) determines which button is the default; and the fourth group (0,4096) determines the modality of the message box. When adding numbers to create a final value for the argument buttons, use only one number from each group.

Value	Constant	Description
0	vbOKOnly	Display OK button only
1	vbOKCancel	Display OK and Cancel buttons
2	vbAbortRetryIgnore	Display Abort, Retry, and Ignore buttons
3	vbYesNoCancel	Display Yes, No, and Cancel buttons
4	vbYesNo	Display Yes and No buttons
5	vbRetryCancel	Display Retry and Cancel buttons

Value	Constant	Description
16	vbCritical	Display Critical Message icon
32	vbQuestion	Display Warning Query icon
38	vbExclamation	Display Warning Message icon
64	vbInformation	Display Information Message icon
0	vbDefaultButton1	First button is default
256	vbDefaultButton2	Second button is default
512	vbDefaultButton3	Third button is default
0	vbApplicationModal	Application modal; the user must respond to the message box before continuing work in the current application.
4096	vbSystemModal	System modal; all applications are suspended until the user responds to the message box.

`intMessageBoxReturnValue` is a writable IntegerRefVar which accepts the following values:

- 0: Event not handled. XMetaL XMAX will display its default message box.
- vbOK (1): OK button was clicked. XMetaL XMAX will not display its default message box.
- vbCancel (2): Cancel button was clicked. XMetaL XMAX will not display its default message box.
- vbAbort (3): Abort button was clicked. XMetaL XMAX will not display its default message box.
- vbRetry (4): Retry button was clicked. XMetaL XMAX will not display its default message box.
- vbIgnore (5): Ignore button was clicked. XMetaL XMAX will not display its default message box.
- vbYes (6): Yes button was clicked. XMetaL XMAX will not display its default message box.
- vbNo (7): No button was clicked. XMetaL XMAX will not display its default message box.

`intMessageID` is a read-only 32-bit integer value identifying the message. Currently, this parameter always returns 0, and is reserved for future use.

`strDefaultDisplayMessage` is a read-only default Unicode string, written in English, that contains the default message string to be displayed.

Usage in JScript

```
function myXMControl::OnMessage(intMessageID, strDefaultDisplayMessage,
intMessageBoxType, intMessageBoxReturnValue)
```

Usage in VBScript

```
Sub myXMControl_OnMessage(intMessageID, strDefaultDisplayMessage,
intMessageBoxType, intMessageBoxReturnValue)
```

Example

```
' VBScript example
' Put the following code on the web page inside the BODY element

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
```



```
ID="myXMControl" HEIGHT="500" WIDTH="500">

<SCRIPT>
Sub myXMControl_OnMessage(intMessageID, strDefaultDisplayMessage,
intMessageBoxType, intMessageBoxReturnValue)
intMessageBoxReturnValue.value = MsgBox (strDefaultDisplayMessage,
intMessageBoxType)
End Sub
</SCRIPT>
```

OnStatus

Fires whenever text is supposed to show in the status bar. A standard way of hooking up status notifications with the container.

Applies to

XMetaL XMAX

Usage in JScript

```
function myXMControl::OnStatus(intMessageID, strDefaultDisplayMessage,
intDefaultStatusBarPane)
```

`intMessageID` is a read-only 32-bit integer value identifying the message. Currently, this parameter always returns 0, and is reserved for future use.

`strDefaultDisplayMessage` is a read-only default Unicode string, written in English, that contains the default message string to be displayed.

`intDefaultStatusPane` is an optional read-only integer parameter that specifies which status bar pane the message should appear in. The status bar panes are numbered in ascending order, starting from 0, from left to right.

Usage in VBScript

```
Sub myXMControl_OnStatus(intMessageID, strDefaultDisplayMessage,
intDefaultStatusBarPane) ' handle eventEnd Sub
```

Example

```
' VBScript example
' Put the following code on the web page inside the BODY element

<OBJECT CLASSID="CLSID:55EC30BE-EA43-45C4-ABD7-DDA7F6943D46"
ID="myXMControl" HEIGHT="500" WIDTH="500">

<SCRIPT>
Sub myXMControl_OnStatus(intMessageID, strDefaultDisplayMessage,
intDefaultStatusBarPane)
If (strDefaultDisplayMessage <> "") Then
window.status = strDefaultDisplayMessage
End If
End Sub
</SCRIPT>
```

OnQueryService

Fires fired whenever the ActiveDocument.Host.QueryService(BSTR) is called.

Applies to

XMetaL XMAX

Object

_IXMetaLControlEvents interface

Type

COM Eventsink Method

Parameters

Application.QueriedServiceName

Application.QueriedServiceImpl

Restrictions

N/A

Example

```
<MACRO name="On_Application_Query_Service">
var name = Application.QueriedServiceName;
if (name == "MyService") {
Application.QueriedServiceImpl = getMyService();
}
</MACRO>
```

OnResolveEntity

Fires when entity references are resolved. This event replaces the On_Application_Resolve_Entity event macro.

Applies to

XMetaL XMAX

Usage in JScript

```
function myXMControl::OnResolveEntity(resolveEntityInfo) {  //handle event  }
```

Usage in VBScript

```
Sub myXMControl_OnResolveEntity(resolveEntityInfo) 'handle event  End Sub
```

The `resolveEntityInfo` parameter is a ResolveEntityInfo object, and it contains information about the entity to be resolved. The parameter is readable and writable.

Example

```
' VBScript example:
' Condense the SystemID of an XML document to a filename
' The SystemID would then use the DTD in the same folder as the XML
document
' pre: myXMControl has previously been defined as an XMetaL XMAX object
```



```
Sub myXMControl_OnResolveEntity(resolveEntityInfo)
Dim newSystemID
newSystemID = resolveEntityInfo.SystemID
newSystemID = Right(newSystemID, Len(newSystemID) - InstrRev(newSystemID,
 "/"))
newSystemID = Right(newSystemID, Len(newSystemID) - InstrRev(newSystemID,
 "\"))
resolveEntityInfo.SystemID = newSystemID
End Sub
```

OnResolveImageURL

Fires when image URLs are resolved. This event replaces the On_Application_Resolve_Image_URL event macro.

Applies to

XMetaL XMAX

Usage in JScript

```
function myXMControl::OnResolveImageURL(strResolveURL) {  //handle event  }
```

Usage in VBScript

```
Sub myXMControl_OnResolveImageURL(strResolveURL) 'handle event  End Sub
```

The `strResolveURL` parameter is a StringRefVar that contains an entity reference. It is primarily intended to resolve a URL referencing an image that cannot be located by XMetaL XMAX. It is initialized to the value from the document that contains the entity reference. The parameter is readable and writable.

Example

```
' VBScript example:
' pre: myXMControl has previously been defined as an XMetaL XMAX object

Sub myXMControl_OnResolveImageURL(strResolveURL)
' Resolve an URL by changing it into a local path
strResolveURL.value = "C:\MyLocalPath\" & strResolveURL.value
strResolveURL.value = Replace(strResolveURL.value, "/", "\")
End Sub
```

OnResolveTranscludedLink

This event is reserved for internal use only.

OnUpdateUI

Fired when the document context changes. You may need a script to update a toolbar button or menu item's enable status so that you can enable or disable toolbar buttons and menu items that run macros.

Applies to

XMetaL XMAX

Description

This event is fired when:

- The selection moves to a different element
- A document is opened
- A command is executed
- A character is typed into an element (but not when subsequent characters are entered into the same element). This event is fired after the initial OnUpdateUI event is fired as a result of a selection being moved to a different element.

This event is not allowed to modify the document.

Usage in JScript

```
function myXMControl::OnUpdateUI()
```

Usage in VBScript

```
Sub myXMControl_OnUpdateUI
```

Example

```
// JSCRIPT example:
function myXMControl::OnUpdateUI() {
// get the updated insertion and change lists
var myInsertElementList = Selection.CanInsertList;
var myChangeElementList = Selection.CanChangeList;
// code to update the UI goes here ...
}
```

OnValidationError

Fires when the document or selection fails to validate after invoking a validation API, loading a XML document or doing a view-mode switch from plain-text view.

Applies to

XMetaL XMAX

Returns

No return value

Description

Usage in JScript

```
function myXMControl::OnValidationError(veList)
```

`veList` is a ValidationErrorList object.

Usage in VBScript

```
Sub myXMControl_OnValidationError(veList)
```

Example

```
// JScript example:
function myXMControl_OnValidationError(veList)
```



```
{
  if (veList == null || veList.Count == 0) {
    alert("The document has no validation errors.");
  } else {
    alert("The document has " + veList.Count + " validation error(s).");
  }
}
```

OnViewChange

Fires when a user switches between Normal and Tags On views. The On_View_Change event macro is triggered after this event is fired.

Applies to

XMetaL XMAX

Returns

No return value

Description

Usage in JScript

`function myXMControl::OnViewChange(intPreviousViewType, intCurrentViewType)`

`intPreviousViewType` is the integer corresponding to the previous view type. Its value is identical to the value of Document.PreviousViewType.

`intCurrentViewType` is the integer corresponding to the current view type. Its value is identical to the value of Document.ViewType.

The following values are valid for both `intPreviousViewType` and `intCurrentViewType`:

- -1 (Unspecified)
- 0 (Normal view)
- 1 (Tags On view)

Usage in VBScript

`Sub myXMControl_OnViewChange(intPreviousViewType, intCurrentViewType)`

Example

```
' VBScript example:
Sub myXMControl_OnViewChange(intPreviousViewType, intCurrentViewType)
If (intPreviousViewType <> 1) Then
MsgBox "Previous view was not Tags On view.", vbInformation, "OnViewChange"
End If
End Sub
```

InPlaceControl interface

The InPlaceControl interface enables XMetaL macros to communicate with in-place OLE controls.

The `Application.ActiveInPlaceControl` property returns an InPlaceControl object whose properties enable you to manipulate the control and obtain information about its location in the document.

Properties

Control

Returns the IDispatch interface object for the instance of the control that is assigned to this element. This object gives the XMetaL macro access to all of the control's properties and methods. This property is not available in the OnShouldCreate macro.

Applies To

XMetaL Author and XMetaL XMAX

Access

Read only

Returns

IDispatch interface object

Usage in JScript

```
InPlaceControl_object.Control;
```

Usage in VBScript

```
InPlaceControl_object.Control
```

Example

```
<MACRO name="CGM_OnInitialize" lang="JScript">
// XMetaL Script Language JSCRIPT:
// Control is the IsoView CGM viewer in this example
var aipc = Application.ActiveInPlaceControl;
var isoview = aipc.Control;
// Filename is a control-specific property!
isoview.Filename = "c:\\Images\\default.cgm";
</MACRO>
```

Document

Returns the Document object for the document that hosts the control.

Applies to

XMetaL Author and XMetaL XMAX

Access

Read only

Returns

Document object

Usage in JScript

```
InPlaceControl_object.Document;
```

Usage in VBScript

```
InPlaceControl_object.Document
```

Example

```
<MACRO name="CGM_OnSynchronize" lang="JScript">
<[CDATA[
var aipc = Application.ActiveInPlaceControl;
var domnode = aipc.Node
// ...code deleted....
var attrnode = domnode.attributes.getNamedItem("Height");
// If the height is different, then update it...
if (attrnode != null && attrnode.value != aipc.Height) {
var rg = aipc.Document.Range;
rg.ContainerAttribute("Depth") = aipc.Height;
}
// ...code deleted... ]]
<MACRO>
```

Height

Gets or sets the height of the control, in pixels. This property is not valid inside the OnShouldCreate and OnSynchronize macros. It should be set only in the OnInitialize and OnFocus macros.

Applies To

XMetaL Author and XMetaL XMAX

Access

Read/write

Returns

Integer

Usage in JScript

```
var ctrlY=InPlaceControl_object.Height;
InPlaceControl_object.Height = numPixels;
```

Usage in VBScript

```
dim ctrlY=InPlaceControl_object.Height
InPlaceControl_object.Height = numPixels
```

Example

```
<MACRO name="CGM_OnFocus" lang="JScript">
<[CDATA[
var aipc = Application.ActiveInPlaceControl;
var domnode = aipc.Node;
// ...code deleted....
// If the user moved into the control
if (aipc.UserMovedIntoControl) {
var attrnode = domnode.attributes.getNamedItem("Height");
// If the height in the control is different from
// the height in the document then update the control
if (attrnode != null && attrnode.value != aipc.Height) {
var rg = attrnode.ownerDocument.Range;
aipc.Height = rg.ContainerAttribute("Depth");
}
}
// ...code deleted... ]]
<MACRO>
```

NextEventParam

This property is valid only during event-callback macros. Event-callbacks come from a control and have their own unique parameters. The event-callback script can access each of these parameters via successive calls to this property. The first call to this property returns the first argument for the callback; the second call returns the next argument and so on until the null value is returned. Retrieving this property again after receiving the null value repeats the process and returns the first parameter again and so on.

Applies to

XMetaL Author and XMetaL XMAX

Access

Read only

Returns

Variant

Usage in JScript

```
OleInPlaceGlue_object.NextEventParam;
```

Usage in VBScript

```
OleInPlaceGlue_object.NextEventParam
```

node

Returns the DOMNode that the in-place control is associated with. It is valid in the OnShouldCreate, OnInitialize, OnSynchronize, OnFocus, and event-callback macros.

Applies To

XMetaL Author and XMetaL XMAX

Access

Read only

Returns

DOMNode object

Usage in JScript

```
InPlaceControl_object.node;
```

Usage in VBScript

```
InPlaceControl_object.node;
```

Example

```
<MACRO name="CGM_OnSynchronize" lang="JScript">
// XMetaL Script Language JSCRIPT:
var aipc = Application.ActiveInPlaceControl;
var domnode = aipc.node;
// ...
</MACRO>
```

ProgID

This property is valid only inside the OnShouldCreate, OnInitialize, OnSynchronize, amd OnFocus macros. This property is set to ActiveX ProgID for the control being used in-place.

Applies To

XMetaL Author and XMetaL XMAX

Access

Read only

Returns

String

Usage in JScript

```
InPlaceControl_object.ProgID;
```

Usage in VBScript

```
InPlaceControl_object.ProgrID
```

Example

```
<MACRO name="Generic_OnInitialize" key="" lang="JScript">
// Check to see which ActiveX control is being used

var aipc = Application.ActiveInPlaceControl;
if (aipc != null && aipc.ProgID == "Shell.Explorer.2") {
// MSIE WebBrowser control
aipc.Control.Navigate("about:blank");
}
<MACRO>
```

ShouldCreate

This property is valid only inside the OnShouldCreate macro. If this property is set to True (the default), the macro creates the in-place control. If set to False the control is not created and the element in question has its assigned Treat As behavior (Paragraph, List, Image, General).

Applies to

XMetaL Author and XMetaL XMAX

Access

Write-only

Returns

Boolean

Usage in JScript

```
InPlaceControl_object.ShouldCreate;
```

Usage in VBScript

```
InPlaceControl_object.ShouldCreate
```

Example

```
<MACRO name="CGM_OnShouldCreate" key="" lang="JScript">
// Check if InlineGraphic's FileRef is *.cgm,
// otherwise use default to XMetaL image handling. var aipc =
Application.ActiveInPlaceControl;
if (aipc != null) {
// Only create for FileRefs with .cgm extensions
aipc.ShouldCreate = false;
var domnode = aipc.Node;
var attrnode = domnode.attributes.getNamedItem("FileRef");
if (attrnode != null && attrnode.value != null) {
var i = attrnode.value.lastIndexOf(".cgm");
if (i != -1) {
aipc.ShouldCreate = true; // Has .cgm extension!
}
}
}
<MACRO>
```


UpdateFromDocument

This property is available only inside the OnSynchronize macro. It returns True if the control may need to have its properties updated, based on the (possibly) changed state of the document. Returns False if the document may need its content or markup changed, based on the (possibly) changed state of the control.

Applies to

XMetaL Author and XMetaL XMAX

Access

Read only

Returns

Boolean

Usage in JScript

```
InPlaceControl_object.UpdateFromDocument;
```

Usage in VBScript

```
InPlaceControl_object.UpdateFromDocument
```

Example

```
<MACRO name="CGM_OnSynchronize" lang="JScript">
// XMetaL Script Language JSCRIPT:
var aipc = Application.ActiveInPlaceControl;
var isoview = aipc.Control;
var domnode = aipc.Node;

// If "UpdateFromDocument", then may need to
// update the control
// from the document's latest attribute values.
if (aipc.UpdateFromDocument) {
var attrnode = domnode.attributes.getNamedItem("FileRef");
if (attrnode != null) {
isoview.FileName = attrnode.value;
} else {
isoview.FileName = "";
}
}
</MACRO>
```

userData

If an in-place control was specified using the `DOMDocumentType.addInPlaceControlOverride` method, and its `strData` argument was specified, this property can be used to obtain the value of that argument. It is available to all macros.

Applies to

XMetaL Author and XMetaL XMAX

Access

Read only

Returns

String

Usage in JScript

```
vbl = InPlaceControl_object.userData;
```

Usage in VBScript

```
vbl = InPlaceControl_object.userData
```

UserMovedIntoControl

This property is available only inside the OnFocus macro. It returns True if the user has moved from the document into the in-place control, thereby enabling to you modify the control's interface, for example, by turning on special UI features. Returns False if the user has moved out of the in-place control (by manually moving the insertion point, switching applications, or switching to Plain Text view); this enables you to you modify the control's interface.

Applies to

XMetaL Author and XMetaL XMAX

Access

Read only

Returns

Boolean

Usage in JScript

```
InPlaceControl_object.UserMovedIntoControl
```

Usage in VBScript

```
InPlaceControl_object.UserMovedIntoControl
```

Example

```
<MACRO name="CGM_OnFocus" lang="JScript">
// XMetaL Script Language JSCRIPT:
var aipc = Application.ActiveInPlaceControl;
var isoview = aipc.Control;
var domnode = aipc.Node;

// If "UserMovedIntoControl", then may need to
// update the control
if (aipc.UserMovedIntoControl) {
// Show About box when user moves into control
isoView.AboutBox;
```



```
}
</MACRO>
```

Width

Gets or sets the width of the control, in pixels. This property is not valid inside the OnShouldCreate and OnSynchronize macros. It should be set only in the OnInitialize and OnFocus macros.

Applies To

XMetaL Author and XMetaL XMAX

Access

Read/write

Returns

Integer

Usage in JScript

```
var ctrlX=InPlaceControl_object.Width;
InPlaceControl_object.Width = numPixels;
```

Usage in VBScript

```
Dim ctrlY=InPlaceControl_object.Width
InPlaceControl_object.Width= numPixels
```

Example

```
<MACRO name="CGM_OnFocus" lang="JScript">
var aipc = Application.ActiveInPlaceControl;
var domnode = aipc.Node
// ...code deleted....
// If the user moved into the control
if (aipc.UserMovedIntoControl) {
var attrnode = domnode.attributes.getNamedItem("Width");
// If the width in the control is different from
// the width in the document then update the control
if (attrnode != null && attrnode.value != aipc.Width) {
var rg = attrnode.ownerDocument.Range;
aipc.Width = rg.ContainerAttribute("Width");
}
}
// ...code deleted...
<MACRO>
```

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y