[MS-DOM1X]:

Microsoft XML Document Object Model (DOM) Level 1 Standards Support

This document provides a statement of support for protocol implementations. It is intended for use in conjunction with the Microsoft protocol technical specifications, publicly available standard specifications, network programming art, and Microsoft distributed systems concepts. It assumes that the reader is either familiar with the aforementioned material or has immediate access to it.

A protocol conformance document does not require the use of Microsoft programming tools or programming environments in order to implement the protocols in the system. Developers who have access to Microsoft programming tools and environments are free to take advantage of them.

specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

This document describes the choices made when implementing the <*target name*> protocol. It identifies ambiguities and implementor choices and indicates the approach taken in the implementation. These details of the protocols are described in the protocol specifications for each of the protocols and data structures not in this document.

Revision Summary

Date	Revision History	Revision Class	Comments
3/17/2010	0.1	New	Released new document.
3/26/2010	1.0	None	Introduced no new technical or language changes.
5/26/2010	1.2	None	Introduced no new technical or language changes.
9/8/2010	1.3	Major	Significantly changed the technical content.
2/10/2011	2.0	None	Introduced no new technical or language changes.
2/22/2012	3.0	Major	Significantly changed the technical content.
7/25/2012	3.1	Minor	Clarified the meaning of the technical content.
6/26/2013	4.0	Major	Significantly changed the technical content.
3/31/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
1/22/2015	5.0	Major	Updated for new product version.
7/7/2015	5.1	Minor	Clarified the meaning of the technical content.
11/2/2015	5.1	None	No changes to the meaning, language, or formatting of the technical content.
3/22/2016	5.2	Minor	Clarified the meaning of the technical content.
11/2/2016	5.2	None	No changes to the meaning, language, or formatting of the technical content.
3/14/2017	5.2	None	No changes to the meaning, language, or formatting of the technical content.
10/3/2017	5.2	None	No changes to the meaning, language, or formatting of the technical content.
2/22/2018	5.2	None	No changes to the meaning, language, or formatting of the technical content.
3/23/2018	5.2	None	No changes to the meaning, language, or formatting of the technical content.
8/28/2018	5.2	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document describes the level of support provided by the Microsoft XML Core Services (MSXML) 3.0 and 6.0 for the *Document Object Model (DOM) Level 1 Specification Version 1.0* [DOM Level 1], published 1 October 1998.

The specification may contain guidance for authors of webpages and browser users, in addition to user agents (browser applications). Statements found in this document apply only to normative requirements in the specification targeted to user agents, not those targeted to authors.

1.1 Glossary

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[DOM Level 1] World Wide Web Consortium, "Document Object Model (DOM) Level 1 Specification Version 1.0", W3C Recommendation 1 October 1998, http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

1.2.2 Informative References

[MS-XPATH] Microsoft Corporation, "Microsoft XML XPath Standards Support Document".

[W3C-XSLT] World Wide Web Consortium, "XSL Transformations (XSLT) Version 1.0", W3C Recommendation 16 November 1999, http://www.w3.org/TR/1999/REC-xslt-19991116

[XPATH] Clark, J. and DeRose, S., "XML Path Language (XPath), Version 1.0", W3C Recommendation, November 1999, http://www.w3.org/TR/1999/REC-xpath-19991116/

1.3 Microsoft Implementations

Throughout this document, Microsoft XML Core Services (MSXML) 3.0 is referred to as *MSXML3* and Microsoft XML Core Services (MSXML) 6.0 is referred to as *MSXML6*.

MSXML3 is the only version of MSXML that is implemented in Windows Internet Explorer 7 and Windows Internet Explorer 8. Both MSXML3 and MSXML6 are implemented in Windows Internet Explorer 9, Windows Internet Explorer 10, Internet Explorer 11, and Internet Explorer 11 for Windows 10. MSXML3 is used in IE7 Mode and IE8 Mode, and MSXML6 is used in all other modes. MSXML6 is the only version of MSXML implemented in Microsoft Edge, which uses it only to implement XSLT

[W3C-XSLT]. Microsoft Edge provides [XPATH] functionality natively; see [MS-XPATH] for more information.

1.4 Standards Support Requirements

To conform to [DOM Level 1], a user agent must implement all required portions of the specification. Any optional portions that have been implemented must also be implemented as described by the specification. Normative language is usually used to define both required and optional portions. (For more information, see [RFC2119].)

The following table lists the sections of [DOM Level 1] and whether they are considered normative or informative.

Chapters	Normative/Informative
1-2	Normative
Appendices A-E	Informative

1.5 Notation

The following notations are used in this document to differentiate between notes of clarification, variation from the specification, and extension points.

Notation	Explanation
C####	This identifies a clarification of ambiguity in the target specification. This includes imprecise statements, omitted information, discrepancies, and errata. This does not include data formatting clarifications.
V####	This identifies an intended point of variability in the target specification such as the use of MAY, SHOULD, or RECOMMENDED. (See [RFC2119].) This does not include extensibility points.
E####	Identifies extensibility points (such as optional implementation-specific data) in the target specification, which can impair interoperability.

2 Standards Support Statements

This section contains all variations and clarifications for the Microsoft implementation of [DOM Level 1].

- Section 2.1 describes normative variations from the MUST requirements of the specification.
- Section 2.2 describes clarifications of the MAY and SHOULD requirements.
- Section 2.3 considers error handling aspects of the implementation.
- Section 2.4 considers security aspects of the implementation.

2.1 Normative Variations

There are no normative variations from the MUST requirements of [DOM Level 1].

2.2 Clarifications

The following subsections describe clarifications of the MAY and SHOULD requirements of [DOM Level 1].

2.2.1 [DOM Level 1] Section 1.1.5, The DOMString type

C0001:

The specification states:

```
A DOMString is a sequence of 16-bit quantities. This may be expressed in IDL terms as:
typedef sequence<unsigned short> DOMString;
```

MSXML3 and MSXML6

The **DOMString** data type is not supported; instead, the **BSTR** data type is supported.

2.2.2 [DOM Level 1] Section 1.2, Fundamental Interfaces

C0002:

The specification states:

```
Some languages and object systems do not support the concept of exceptions.
For such systems, error conditions may be indicated using native error reporting
mechanisms. For some bindings, for example, methods may return error codes similar
to those listed in the corresponding method descriptions.
```

MSXML3 and MSXML6

The **DOMException** interface is not supported; instead, an **HRESULT** value is supported.

C0003:

The specification states:

```
IDL Definition
```

```
exception DOMException {
  unsigned short   code;
};
```

MSXML3 and MSXML6

The **DOMException** interface is not supported; instead, an **HRESULT** value is supported.

C0004:

The specification states:

```
hasFeature Method
Test if the DOM implementation implements a specific feature.

Parameters
feature  The package name of the feature to test. In Level 1, the legal values are
"HTML" and "XML" (case-insensitive).
version  This is the version number of the package name to test. In Level 1, this
is the string "1.0". If the version is not specified, supporting any version of the
feature will cause the method to return true.

Return Value
true if the feature is implemented in the specified version, false otherwise.

This method raises no exceptions.
```

MSXML3 and MSXML6

The **hasFeature** method of the **DOMException** exception does not return `true` when the version parameter is not specified even if the feature is supported.

For example, the following method call returns `false`:

```
hasFeature("XML", "");
```

However, the following method call returns `true`:

```
hasFeature("XML", "1.0");
```

E0001:

The specification states:

```
interface Document : Node {
  readonly attribute  DocumentType         doctype;
  readonly attribute  DOMImplementation    implementation;
  readonly attribute  Element              documentElement;
  Element                   createElement(in DOMString tagName)
                                          raises(DOMException);
  DocumentFragment          createDocumentFragment();
  Text                      createTextNode(in DOMString data);
  Comment                   createComment(in DOMString data);
  CDATASection              createCDATASection(in DOMString data)
                                                raises(DOMException);
  ProcessingInstruction     createProcessingInstruction(in DOMString target,
                                                         in DOMString data)
```

```
                                                                  raises(DOMException);
      Attr                      createAttribute(in DOMString name)
                                                raises(DOMException);
      EntityReference           createEntityReference(in DOMString name)
                                                      raises(DOMException);
      NodeList                  getElementsByTagName(in DOMString tagname);
    };
```

MSXML3 and MSXML6

The following additional attributes are supported on the **Document** interface:

- **async**
- **parseError**
- **preserveWhiteSpace**
- **readyState**
- **resolveExternals**
- **url**
- **validateOnParse**

The following additional methods are supported on the **Document** interface:

- **abort**
- **createNode**
- **load**
- **loadXML**
- **nodeFromID**
- **save**

The following additional events are supported on the **Document** interface:

- **ondataavailable**
- **onreadystatechange**
- **ontransformnode**

The **documentElement** attribute is read/write, not read-only.

C0005:

The specification states:

```
Attribute
documentElement
This is a convenience attribute that allows direct access to the child node that is
the root element of the document. For HTML documents, this is the element with the
tagName "HTML".
```

MSXML3 and MSXML6

The value of the **documentElement** attribute of the **Document** interface is not read-only.

E0002:

The specification states:

IDL Definition

```
interface Node {
  // NodeType
  const unsigned short      ELEMENT_NODE       = 1;
  const unsigned short      ATTRIBUTE_NODE     = 2;
  const unsigned short      TEXT_NODE          = 3;
  const unsigned short      CDATA_SECTION_NODE = 4;
  const unsigned short      ENTITY_REFERENCE_NODE = 5;
  const unsigned short      ENTITY_NODE        = 6;
  const unsigned short      PROCESSING_INSTRUCTION_NODE = 7;
  const unsigned short      COMMENT_NODE       = 8;
  const unsigned short      DOCUMENT_NODE      = 9;
  const unsigned short      DOCUMENT_TYPE_NODE = 10;
  const unsigned short      DOCUMENT_FRAGMENT_NODE = 11;
  const unsigned short      NOTATION_NODE      = 12;

  readonly attribute  DOMString            nodeName;
           attribute  DOMString            nodeValue;
                                           // raises(DOMException) on setting
                                           // raises(DOMException) on retrieval
  readonly attribute  unsigned short       nodeType;
  readonly attribute  Node                 parentNode;
  readonly attribute  NodeList             childNodes;
  readonly attribute  Node                 firstChild;
  readonly attribute  Node                 lastChild;
  readonly attribute  Node                 previousSibling;
  readonly attribute  Node                 nextSibling;
  readonly attribute  NamedNodeMap         attributes;
  readonly attribute  Document             ownerDocument;
  Node                      insertBefore(in Node newChild,
                                         in Node refChild)
                                         raises(DOMException);
  Node                      replaceChild(in Node newChild,
                                         in Node oldChild)
                                         raises(DOMException);
  Node                      removeChild(in Node oldChild)
                                        raises(DOMException);
  Node                      appendChild(in Node newChild)
                                        raises(DOMException);
  boolean                   hasChildNodes();
  Node                      cloneNode(in boolean deep);
};
```

MSXML3 and MSXML6

The following additional attributes are supported on the **Node** interface:

- **baseName**
- **dataType**
- **definition**
- **namespaceUri**
- **nodeTypeString**
- **nodeTypedValue**

- **parsed**
- **prefix**
- **specified**
- **text**
- **xml**

The following additional methods are supported on the **Node** interface:

- **selectNodes**
- **selectSingleNode**
- **transformNode**
- **transformNodeToObject**

`NODE_xxx` values are defined instead of `xxx_NODE` values. The **NodeType** constants are replaced with a **DOMNodeType** enumeration.

```
enum tagDOMNodeType
{
    NODE_INVALID, // = 0
    NODE_ELEMENT, // = 1
    NODE_ATTRIBUTE, // = 2
    NODE_TEXT, // = 3
    NODE_CDATA_SECTION, // = 4
    NODE_ENTITY_REFERENCE, // = 5
    NODE_ENTITY, // = 6
    NODE_PROCESSING_INSTRUCTION, // = 7
    NODE_COMMENT, // = 8
    NODE_DOCUMENT, // = 9
    NODE_DOCUMENT_TYPE, // = 10
    NODE_DOCUMENT_FRAGMENT, // = 11
    NODE_NOTATION // = 12
} DOMNodeType;
```

C0006:

The specification states:

```
Attribute
nodeValue
The value of this node, depending on its type.
```

MSXML3 and MSXML6

The **DOMSTRING_SIZE_ERR** error code is not implemented. The value of the **nodeValue** attribute of the **Node** interface is constrained only by the size limit of physical memory.

C0007:

The specification states:

```
Attribute
```

```
attributes
A NamedNodeMap containing the attributes of this node (if it is an Element) or null
otherwise.
```

MSXML3 and MSXML6

The **attributes** attribute returns an **IXMLDOMNamedNodeMap** object that contains a collection of attributes for nodes of types:

- **Element**
- **DocumentType**
- **Entity**
- **Notation**

For nodes of type **ProcessingInstruction,** the value returned by the **attributes** attribute is null except for the XML declaration that appears as the first line in an XML document, for example `<?xml version="1.0" encoding="UTF-8"?>`.

C0008:

The specification states:

```
Methods
insertBefore
Inserts the node newChild before the existing child node refChild. If refChild is
null, insert newChild at the end of the list of children.
If newChild is a DocumentFragment object, all of its children are inserted, in the
same order, before refChild. If the newChild is already in the tree, it is first removed.

Parameters
newChild  The node to insert.
refChild  The reference node, i.e., the node before which the new node must be inserted.

Return Value
The node being inserted.

Exceptions
DOMException
HIERARCHY_REQUEST_ERR: Raised if this node is of a type that does not allow
children of the type of the newChild node, or if the node to insert is one of this
node's ancestors.
WRONG_DOCUMENT_ERR: Raised if newChild was created from a different document than
the one that created this node.
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
NOT_FOUND_ERR: Raised if refChild is not a child of this node.
```

MSXML3 and MSXML6

With the **insertBefore** method of the **Node** interface, the **WRONG_DOCUMENT_ERR** error is not raised if the new child node (**newChild**) was created in a different document than the one that contains the reference node (**refChild**). The **newChild** and **refChild** parameters can represent nodes in the same document or in different documents.

C0009:

The specification states:

```
Method
```

```
replaceChild
Replaces the child node oldChild with newChild in the list of children, and returns
the oldChild node. If the newChild is already in the tree, it is first removed.

Parameters
newChild  The new node to put in the child list.
oldChild  The node being replaced in the list.

Return Value
The node replaced.

Exceptions
DOMException
HIERARCHY_REQUEST_ERR: Raised if this node is of a type that does not allow
children of the type of the newChild node, or it the node to put in is one of this
node's ancestors.
WRONG_DOCUMENT_ERR: Raised if newChild was created from a different document than
the one that created this node.
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
NOT_FOUND_ERR: Raised if oldChild is not a child of this node.
```

MSXML3 and MSXML6

With the **replaceChild** method of the **Node** interface, the **WRONG_DOCUMENT_ERR** error is not raised if the new node (**newChild**) was created in a different document than the one that contains the reference node (**refChild**). The **newChild** and **refChild** parameters can represent nodes in the same document or in different documents.

C0010:

The specification states:

```
Method
appendChild
Adds the node newChild to the end of the list of children of this node. If the
newChild is already in the tree, it is first removed.

Parameters
newChild  The node to add.
If it is a DocumentFragment object, the entire contents of the document fragment
are moved into the child list of this node

Return Value
The node added.

Exceptions
DOMException
HIERARCHY_REQUEST_ERR: Raised if this node is of a type that does not allow
children of the type of the newChild node, or if the node to append is one of this
node's ancestors.
WRONG_DOCUMENT_ERR: Raised if newChild was created from a different document than
the one that created this node.
```

MSXML3 and MSXML6

With the **appendChild** methods of the **Node** interface, the **WRONG_DOCUMENT_ERR** error is not raised if the **newChild** was created in a different document than the one to which it is being appended.

E0003:

The specification states:

```
IDL Definition
interface NodeList {
  Node                      item(in unsigned long index);
  readonly attribute  unsigned long        length;
};
```

MSXML3 and MSXML6

The **NodeList** interface supports the **item** Method.

The additional following methods are supported:

- **nextNode**
- **reset**

The **length** attribute and **index** of **item** method is of type long instead of unsigned long.

E0004:

The specification states:

```
IDL Definition
interface NamedNodeMap {
  Node                      getNamedItem(in DOMString name);
  Node                      setNamedItem(in Node arg)
                                         raises(DOMException);
  Node                      removeNamedItem(in DOMString name)
                                            raises(DOMException);
  Node                      item(in unsigned long index);
  readonly attribute  unsigned long        length;
};
```

MSXML3 and MSXML6

The **IXMLDOMNamedNodeMap** interface contains four additional methods:

- **getQualifiedItem**
- **removeQualifiedItem**
- **nextNode**
- **reset**

The **length** attribute is of type long instead of unsigned long.

C0012:

The specification states:

```
Method
setNamedItem
Adds a node using its nodeName attribute.
As the nodeName attribute is used to derive the name which the node must be stored
under, multiple nodes of certain types (those that have a "special" string value)
cannot be stored as the names would clash. This is seen as preferable to allowing
nodes to be aliased.

Parameters
arg  A node to store in a named node map. The node will later be accessible using
```

```
the value of the nodeName attribute of the node. If a node with that name is
already present in the map, it is replaced by the new one.

Return Value
If the new Node replaces an existing node with the same name the previously
existing Node is returned, otherwise null is returned.

Exceptions
DOMException
WRONG_DOCUMENT_ERR: Raised if arg was created from a different document than the
one that created the NamedNodeMap.
NO_MODIFICATION_ALLOWED_ERR: Raised if this NamedNodeMap is readonly.
INUSE_ATTRIBUTE_ERR: Raised if arg is an Attr that is already an attribute of
another Element object. The DOM user must explicitly clone
```

MSXML3 and MSXML6

The following variations apply:

- The **setNamedItem** method of the **NamedNodeMap** interface returns the attribute that is successfully added to the collection. If the new Node replaces an existing Node with the same name, the new Node will be returned instead of the previously existing Node.
- With the **setNamedItem** method of the **Node** interface, the **WRONG_DOCUMENT_ERR** error is not raised if the new node (**arg**) was created in a different document than the one that contains the previously existing node. The new node and previously existing node can represent nodes in the same document or in different documents.

C0013:

The specification states:

```
Method
removeNamedItem
Removes a node specified by name. If the removed node is an Attr with a default
value it is immediately replaced.

Parameters
name  The name of a node to remove.

Return Value
The node removed from the map or null if no node with such a name exists.

Exceptions
DOMException
NOT_FOUND_ERR: Raised if there is no node named name in the map.
```

MSXML3 and MSXML6

The **removeNamedItem** method of the **NamedNodeMap** interface returns null if there is no node for the specified name; the expected NOT_FOUND_ERR exception is never thrown.

C0014:

The specification states:

```
IDL Definition
interface CharacterData : Node {
           attribute  DOMString            data;
                                 // raises(DOMException) on setting
                                 // raises(DOMException) on retrieval
  readonly attribute  unsigned long        length;
```

```
    DOMString                 substringData(in unsigned long offset,
                                            in unsigned long count)
                                            raises(DOMException);
    void                      appendData(in DOMString arg)
                                         raises(DOMException);
    void                      insertData(in unsigned long offset,
                                         in DOMString arg)
                                         raises(DOMException);
    void                      deleteData(in unsigned long offset,
                                         in unsigned long count)
                                         raises(DOMException);
    void                      replaceData(in unsigned long offset,
                                          in unsigned long count,
                                          in DOMString arg)
                                          raises(DOMException);
  };
```

MSXML3 and MSXML6

The **length** attribute of the **CharacterData** interface is of type `long` instead of `unsigned long`. Long integers are used for the **offset** and **count** parameters in the following methods:

- **substringData**
- **insertData**
- **deleteData**
- **replaceData**

C0015:

The specification states:

```
Attribute
data
The character data of the node that implements this interface. The DOM
implementation may not put arbitrary limits on the amount of data that may be
stored in a CharacterData node. However, implementation limits may mean that the
entirety of a node's data may not fit into a single DOMString. In such cases, the
user may call substringData to retrieve the data in appropriately sized pieces.

Exceptions on setting
DOMException
NO_MODIFICATION_ALLOWED_ERR: Raised when the node is readonly.

Exceptions on retrieval
DOMException
DOMSTRING_SIZE_ERR: Raised when it would return more characters than fit in a
DOMString variable on the implementation platform.
```

MSXML3 and MSXML6

The value for the **data** attribute of the **CharacterData** interface is of type `BSTR`. The **DOMSTRING_SIZE_ERR** error code is not implemented. The value of the **data** attribute is constrained only by the size limit of physical memory.

C0016:

The specification states:

```
Method
```

```
substringData
Extracts a range of data from the node.

Parameters
offset  Start offset of substring to extract.
count  The number of characters to extract.

Return Value
The specified substring. If the sum of offset and count exceeds the length, then
all characters to the end of the data are returned.

Exceptions
DOMException
INDEX_SIZE_ERR: Raised if the specified offset is negative or greater than the
number of characters in data, or if the specified count is negative.
DOMSTRING_SIZE_ERR: Raised if the specified range of text does not fit into a DOMString.
```

MSXML3 and MSXML6

The return value for the **substringData** method of the **CharacterData** interface is of type `BSTR`. The **DOMSTRING_SIZE_ERR** error code is not implemented. The **BSTR** data type is constrained only by the size limit of physical memory.

C0017:

The specification states:

```
IDL Definition
interface Attr : Node {
  readonly attribute  DOMString            name;
  readonly attribute  boolean              specified;
           attribute  DOMString            value;
};
```

MSXML3 and MSXML6

The following clarifications apply:

- The **value** attribute of the **CharacterData** interface returns a value of type `VARIANT`.
- The **specified** attribute value is of type `VARIANT_BOOL`.

C0018:

The specification states:

```
IDL Definition
interface Element : Node {
readonly attribute DOMString tagName;
DOMString getAttribute(in DOMString name);
void setAttribute(in DOMString name,
                                         in DOMString value)
                                         raises(DOMException);
void removeAttribute(in DOMString name)
                                            raises(DOMException);
Attr getAttributeNode(in DOMString name);
Attr setAttributeNode(in Attr newAttr)
                                             raises(DOMException);
                   Attr  removeAttributeNode(in Attr oldAttr)
                                                raises(DOMException);
NodeList getElementsByTagName(in DOMString name);
void normalize();
};
```

MSXML3 and MSXML6

The following clarifications apply:

- The **getAttribute** method of the **Element** interface returns a value of type `VARIANT`.
- The **setAttribute** method requires that the **value** attribute be of type `VARIANT`.

C0019:

The specification states:

```
Method
getAttribute
Retrieves an attribute value by name.

Parameters
name  The name of the attribute to retrieve.

Return Value
The Attr value as a string, or the empty string if that attribute does not have a
specified or default value.

This method raises no exceptions.
```

MSXML3 and MSXML6

The **getAttribute** method of the **Element** interface returns null if the attribute does not have a specified or default value.

C0020:

The specification states:

```
Method
setAttributeNode
Adds a new attribute. If an attribute with that name is already present in the
element, it is replaced by the new one.

Parameters
newAttr  The Attr node to add to the attribute list.

Return Value
If the newAttr attribute replaces an existing attribute with the same name, the
previously existing Attr node is returned, otherwise null is returned.

Exceptions
DOMException
WRONG_DOCUMENT_ERR: Raised if newAttr was created from a different document than
the one that created the element.
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
INUSE_ATTRIBUTE_ERR: Raised if newAttr is already an attribute of another Element
object. The DOM user must explicitly clone Attr nodes to re-use them in other elements.
```

MSXML3 and MSXML6

With the **setAttributeNode** method of the **Element** interface, the **WRONG_DOCUMENT_ERR** exception is not raised if the new node (which is specified by the **newAttr** parameter) is created in a different document than the one that created the element. The **newAttr** node and element can represent nodes in the same document or in different documents.

C0021:

The specification states:

```
IDL Definition
interface Text : CharacterData {
  Text                      splitText(in unsigned long offset)
                            raises(DOMException);
};
```

MSXML3 and MSXML6

For the **splitText** method of the **Text** interface, the **offset** parameter is of type `long integer` instead of an `unsigned long integer`.

C0022:

The specification states:

```
Method
splitText Breaks this Text node into two Text nodes at the specified offset,
keeping both in the tree as siblings. This node then only contains all the content
up to the offset point. And a new Text node, which is inserted as the next sibling
of this node, contains all the content at and after the offset point.

Parameters
offset The offset at which to split, starting from 0.

Return Value
The new Text node.
```

MSXML3 and MSXML6

The **splitText** method returns null when the **offset** parameter contains a value equal to the length of the **Text** node.

2.2.3 [DOM Level 1] Section 1.3, Extended Interfaces

C0023:

The specification states:

```
A Notation node does not have any parent.
```

MSXML3 and MSXML6

The parent of **Notation** node is the **DocumentType** node that contains the **Notation** node.

C0024:

The specification states:

```
IDL Definition
interface Notation : Node {
  readonly attribute  DOMString            publicId;
  readonly attribute  DOMString            systemId;
};
```

MSXML3 and MSXML6

The values of the **publicId** and **systemId** attributes of the **Notation** interface are of type `VARIANT` instead of a string type.

C0025:

The specification states:

```
Attribute
publicId
The public identifier of this notation. If the public identifier was not specified,
this is null.
```

MSXML3 and MSXML6

If the public identifier is not specified, the **publicId** attribute of the **Notation** interface returns an empty string.

C0026:

The specification states:

```
Attribute
systemId
The system identifier of this notation. If the system identifier was not specified,
this is null.
```

MSXML3 and MSXML6

If the public identifier is not specified, the **systemId** attribute of the **Notation** interface returns an empty string.

C0027:

The specification states:

```
An XML processor may choose to completely expand entities before the structure
model is passed to the DOM; in this case there will be no EntityReference nodes in
the document tree.
```

MSXML3 and MSXML6

Instead of providing **EntityReference** nodes in the **Entity** interface, the entities are completely expanded before the structure model is passed to the DOM.

C0028:

The specification states:

```
An Entity node does not have any parent.
```

MSXML3 and MSXML6

The parent of **Entity** node is the **DocumentType** node that contains the **Entity** node.

C0029:

The specification states:

```
IDL Definition
interface Notation : Node {
  readonly attribute  DOMString            publicId;
  readonly attribute  DOMString            systemId;
  readonly attribute  DOMString            notationName;
};
```

MSXML3 and MSXML6

The **publicId** and **systemId** attribute values are of type `VARIANT`.

C0030:

The specification states:

```
Attribute
publicId
The public identifier associated with the entity, if specified. If the public
identifier was not specified, this is null.
```

MSXML3 and MSXML6

If the public identifier is not specified, **publicId** returns an empty string.

C0031:

The specification states:

```
Interface Notation

Attribute
systemId
The system identifier associated with the entity, if specified. If the system
identifier was not specified, this is null.
```

MSXML3 and MSXML6

If the system identifier is not specified, instead of returning null, **systemId** returns an empty string.

C0032:

The specification states:

```
Attribute
notationName
For unparsed entities, the name of the notation for the entity. For parsed
entities, this is null.
```

MSXML3 and MSXML6

For parsed entities the **notationName** attribute of the **Entity** interface returns an empty string.

C0033:

The specification states:

```
Attribute
data
The content of this processing instruction. This is from the first non white space
character after the target to the character immediately preceding the ?>.

Exceptions on setting
DOMException
NO_MODIFICATION_ALLOWED_ERR: Raised when the node is readonly.
```

MSXML3 and MSXML6

The **data** attribute of the **ProcessingInstruction** interface contains the content from the first non–white-space character that follows the `target` declaration to the last non–white-space character that precedes the closing `?>` characters.

2.2.4 [DOM Level 1] Section 2.1, Introduction

C0034:

The specification states:

```
This section extends the Level 1 Core API to describe objects and methods specific
to HTML documents. In general, the functionality needed to manipulate hierarchical
document structures, elements, and attributes will be found in the core section;
functionality that depends on the specific elements defined in HTML will be found
in this section.
```

MSXML3 and MSXML6

None of the interfaces in section 2 of the [DOM Level 1] specification are supported.

2.3 Error Handling

There are no additional error handling considerations.

2.4 Security

There are no additional security considerations.

3 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

4 Index