[MS-DOM2CX]:

Microsoft XML Document Object Model (DOM) Level 2 Core 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.

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards

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 implementer 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	5.2	Minor	Clarified the meaning of the technical content.
3/22/2016	5.3	Minor	Clarified the meaning of the technical content.
11/2/2016	5.3	None	No changes to the meaning, language, or formatting of the technical content.
3/14/2017	5.3	None	No changes to the meaning, language, or formatting of the technical content.
10/3/2017	5.3	None	No changes to the meaning, language, or formatting of the technical content.
2/22/2018	5.3	None	No changes to the meaning, language, or formatting of the technical content.
3/23/2018	5.3	None	No changes to the meaning, language, or formatting of the technical content.
8/28/2018	5.3	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 2 Core Specification Version 1.0* [DOM Level 2 - Core], W3C Recommendation 13 November, 2000.

The [DOM Level 2 - Core] 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 2 - Core] World Wide Web Consortium, "Document Object Model (DOM) Level 2 Core Specification Version 1.0", W3C Recommendation 13 November 2000, http://www.w3.org/TR/DOM-Level-2-Core/

[NamespacesXML1.1] World Wide Web Consortium, "Namespaces in XML 1.1 (Second Edition)", W3C Recommendation 16 August 2006, http://www.w3.org/TR/xml-names11/

[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 2 - Core], 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 2 - Core] and whether they are considered normative or informative.

Section	Normative/Informative
1	Normative
Appendices A-F	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####	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####	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.

For document mode and browser version notation, see section 1.3.

2 Standards Support Statements

This section contains all variations and clarifications for the Microsoft implementation of [DOM Level 2 - Core].

- 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 2 - Core].

2.2 Clarifications

The following subsections describe clarifications of the MAY and SHOULD requirements of [DOM Level 2 - Core].

2.2.1 [DOM Level 2 - Core] Section 1.1.5, The DOMString type

C0001:

The specification states:

```
Type Definition DOMString
A DOMString is a sequence of 16-bit units.

IDL Definition
valuetype DOMString sequence<unsigned short>;
```

MSXML3 and MSXML6

The **DOMString** type is not defined. A BSTR value is used instead.

2.2.2 [DOM Level 2 - Core] Section 1.1.6, The DOMTimeStamp type

C0002:

The specification states:

```
•Type Definition DOMTimeStamp
A DOMTimeStamp represents a number of milliseconds.

IDL Definition
typedef unsigned long long DOMTimeStamp;

•Note: Even though the DOM uses the type DOMTimeStamp, bindings may use different types. For
example for Java, DOMTimeStamp is bound to the long type. In ECMAScript, TimeStamp is bound
to the Date type because the range of the integer type is too small.
```

MSXML3 and MSXML6

The **DOMTimeStamp** definition type is not defined, and there are no APIs that use parameters related to the **DOMTimeStamp**.

2.2.3 [DOM Level 2 - Core] Section 1.2, Fundamental Interfaces

C0003:

The specification states:

```
IDL Definition
exception DOMException {
  unsigned short   code;
};
// ExceptionCode
const unsigned short      INDEX_SIZE_ERR                 = 1;
const unsigned short      DOMSTRING_SIZE_ERR             = 2;
const unsigned short      HIERARCHY_REQUEST_ERR          = 3;
const unsigned short      WRONG_DOCUMENT_ERR             = 4;
const unsigned short      INVALID_CHARACTER_ERR          = 5;
const unsigned short      NO_DATA_ALLOWED_ERR            = 6;
const unsigned short      NO_MODIFICATION_ALLOWED_ERR    = 7;
const unsigned short      NOT_FOUND_ERR                  = 8;
const unsigned short      NOT_SUPPORTED_ERR              = 9;
const unsigned short      INUSE_ATTRIBUTE_ERR            = 10;
// Introduced in DOM Level 2:
const unsigned short      INVALID_STATE_ERR              = 11;
// Introduced in DOM Level 2:
const unsigned short      SYNTAX_ERR                     = 12;
// Introduced in DOM Level 2:
const unsigned short      INVALID_MODIFICATION_ERR       = 13;
// Introduced in DOM Level 2:
const unsigned short      NAMESPACE_ERR                  = 14;
// Introduced in DOM Level 2:
const unsigned short      INVALID_ACCESS_ERR             = 15;
```

MSXML3 and MSXML6

DOMException is not implemented; instead, HRESULT values are returned. The HRESULT code is very different in terms of error types and meanings. Windows terminology uses all uppercase for HRESULT.

V0001:

The specification states:

```
IDL Definition
interface DOMImplementation {
  boolean            hasFeature(in DOMString feature,
                                in DOMString version);
  // Introduced in DOM Level 2:
  DocumentType       createDocumentType(in DOMString qualifiedName,
                                        in DOMString publicId,
                                        in DOMString systemId)
                                        raises(DOMException);
  // Introduced in DOM Level 2:
  Document           createDocument(in DOMString namespaceURI,
                                    in DOMString qualifiedName,
                                    in DocumentType doctype)
                                        raises(DOMException);
};
```

MSXML3 and MSXML6

The **createDocumentType** and **createDocument** methods of the **DOMImplementation** interface are not supported.

C0004:

The specification states:

```
Method
hasFeature
Test if the DOM implementation implements a specific feature.
Parameters
feature of type DOMString
The name of the feature to test (case-insensitive). The values used by DOM features are
defined throughout the DOM Level 2 specifications and listed in the Conformance section. The
name must be an XML name. To avoid possible conflicts, as a convention, names referring to
features defined outside the DOM specification should be made unique by reversing the name of
the Internet domain name of the person (or the organization that the person belongs to) who
defines the feature, component by component, and using this as a prefix. For instance, the
W3C SVG Working Group defines the feature "org.w3c.dom.svg".

version of type DOMString
This is the version number of the feature to test. In Level 2, the string can be either "2.0"
or "1.0". If the version is not specified, supporting any version of the feature causes the
method to return true.

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

No Exceptions
```

MSXML3

The **hasFeature** method of the **DOMImplementation** interface supports only the following parameter values:

- **feature**= "`XML`"
- **version**="`1.0`"

The **version** parameter cannot be empty. For more information about the "`XML`" value, see [DOM Level 2 - Core], Conformance.

MSXML6

The **hasFeature** method of the **DOMImplementation** interface supports only the **feature**= "`XML`" parameter value.

C0005:

The specification defines the **Document** interface.

MSXML3 and MSXML6

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

- **async**
- **parseError**
- **preserveWhiteSpace**
- **readyState**
- **resolveExternals**

- **url**
- **validateOnParse**

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

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

The following **Document** interface events are supported:

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

The following methods are not supported:

- **importNode**
- **createElementNS**
- **createAttributeNS**
- **getElementsByTagNameNS**
- **getElementById**

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

C0006:

The specification states:

```
Interface Document

Attribute

documentElement of type Element, readonly
        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 **documentElement** attribute of the **Document** interface is read/write, not read-only.

C0007:

The specification defines the interface **node**.

MSXML3 and MSXML6

The following clarifications apply:

- The methods **isSupported**, **hasAttributes**, and **normalize** are not supported.
- The **normalize** method is defined in `Element`.
- The **localName** method is not supported, but **baseName** is supported for similar functionality.
- Additional attributes are supported:
  - **dataType**
  - **definition**
  - **nodeTypeString**
  - **nodeTypedValue**
  - **parsed**
  - **specified**
  - **text**
  - **xml**
- Additional methods are supported:
  - **selectNodes**
  - **selectSingleNode**
  - **transformNode**
  - **transformNodeToObject**
- The *nodeType* is defined as:

```
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;
```

  The names of node types are defined as **NODE_xxx** instead of **xxx_NODE**.

C0008:

The specification states:

```
attributes of type NamedNodeMap, readonly
```

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

MSXML3 and MSXML6

The **attributes** attribute of the **NamedNodeMap** interface may return the **IXMLDOMNamedNodeMap** object for the following constants:

- NODE_ELEMENT
- NODE_PROCESSING_INSTRUCTION
- NODE_DOCUMENT_TYPE
- NODE_ENTITY
- NODE_NOTATION

C0009:

The specification states:

```
Attribute

localName of type DOMString, readonly, introduced in DOM Level 2
        Returns the local part of the qualified name of this node.
        For nodes of any type other than ELEMENT_NODE and ATTRIBUTE_NODE and nodes created
with a DOM Level 1 method, such as createElement from the Document interface, this is always
null.
```

MSXML3 and MSXML6

The **baseName** attribute of the **Node** interface is supported and provides the same functionality as **localName**. However, **baseName** returns an empty string for nodes other than **element** and **attribute**.

C0010:

The specification states:

```
Attribute

namespaceURI of type DOMString, readonly, introduced in DOM Level 2
        The namespace URI of this node, or null if it is unspecified.
        This is not a computed value that is the result of a namespace lookup based on an
examination of the namespace declarations in scope. It is merely the namespace URI given at
creation time.
        For nodes of any type other than ELEMENT_NODE and ATTRIBUTE_NODE and nodes created
with a DOM Level 1 method, such as createElement from the Document interface, this is always
null.
        Note: Per the Namespaces in XML Specification [NamespacesXML1.1] an attribute does
not inherit its namespace from the element it is attached to. If an attribute is not
explicitly given a namespace, it simply has no namespace.
```

MSXML3 and MSXML6

The following clarifications apply:

- The **namespaceURI** attribute of the **Node** interface returns an empty string if the namespace URI of the node is not specified.

- The **namespaceURI** attribute returns an empty string for nodes other than **element** and **attribute**.

V0002:

The specification states:

```
Attribute

nodeValue of type DOMString
        The value of this node, depending on its type; see the table above. When it is
defined to be null, setting it has no effect.

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 **DOMSTRING_SIZE_ERR** exception for the **nodeValue** attribute of the **Node** interface is not returned because the only limit of a BSTR value is physical memory.

C0011:

The specification states:

```
Attribute
prefix of type DOMString, introduced in DOM Level 2
The namespace prefix of this node, or null if it is unspecified.
Note that setting this attribute, when permitted, changes the nodeName attribute, which holds
the qualified name, as well as the tagName and name attributes of the Element and Attr
interfaces, when applicable.
Note also that changing the prefix of an attribute that is known to have a default value,
does not make a new attribute with the default value and the original prefix appear, since
the namespaceURI and localName do not change.
For nodes of any type other than ELEMENT_NODE and ATTRIBUTE_NODE and nodes created with a DOM
Level 1 method, such as createElement from the Document interface, this is always null.

Exceptions on setting
        DOMException INVALID_CHARACTER_ERR: Raised if the specified prefix contains an
illegal character.
        NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
        NAMESPACE_ERR: Raised if the specified prefix is malformed, if the namespaceURI of
this node is null, if the specified prefix is "xml" and the namespaceURI of this node is
different from "http://www.w3.org/XML/1998/namespace", if this node is an attribute and the
specified prefix is "xmlns" and the namespaceURI of this node is different from
"http://www.w3.org/2000/xmlns/", or if this node is an attribute and the qualifiedName of
this node is "xmlns" [NamespacesXML1.1].
```

MSXML3 and MSXML6

The following clarifications apply:

- The **prefix** attribute of the **Node** interface is read-only. Any attempt to set a new value for **prefix** causes an exception.
- An empty string is returned if the namespace prefix of the node is not specified, or if the node is not an element or an attribute.

C0012:

The specification states:

```
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 of type Node
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
Node
 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.

NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
```

MSXML3 and MSXML6

The **appendChild** method of the **Node** interface can accept a **newChild** node that was created from a different document. A **WRONG_DOCUMENT_ERR** exception is not thrown in this case.

C0013:

The specification states:

```
replaceChild
Replaces the child node oldChild with newChild in the list of children, and returns the
oldChild node.
If newChild is a DocumentFragment object, oldChild is replaced by all of the DocumentFragment
children, which are inserted in the same order. If the newChild is already in the tree, it is
first removed.
Parameters
newChild of type Node
The new node to put in the child list.

oldChild of type Node
The node being replaced in the list.

Return Value
Node
 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 if 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 or the parent of the new node is readonly.
```

```
NOT_FOUND_ERR: Raised if oldChild is not a child of this node.
```

MSXML3 and MSXML6

The **replaceChild** method of the **Node** interface accepts a **newChild** node that was created from a different document. A **WRONG_DOCUMENT_ERR** exception is not thrown in this case.

E0001:

The specification states:

IDL Definition

```
interface NodeList {

  Node               item(in unsigned long index);

  readonly attribute unsigned long    length;

};
```

MSXML3 and MSXML6

The **NodeList** interface has two additional methods, **nextNode** and **reset**. The **length** attribute and **index** of the **item** method is of type `long` instead of `unsigned long`.

E0002:

The specification defines the **NamedNodeMap** interface.

MSXML3 and MSXML6

The following **NamedNodeMap** interface methods are not supported:

- **getNamedItemNS**
- **removeNamedItemNS**
- **setNamedItemNS**

The following additional **NamedNodeMap** interface methods are supported:

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

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

C0014:

The specification states:

```
removeNamedItem
Removes a node specified by name. When this map contains the attributes attached to an
element, if the removed attribute is known to have a default value, an attribute immediately
appears containing the default value as well as the corresponding namespace URI, local name,
and prefix when applicable.
```

```
Parameters
name of type DOMString
The nodeName of the node to remove.

Return Value
Node
 The node removed from this map if a node with such a name exists.


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

NO_MODIFICATION_ALLOWED_ERR: Raised if this map is readonly.
```

MSXML3 and MSXML6

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

C0015:

The specification states:

```
setNamedItem
Adds a node using its nodeName attribute. If a node with that name is already present in this
map, it is replaced by the new one.
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 of type Node
A node to store in this map. The node will later be accessible using the value of its
nodeName attribute.

Return Value
Node
 If the new Node replaces an existing node the replaced 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 this map.

NO_MODIFICATION_ALLOWED_ERR: Raised if this map 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 Attr nodes to re-use them in other elements.
```

MSXML3 and MSXML6

The **setNamedItem** method of the **NamedNodeMap** interface has the following behaviors:

- The method returns the node that was successfully added to the collection, not the previously existing node.
- The method accepts a node that was created by a different document, so it never throws the **WRONG_DOCUMENT_ERR** exception.

C0016:

The specification defines the **CharacterData** interface.

MSXML3 and MSXML6

The **length**, **offset**, and **count** attributes of the **CharacterData** interface are of type `long`, instead of `unsigned long`.

C0017:

The specification states:

```
data of type DOMString
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.


length of type unsigned long, readonly
The number of 16-bit units that are available through data and the substringData method
below. This may have the value zero, i.e., CharacterData nodes may be empty.
```

MSXML3 and MSXML6

The **data** attribute of the **CharacterData** interface does not return an error code in the same manner as the **DOMSTRING_SIZE_ERR** because the only limit of a BSTR value is physical memory.

C0018:

The specification states:

```
substringData
Extracts a range of data from the node.
Parameters
offset of type unsigned long
Start offset of substring to extract.

count of type unsigned long
The number of 16-bit units to extract.

Return Value
DOMString
 The specified substring. If the sum of offset and count exceeds the length, then all 16-bit
units 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 16-
bit units 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 **substringData** method of the **CharacterData** interface does not return an error code in the same manner as the **DOMSTRING_SIZE_ERR** because the only limit of a BSTR value is physical memory.

C0019:

The specification states:

```
IDL Definition
interface Attr : Node {
  readonly attribute DOMString        name;
  readonly attribute boolean          specified;
           attribute DOMString        value;
                                        // raises(DOMException) on setting

  // Introduced in DOM Level 2:
  readonly attribute Element          ownerElement;
};
```

MSXML3 and MSXML6

The following clarifications apply:

- The **ownerElement** attribute of the **Attr** interface is not supported.
- The type of the **value** attribute is `VARIANT`.
- The type of the **specified** attribute is `VARIANT_BOOL`.

C0020:

The specification defines the **Element** interface.

MSXML3 and MSXML6

The following methods of the **Element** interface are supported:

- **getAttribute**
- **getAttributeNode**
- **getElementsByTagName**
- **removeAttribute**
- **removeAttributeNode**
- **setAttribute**
- **setAttributeNode**

The following methods are not supported:

- **getAttributeNodeNS**
- **getAttributeNS**
- **getElementsByTagNameNS**
- **hasAttribute**

- **hasAttributeNS**
- **removeAttributeNS**
- **setAttributeNodeNS**
- **setAttributeNS**

The following clarifications apply:

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

C0021:

The specification states:

```
getAttribute
Retrieves an attribute value by name.
Parameters
name of type DOMString
The name of the attribute to retrieve.

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

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.

C0022:

The specification states:

```
setAttributeNode
Adds a new attribute node. If an attribute with that name (nodeName) is already present in
the element, it is replaced by the new one.
To add a new attribute node with a qualified name and namespace URI, use the
setAttributeNodeNS method.
Parameters
newAttr of type Attr
The Attr node to add to the attribute list.

Return Value
Attr
 If the newAttr attribute replaces an existing attribute, the replaced 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

The **setAttributeNode** method of the **Element** interface can accept new attribute node *newAttr* that was created from a different document. A **WRONG_DOCUMENT_ERR** error is not thrown in such case.

C0023:

The specification states:

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

MSXML3 and MSXML6

The **offset** parameter of the **splitText** method of the **Text** interface is of type `long`.

V0003:

The specification states:

```
Method

splitText
        Breaks this node into two nodes at the specified offset, keeping both in the tree as
siblings. After being split, this node will contain all the content up to the offset point. A
new node of the same type, which contains all the content at and after the offset point, is
returned. If the original node had a parent node, the new node is inserted as the next
sibling of the original node. When the offset is equal to the length of this node, the new
node has no data.

Parameters
offset of type unsigned long
        The 16-bit unit offset at which to split, starting from 0.

Return Value
Text        The new node, of the same type as this node.

Exceptions
DOMException
                INDEX_SIZE_ERR: Raised if the specified offset is negative or greater than
the number of 16-bit units in data.
                NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
```

MSXML3 and MSXML6

The **splitText** method of the **Text** interface returns null when the value of **offset** parameter is the length of the data.

2.2.4 [DOM Level 2 - Core] Section 1.3, Extended Interfaces

C0024:

The specification states:

```
IDL Definition
interface DocumentType : Node {
  readonly attribute DOMString        name;
  readonly attribute NamedNodeMap     entities;
  readonly attribute NamedNodeMap     notations;
  // Introduced in DOM Level 2:
  readonly attribute DOMString        publicId;
  // Introduced in DOM Level 2:
  readonly attribute DOMString        systemId;
  // Introduced in DOM Level 2:
  readonly attribute DOMString        internalSubset;
};
```

MSXML3 and MSXML6

The following attributes of the **DocumentType** interface are supported:

- **name**
- **entities**
- **notations**

The following attributes of the **DocumentType** interface are not supported:

- **internalSubset**
- **publicId**
- **systemId**

C0025:

The specification states:

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

MSXML3 and MSXML6

The parent of a **Notation** node is the **DocumentType** node.

V0004:

The specification states:

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

MSXML3 and MSXML6

The attributes **publicId** and **systemId** of the **Notation** interface are of type `VARIANT`.

C0026:

The specification states:

```
Attributes
publicId of type DOMString, readonly
The public identifier of this notation. If the public identifier was not specified, this is
null.
systemId of type DOMString, readonly
The system identifier of this notation. If the system identifier was not specified, this is
null.
```

MSXML3 and MSXML6

The **publicId** and **systemId** attributes of the **Notation** interface return an empty string if the public or system identifier, respectively, was not specified.

C0027:

The specification states:

```
Interface Entity
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

The references for the **Entity** interface are completely expanded when the XML content is parsed.

C0028:

The specification states:

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

MSXML3 and MSXML6

The parent of an **entity** node is the **documentType** node to which it belongs.

V0005:

The specification states:

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

MSXML3 and MSXML6

The type of **publicId** and **systemId** attributes in the **Entity** interface is `VARIANT`.

C0029:

The specification states:

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

MSXML3 and MSXML6

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

C0030:

The specification states:

```
publicId of type DOMString, readonly
The public identifier associated with the entity, if specified. If the public identifier was
not specified, this is null.
```

MSXML3 and MSXML6

The **publicId** attribute of the **Notation** interface returns an empty string if the public identifier was not specified.

C0031:

The specification states:

```
systemId of type DOMString, readonly
The system identifier associated with the entity, if specified. If the system identifier was
not specified, this is null.
```

MSXML3 and MSXML6

The **systemId** attribute of the **Entity** interface returns an empty string if the system identifier was not specified.

C0032:

The specification states:

```
data of type DOMString
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 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.

2.3 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