[MS-XML]:
Microsoft Extensible Markup Language (XML) 1.0 Fourth Edition Standards Support Document

---

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **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 may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events 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 specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do 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 are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
03/17/2010	0.1	New	Released new document.
03/26/2010	1.0	None	Introduced no new technical or language changes.
05/26/2010	1.2	None	Introduced no new technical or language changes.
09/08/2010	1.3	Major	Significantly changed the technical content.
10/13/2010	1.4	Minor	Clarified the meaning of the technical content.
02/10/2011	2.0	No change	Introduced no new technical or language changes.
02/22/2012	3.0	Major	Significantly changed the technical content.
07/25/2012	3.1	Minor	Clarified the meaning 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 *Extensible Markup Language (XML) 1.0 (Fourth Edition)* [XML], W3C Recommendation 16 August 16 2006, edited in place 29 September 2006.

The [XML] 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 described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

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. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[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

[RFC3066] Alvestrand, H., "Tags for the Identification of Language", RFC 3066, January 2001, http://www.ietf.org/rfc/rfc3066.txt

[XML] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", W3C Recommendation, August 2006, http://www.w3.org/TR/2006/REC-xml-20060816/

1.2.2 Informative References

None.

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 and Windows® Internet Explorer® 10.

1.4 Standards Support Requirements

To conform to [XML], 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 [XML] 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####	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 a full list of clarifications in the Microsoft implementation of [XML].

- Section 2.1 includes only those variations that violate a MUST requirement in the target specification.
- Section 2.2 describes further variations from MAY and SHOULD requirements.
- Section 2.3 identifies variations in error handling.
- Section 2.4 identifies variations that impact security.

2.1 Normative Variations

The following subsections detail the normative variations from MUST requirements in [XML].

2.1.1 [XML] Section 2.3, Common Syntactic Constructs

V0001:

The specification states:

```
[Definition: A Name is a token beginning with a letter or one of a few punctuation
characters, and continuing with letters, digits, hyphens, underscores, colons, or
full stops, together known as name characters.] Names beginning with the string
"xml", or with any string which would match (('X'|'x') ('M'|'m') ('L'|'l')), are
reserved for standardization in this or future versions of this specification.
```

MSXML3 and MSXML6

Any name that begins with a string that matches (('X'|'x') ('M'|'m') ('L'|'l')) is valid and not defined as reserved.

2.1.2 [XML] Section 2.8, Prolog and Document Type Declaration

V0002:

The specification states:

```
Well-formedness constraint: PEs in Internal Subset
In the internal DTD subset, parameter-entity references MUST NOT occur within
markup declarations; they may occur where markup declarations can occur. (This does
not apply to references that occur in external parameter entities or to the
external subset.)
```

MSXML3 and MSXML6

In an internal DTD subset, parameter-entity references can occur in processing instructions, notation declarations, and comments but not in entity declarations, element-type declarations, or attribute-list declarations. Parameter-entity references are not recognized in comment declarations.

2.1.3 [XML] Section 2.11, End-of-Line Handling

V0012:

The specification states:

```
To simplify the tasks of applications, the XML processor MUST behave as if it normalized all
line breaks in external parsed entities (including the document entity) on input, before
parsing, by translating both the two-character sequence #xD #xA and any #xD that is not
followed by #xA to a single #xA character.
```

MSXML3 and MSXML6

The "
" character reference sequence and the "#xD #xA" character sequence are normalized in external parsed entities.

2.1.4 [XML] Section 3, Logical Structures

V0003:

The specification states:

```
Validity constraint: Element Valid
An element is valid if there is a declaration matching elementdecl where the Name
matches the element type, and one of the following holds:
1. The declaration matches EMPTY and the element has no content (not even entity
references, comments, PIs or white space).
2. The declaration matches children and the sequence of child elements belongs to
the language generated by the regular expression in the content model, with
optional white space, comments and PIs (i.e. markup matching production [27] Misc)
between the start-tag and the first child element, between child elements, or
between the last child element and the end-tag. Note that a CDATA section
containing only white space or a reference to an entity whose replacement text is
character references expanding to white space do not match the nonterminal S, and
hence cannot appear in these positions; however, a reference to an internal entity
with a literal value consisting of character references expanding to white space
does match S, since its replacement text is the white space resulting from
expansion of the character references.
```

MSXML3 and MSXML6

A **CDATA** section is valid if it contains white space only or if it contains a reference to an entity whose replacement text is character references that expand to white space.

MSXML3

An element with a declaration that matches **EMPTY** is valid if it contains comments, processing instructions, and entity references. All comments, processing instructions, and entity references are resolved to empty strings.

MSMXL6

An element with a declaration that matches **EMPTY** is valid if it contains entity references. Entity references are resolved to empty strings.

2.1.5 [XML] Section 3.1, Start-Tags, End-Tags, and Empty-Element Tags

V0004:

The specification states:

```
Well-formedness constraint: Unique Att Spec
An attribute name MUST NOT appear more than once in the same start-tag or empty-
element tag.
```

MSXML3 and MSXML6

An attribute can appear more than once in the same start-tag or empty-element tag. However, the first declaration takes precedence and the subsequent declarations are ignored.

2.1.6 [XML] Section 3.3.1, Attribute Types

V0005:

The specification states:

```
Validity constraint: No Duplicate Tokens
The notation names in a single NotationType attribute declaration, as well as the
NmTokens in a single Enumeration attribute declaration, MUST all be distinct.
```

MSXML3

Duplicate notation names in a single **NotationType** attribute declaration are allowed.

2.1.7 [XML] Section 3.3.2, Attribute Defaults

V0006:

The specification states:

```
Validity constraint: Attribute Default Value Syntactically Correct
The declared default value MUST meet the syntactic constraints of the declared
attribute type. That is, the default value of an attribute:
 - of type IDREF or ENTITY must match the Name production;
 - of type IDREFS or ENTITIES must match the Names production;
 - of type NMTOKEN must match the Nmtoken production;
 - of type NMTOKENS must match the Nmtokens production;
 - of an enumerated type (either a NOTATION type or an enumeration) must match one
of the enumerated values.
Note that only the syntactic constraints of the type are required here; other
constraints (e.g. that the value be the name of a declared unparsed entity, for an
attribute of type ENTITY) will be reported by a validating parser only if an
element without a specification for this attribute actually occurs.
```

MSXML3 and MSXML6

When an attribute definition has a declared default value that has a reference to an undefined entity, but is never used in a document, the expected result is to have no errors; the actual result is a "Reference to undefined entity 'X'" error, where X is the declared default value.

2.1.8 [XML] Section B, Character Classes

V0007:

The specification states:

```
Name start characters must have one of the categories Ll, Lu, Lo, Lt, Nl.
```

MSXML3 and MSXML6

Name start characters in the category Lm (Letter, Modifier) are valid.

V0008:

The specification states:

```
Characters which have a font or compatibility decomposition (i.e. those with a
"compatibility formatting tag" in field 5 of the database -- marked by field 5
beginning with a "<") are not allowed.
```

MSXML3 and MSXML6

The character `#x005F` (the underscore or low line character) is valid.

V0009:

The specification states:

```
Characters ':' and '_' are allowed as name-start characters.
```

MSXML3 and MSXML6

The ":" is not a valid name-start character.

2.2 Clarifications

The following subsections identify clarifications to recommendations made by [XML].

2.2.1 [XML] Section 2.2, Characters

V0010:

The specification states:

```
[2]   Char   ::=   #x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]
/* any Unicode character, excluding the surrogate blocks, FFFE, and FFFF. */
```

MSXML3 and MSXML6

The processor accepts the surrogate block `#xFFFE` as a valid character reference.

2.2.2 [XML] Section 2.3, Common Syntactic Constructs

C0001:

The specification states:

```
Note:
The Namespaces in XML Recommendation [XML Names] assigns a meaning to names
containing colon characters. Therefore, authors should not use the colon in XML
names except for namespace purposes, but XML processors must accept the colon as a
name character.
```

MSXML3 and MSXML6

XML names that begin with a colon (for example, :doc) are not valid.

Specifying a name that begins with a colon results in an error with an error code of `0xC00CE504` and an error reason of "A name was started with an invalid character."

2.2.3 [XML] Section 2.5, Comments

C0002:

The specification states:

```
Definition: Comments may appear anywhere in a document outside other markup; in
addition, they may appear within the document type declaration at places allowed by
the grammar. They are not part of the document's character data; an XML processor
MAY, but need not, make it possible for an application to retrieve the text of
comments. For compatibility, the string "--" (double-hyphen) MUST NOT occur within
comments.] Parameter entity references MUST NOT be recognized within comments.
```

MSXML3 and MSXML6

The following clarifications apply:

- Comment text can be retrieved.
- A valid comment can contain a double hyphen ("--") when the double hyphen is represented by the character reference "`&#45;&#45;`".

2.2.4 [XML] Section 2.10, White Space Handling

C0003:

The specification states:

```
An XML processor MUST always pass all characters in a document that are not markup
through to the application. A validating XML processor MUST also inform the application which
of these characters constitute white space appearing in element content.
```

MSXML3 and MSXML6

The following clarifications apply:

- The **preserveWhiteSpace** attribute is used to specify the default white space handling behavior.
- When **preserveWhiteSpace** is set to `true`, all white space is preserved, regardless of any `xml:space` attributes specified in the document type definition (DTD). It is equivalent to having an `xml:space="preserve"` attribute on every element. When **preserveWhiteSpace** is set to `false`, the values of any `xml:space` attributes determine where white space is preserved.

C0004:

The specification states:

```
The value "default" signals that applications' default white-space processing modes
are acceptable for this element; the value "preserve" indicates the intent that
applications preserve all the white space. This declared intent is considered to
apply to all elements within the content of the element where it is specified,
unless overridden with another instance of the xml:space attribute. This
specification does not give meaning to any value of xml:space other than "default"
and "preserve". It is an error for other values to be specified; the XML processor
MAY report the error or MAY recover by ignoring the attribute specification or by
reporting the (erroneous) value to the application. Applications may ignore or
reject erroneous values.
```

MSXML3 and MSXML6

An error is reported for values other than `default` or `preserve`.

2.2.5 [XML] Section 2.12, Language Identification

C0005:

The specification states:

```
In document processing, it is often useful to identify the natural or formal
language in which the content is written. A special attribute named xml:lang may be
inserted in documents to specify the language used in the contents and attribute
values of any element in an XML document. In valid documents, this attribute, like
any other, MUST be declared if it is used. The values of the attribute are language
identifiers as defined by [RFC3066], Tags for the Identification of Languages, or
its successor; in addition, the empty string may be specified.
```

MSXML3 and MSXML6

No special format checking is performed on the `xml:lang` attribute value. .

2.2.6 [XML] Section 3, Logical Structures

C0006:

The specification states:

```
This specification does not constrain the application semantics, use, or (beyond
syntax) names of the element types and attributes, except that names beginning with
a match to (('X'|'x')('M'|'m')('L'|'l')) are reserved for standardization in this
or future versions of this specification.
```

MSXML3 and MSXML6

Any name that begins with a string that matches (('X'|'x')('M'|'m')('L'|'l')) is valid and is not defined as reserved.

2.2.7 [XML] Section 3.1, Start-Tags, End-Tags, and Empty-Element Tags

C0007:

The specification states:

```
Well-formedness constraint: No < in Attribute Values
The replacement text of any entity referred to directly or indirectly in an
attribute value MUST NOT contain a <.
```

MSXML3 and MSXML6

Attribute values can contain the less-than sign ("<") if it is represented by either the `&#60;` character reference or the `<` character entity reference.

C0008:

The specification states:

```
Definition: An element with no content is said to be empty.] The representation of
an empty element is either a start-tag immediately followed by an end-tag, or an
empty-element tag. [Definition: An empty-element tag takes a special form:]
Tags for Empty Elements
[44]   EmptyElemTag   ::=   '<' Name (S Attribute)* S? '/>'[WFC: Unique Att Spec]
Empty-element tags may be used for any element which has no content, whether or not
it is declared using the keyword EMPTY. For interoperability, the empty-element tag
SHOULD be used, and SHOULD only be used, for elements which are declared EMPTY.
```

MSXML3 and MSXML6

Empty element tags are acceptable for any element that has no content, whether or not it is declared using the keyword **EMPTY**.

2.2.8 [XML] Section 3.2, Element Type Declarations

C0009:

The specification states:

```
The element structure of an XML document may, for validation purposes, be
constrained using element type and attribute-list declarations. An element type
declaration constrains the element's content. Element type declarations often
constrain which element types can appear as children of the element. At user
option, an XML processor MAY issue a warning when a declaration mentions an element
type for which no declaration is provided, but this is not an error.
```

MSXML3 and MSXML6

The following clarifications apply:

- The element structure is constrained when defined in the DTD or when the **validateOnParse** property is set to `true`; otherwise, validation does not occur.
- An error is raised when an element type is used in the XML document but not defined. No error is raised if the element type is not used.

2.2.9 [XML] Section 3.2.1, Element Content

C0010:

The specification states:

```
Validity constraint: Proper Group/PE Nesting
Parameter-entity replacement text MUST be properly nested with parenthesized
groups. That is to say, if either of the opening or closing parentheses in a
choice, seq, or Mixed construct is contained in the replacement text for a
parameter entity, both MUST be contained in the same replacement text.
For interoperability, if a parameter-entity reference appears in a choice, seq, or
Mixed construct, its replacement text SHOULD contain at least one non-blank
character, and neither the first nor last non-blank character of the replacement
text SHOULD be a connector (| or ,).
```

MSXML3 and MSXML6

The following clarifications apply:

- Parameter-entity replacement text can be blank if it is valid as defined in the DTD.
- Parameter-entity replacement text can have the characters '|' or ',' at the beginning or end of the string if it is valid as defined in the DTD.

2.2.10 [XML] Section 3.3, Attribute-List Declarations

C0011:

The specification states:

```
Definition: Attribute-list declarations specify the name, data type, and default
value (if any) of each attribute associated with a given element type:]
Attribute-list Declaration
[52]   AttlistDecl   ::=   '<!ATTLIST' S Name AttDef* S? '>'
[53]   AttDef   ::=   S Name S AttType S DefaultDecl

The Name in the AttlistDecl rule is the type of an element. At user option, an XML
processor MAY issue a warning if attributes are declared for an element type not
itself declared, but this is not an error. The Name in the AttDef rule is the name
of the attribute.
```

MSXML3 and MSXML6

No warning or error messages are generated if attributes are declared for undeclared element types.

C0012:

The specification states:

```
For interoperability, an XML processor MAY at user option issue a warning when more
than one attribute-list declaration is provided for a given element type, or more
than one attribute definition is provided for a given attribute, but this is not an error.
```

MSXML3 and MSXML6

No warning or error messages are generated if more than one attribute-list declaration is provided for a specific element type or more than one attribute definition is provided for a specific attribute.

2.2.11 [XML] Section 3.3.1, Attribute Types

C0013:

The specification states:

```
Validity constraint: Entity Name
Values of type ENTITY MUST match the Name production, values of type ENTITIES MUST
match Names; each Name MUST match the name of an unparsed entity declared in the DTD.
```

MSXML3 and MSXML6

Character references are not allowed in **ENTITY** names.

C0014:

The specification states:

```
Validity constraint: Enumeration
Values of this type MUST match one of the Nmtoken tokens in the declaration.
For interoperability, the same Nmtoken SHOULD NOT occur more than once in the
enumerated attribute types of a single element type.
```

MSXML3 and MSXML6

Duplicate **Nmtoken** tokens can occur in the enumerated attribute types of a single element type.

2.2.12 [XML] Section 3.3.3, Attribute-Value Normalization

C0015:

The specification states:

```
Before the value of an attribute is passed to the application or checked for
validity, the XML processor MUST normalize the attribute value by applying the
algorithm below, or by using some other method such that the value passed to the
application is the same as that produced by the algorithm.

1.  All line breaks MUST have been normalized on input to #xA as described in 2.11 End-of-
Line Handling, so the rest of this algorithm operates on text normalized in this way.
2.  Begin with a normalized value consisting of the empty string.
3.  For each character, entity reference, or character reference in the unnormalized
attribute value, beginning with the first and continuing to the last, do the following:

     For a character reference, append the referenced character to the normalized value.
```

```
     For an entity reference, recursively apply step 3 of this algorithm to the
replacement text of the entity.
     For a white space character (#x20, #xD, #xA, #x9), append a space character (#x20) to
the normalized value.
     For another character, append the character to the normalized value.

If the attribute type is not CDATA, then the XML processor MUST further process the
normalized attribute value by discarding any leading and trailing space (#x20)
characters, and by replacing sequences of space (#x20) characters by a single space
(#x20) character.
```

MSXML3

The following clarifications apply:

- During attribute value normalization, white-space characters (`#x20`, `#xD`, `#xA`, `#x9`) are not replaced with space characters (`#x20`).
- When the attribute type is **NMTOKENS**, the attribute value is not normalized by removing leading and trailing space (`#x20`) characters or by replacing sequences of space characters with one space character.

MSXML6

When using the old parser (that is, the **NewParser** property has not been enabled), the following clarifications apply:

- During attribute value normalization, white-space characters (`#x20`, `#xD`, `#xA`, `#x9`) are not replaced with space characters (`#x20`).
- When the attribute type is **NMTOKENS**, the attribute value is not normalized by removing leading and trailing space (`#x20`) characters or by replacing sequences of space characters with one space character.

C0016:

The specification states:

```
All attributes for which no declaration has been read SHOULD be treated by a non-
validating processor as if declared CDATA.
```

MSXML3 and MSXML6

MSXML3 and MSXML6 are validating parsers.

2.2.13 [XML] Section 4.1, Character and Entity References

C0017:

The specification states:

```
Well-formedness constraint: Entity Declared
In a document without any DTD, a document with only an internal DTD subset which
contains no parameter entity references, or a document with "standalone='yes'", for
an entity reference that does not occur within the external subset or a parameter
```

```
entity, the Name given in the entity reference MUST match that in an entity
declaration that does not occur within the external subset or a parameter entity,
except that well-formed documents need not declare any of the following entities:
amp, lt, gt, apos, quot. The declaration of a general entity MUST precede any
reference to it which appears in a default value in an attribute-list declaration.
Note that non-validating processors are not obligated to to read and process entity
declarations occurring in parameter entities or in the external subset; for such
documents, the rule that an entity must be declared is a well-formedness constraint
only if standalone='yes'
```

MSXML3 and MSXML6

MSXML3 and MSXML6 are validating parsers.

C0018:

The specification states:

```
Validity constraint: Entity Declared

In a document with an external subset or parameter entity references with
"standalone='no'", the Name given in the entity reference MUST match that in an
entity declaration. For interoperability, valid documents SHOULD declare the
entities amp, lt, gt, apos, quot, in the form specified in 4.6 Predefined Entities.
The declaration of a parameter entity MUST precede any reference to it. Similarly,
the declaration of a general entity MUST precede any attribute-list declaration
containing a default value with a direct or indirect reference to that general entity.
```

MSXML3 and MSXML6

A general entity can be declared after an attribute-list declaration if the attribute-list declaration contains a default value with an indirect reference to the general entity.

2.2.14 [XML] Section 4.2.2, External Entities

C0019:

The specification states:

```
It is an error for a fragment identifier (beginning with a # character) to be part
of a system identifier. Unless otherwise provided by information outside the scope
of this specification (e.g. a special XML element type defined by a particular DTD,
or a processing instruction defined by a particular application specification),
relative URIs are relative to the location of the resource within which the entity
declaration occurs. This is defined to be the external entity containing the '<'
which starts the declaration, at the point when it is parsed as a declaration. A
URI might thus be relative to the document entity, to the entity containing the
external DTD subset, or to some other external parameter entity. Attempts to
retrieve the resource identified by a URI may be redirected at the parser level
(for example, in an entity resolver) or below (at the protocol level, for example,
via an HTTP Location: header). In the absence of additional information outside the
scope of this specification within the resource, the base URI of a resource is
always the URI of the actual resource returned. In other words, it is the URI of
the resource retrieved after all redirection has occurred.
```

MSXML3

A fragment identifier that begins with a number sign (#) is allowed in a system identifier to refer to external entity.

MSXM3 and MSXML6

When the system identifier contains a relative URI, its base URI is the original URI of the referrer document before redirection.

C0020:

The specification states:

```
System identifiers (and other XML strings meant to be used as URI references) may
contain characters that, according to [IETF RFC 3986], must be escaped before a URI
can be used to retrieve the referenced resource. The characters to be escaped are
the control characters #x0 to #x1F and #x7F (most of which cannot appear in XML),
space #x20, the delimiters '<' #x3C, '>' #x3E and '"' #x22, the unwise characters
'{' #x7B, '}' #x7D, '|' #x7C, '\' #x5C, '^' #x5E and '`' #x60, as well as all
characters above #x7F. Since escaping is not always a fully reversible process, it
MUST be performed only when absolutely necessary and as late as possible in a
processing chain.
```

MSXML3 and MSXML6

The characters '{', '}', '|', '\', '^', or '`' (and their equivalent character entities `#x7B`, `#x7D`, `#x7C`, `#x5C`, `#x5E`, and `#x60` respectively) are not escaped in a URI reference. These unwise characters are treated as string and are passed along as is.

2.2.15 [XML] Section 4.3.2, Well-Formed Parsed Entities

C0021:

The specification states:

```
The document entity is well-formed if it matches the production labeled document.
An external general parsed entity is well-formed if it matches the production
labeled extParsedEnt. All external parameter entities are well-formed by definition.
Note:
Only parsed entities that are referenced directly or indirectly within the document
are required to be well-formed.
```

MSXML3 and MSXML6

Parsed entities that are not referenced directly or indirectly within the document are also required to be well-formed.

2.2.16 [XML] Section 4.4, XML Processor Treatment of Entities and References

C0022:

The specification states:

```
Reference in Content:
```

```
Entity Type Parameter:               Not recognized
Entity Type Internal General:        Included
Entity Type External Parsed General: Included if validating
Entity Type Unparsed:                Forbidden
Character:                           Included
```

MSXML3 and MSXML6

Character entity references in content are processed as literal text.

2.2.17 [XML] Section 4.7, Notation Declarations

C0023:

The specification states:

```
XML processors MUST provide applications with the name and external identifier(s)
of any notation declared and referred to in an attribute value, attribute
definition, or entity declaration. They MAY additionally resolve the external
identifier into the system identifier, file name, or other information needed to
allow the application to call a processor for data in the notation described. (It
is not an error, however, for XML documents to declare and refer to notations for
which notation-specific applications are not available on the system where the XML
processor or application is running.)
```

MSXML3 and MSXML6

An external identifier is not resolved into the system identifier, file name, or other information needed to allow the application to call a processor for data in the notation described.

2.2.18 [XML] Section 4.8, Document Entity

C0024:

The specification states:

```
This specification does not specify how the document entity is to be located by an
XML processor; unlike other entities, the document entity has no name and might
well appear on a processor input stream without any identification at all.
```

MSXML3 and MSXML6

The document entity name is resolved as "#document".

2.2.19 [XML] Section 5.1, Validating and Non-Validating Processors

C0025:

The specification states:

```
Conforming XML processors fall into two classes: validating and non-
validating.Validating and non-validating processors alike MUST report violations of
this specification's well-formedness constraints in the content of the document
```

entity and any other parsed entities that they read.

MSXML3 and MSXML6

MSXML3 and MSXML6 are validating parsers, and do report violations of well-formedness constraints.

C0026:

The specification states:

Non-validating processors are REQUIRED to check only the document entity, including the entire internal DTD subset, for well-formedness. [Definition: While they are not required to check the document for validity, they are REQUIRED to process all the declarations they read in the internal DTD subset and in any parameter entity that they read, up to the first reference to a parameter entity that they do not read; that is to say, they MUST use the information in those declarations to normalize attribute values, include the replacement text of internal entities, and supply default attribute values.] Except when standalone="yes", they MUST NOT process entity declarations or attribute-list declarations encountered after a reference to a parameter entity that is not read, since the entity may have contained overriding declarations; when standalone="yes", processors MUST process these declarations.

MSXML3 and MSXML6

MSXML3 and MSXML6 are validating parsers.

C0027:

The specification states:

Note that when processing invalid documents with a non-validating processor the application may not be presented with consistent information. For example, several requirements for uniqueness within the document may not be met, including more than one element with the same id, duplicate declarations of elements or notations with the same name, etc. In these cases the behavior of the parser with respect to reporting such information to the application is undefined.

MSXML3 and MSXML6

MSXML3 and MSXML6 are validating parsers.

2.2.20 [XML] Section 5.2, Using XML Processors

C0028:

The specification states:

The behavior of a validating XML processor is highly predictable; it must read every piece of a document and report all well-formedness and validity violations. Less is required of a non-validating processor; it need not read any part of the document other than the document entity. This has two effects that may be important to users of XML processors:

MSXML3 and MSXML6

MSXML3 and MSXML6 are validating parsers and do report all well-formedness and validity violations.

C0029:

The specification states:

```
Certain well-formedness errors, specifically those that require reading external entities,
may fail to be detected by a non-validating processor. Examples include the constraints
entitled Entity Declared, Parsed Entity, and No Recursion, as well as some of the cases
described as forbidden in 4.4 XML Processor Treatment of 3 and MXEntities and References.
```

MSXML3 and MSXML6

MSXML3 and MSXML6 are validating parsers.

C0030:

The specification states:

```
The information passed from the processor to the application may vary, depending on
whether the processor reads parameter and external entities. For example, a non-
validating processor may fail to normalize attribute values, include the
replacement text of internal entities, or supply default attribute values, where
doing so depends on having read declarations in external or parameter entities.
```

MSXML3 and MSXML6

MSXML3 and MSXML6 are validating parsers.

2.3 Error Handling

There are no additional considerations for error handling.

2.4 Security

There are no additional security considerations.

3 Change Tracking

This section identifies changes that were made to the [MS-XML] protocol document between the February 2012 and July 2012 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1 Introduction	Updated document to remove beta tagging.	N	Content updated.

4 Index