[MS-XMLSS]: Microsoft XML Schema (Part 1: Structures) 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
09/08/2010	0.1	New	Released new document.
10/13/2010	0.2	Minor	Clarified the meaning of the technical content.
02/10/2011	1.0	No change	Introduced no new technical or language changes.
02/15/2012	2.0	Major	Significantly changed the technical content.
07/25/2012	2.1	Minor	Clarified the meaning of the technical content.

Table of Contents

1 Introduction

This document describes the level of support provided by Microsoft XML Core Services (MSXML) 6.0 for *XML Schema Part 1: Structures Second Edition* [W3C-XSS], published on 28 October, 2004.

MSXML6 supports *XML Schema Part 1: Structures Second Edition* **[W3C-XSS]** using the *Extensible Markup Language (XML) 1.0 (Fourth Edition)* **[XML]**, W3C Recommendation 16 August 2006, edited in place 29 September 2006.

The [W3C-XSS] 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

[W3C-XSS] W3C, "XML Schema Part 1: Structures Second Edition", W3C Recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

1.2.2 Informative References

None.

1.3 Microsoft Implementations

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

MSXML6 implements the [W3C-XSS] specification.

1.4 Standards Support Requirements

To conform to [W3C-XSS], 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].)

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 variations, clarifications, and extension points in the Microsoft implementation of [W3C-XSS].

- 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 [W3C-XSS].

2.1.1 [W3C-XSS] Section 3.3.1, The Element Declaration Schema Component

V0001:

The specification states:

```
{value constraint} establishes a default or fixed value for an element. If
default is specified, and if the element being ·validated· is empty, then the
canonical form of the supplied constraint value becomes the [schema normalized
value] of the ·validated· element in the ·post-schema-validation infoset·. If
fixed is specified, then the element's content must either be empty, in which
case fixed behaves as default, or its value must match the supplied constraint
value.
```

MSXML6

If an element that is being validated includes a **{value constraint}** value that is specified as `default` and the element is empty, the default value is not supplied.

2.1.2 [W3C-XSS] Section 3.3.4, Element Declaration Validation Rules

V0002:

The specification states:

```
4 If there is an attribute information item among the element information item's
[attributes] whose [namespace name] is identical to
http://www.w3.org/2001/XMLSchema-instance and whose [local name] is type ...

4.2 The ·local name· and ·namespace name· (as defined in QName Interpretation
(§3.15.3)), of the ·actual value· of that attribute information item must resolve
to a type definition, as defined in QName resolution (Instance) (§3.15.4) –
[Definition:]  call this type definition the local type definition;
```

MSXML6

An attribute information item that does not resolve to a valid type does not cause an error.

2.1.3 [W3C-XSS] Section 3.4.6, Constraints on Complex Type Definition Schema Components

V0003:

The specification states:

```
Schema Component Constraint: Derivation Valid (Extension)
If the {derivation method} is extension, the appropriate case among the following
must be true:

1 If the {base type definition} is a complex type definition, then all of the
following must be true:

     1.1 The {final} of the {base type definition} must not contain extension.
     1.2 Its {attribute uses} must be a subset of the {attribute uses} of the complex type
definition itself, that is, for every attribute use in the {attribute uses} of the {base type
definition}, there must be an attribute use in the {attribute uses} of the complex type
definition itself whose {attribute declaration} has the same {name}, {target namespace} and
{type definition} as its attribute declaration.
     1.3 If it has an {attribute wildcard}, the complex type definition must also
have one, and the base type definition's {attribute wildcard}'s {namespace
constraint} must be a subset of the complex type definition's {attribute
wildcard}'s {namespace constraint}, as defined by Wildcard Subset (§3.10.6).
```

MSXML6

The **{namespace constraint}** property of the base type definition's **{attribute wildcard}** property is not required to be a subset of the **{namespace constraint}** of the complex type definition's **{attribute wildcard}** property.

The **{attribute uses}** property of the base type definition is not required to be a subset of the **{attribute uses}** property of the complex type definition.

V0004:

The specification states:

```
2 If the {base type definition} is a simple type definition, then all of the following must
be true:

     2.1 The {content type} must be the same simple type definition.
     2.2 The {final} of the {base type definition} must not contain extension.
```

MSXML6

A **{final}** property of a simple type definition may have a value of `extension`.

2.1.4 [W3C-XSS] Section 3.10.6, Constraints on Wildcard Schema Components

V0005:

The specification states:

```
For a wildcard's {namespace constraint} value to be the intensional union of two other such
values (call them O1 and O2): the appropriate case among the following must be true:

     1 If O1 and O2 are the same value, then that value must be the value.
     2 If either O1 or O2 is any, then any must be the value.
     3 If both O1 and O2 are sets of (namespace names or ·absent·), then the union of those
sets must be the value.
     4 If the two are negations of different values (namespace names or ·absent·), then a
pair of not and ·absent· must be the value.
```

MSXML6

When a wildcard's **{namespace constraint}** value is the union of two values (`O1` and `O2`), if either `O1` or `O2` is `any`, the value is not always `any`.

2.1.5 [W3C-XSS] Section 3.11.4, Identity-constraint Definition Validation Rules

V0006:

The specification states:

```
4 [Definition:]  Call the subset of the ·target node set· for which all the
{fields} evaluate to a node-set with exactly one member which is an element or
attribute node with a simple type the qualified node set...

4.2 If the {identity-constraint category} is key, then all of the following
must be true: ...

4.2.3 No element member of the ·key-sequence· of any member of the ·qualified
node set· was assessed as ·valid· by reference to an element declaration
whose {nillable} is true.
```

MSXML6

If an element of the key sequence of a qualified node set has a **{nillable}** attribute of value `true`, it can be valid.

2.1.6 [W3C-XSS] Section 3.14.4, Simple Type Definition Validation Rules

V0007:

The specification states:

```
Validation Rule: String Valid

For a string to be locally ·valid· with respect to a simple type definition
all of the following must be true:

1 It is schema-valid with respect to that definition as defined by Datatype
Valid in [XML Schemas: Datatypes].

2 The appropriate case among the following must be true:
  2.1 If The definition is ENTITY or is validly derived from ENTITY given
  the empty set, as defined in Type Derivation OK (Simple) (§3.14.6), then
  the string must be a ·declared entity name·.
```

```
2.2 If The definition is ENTITIES or is validly derived from ENTITIES given
the empty set, as defined in Type Derivation OK (Simple) (§3.14.6), then
every whitespace-delimited substring of the string must be a ·declared
entity name·.

2.3 otherwise no further condition applies.
```

MSXML6

Schema validation does not use the **ENTITY** attribute, so all statements that refer to **ENTITY** or the **ENTITIES** attribute are ignored.

2.1.7 [W3C-XSS] Section 3.14.6, Constraints on Simple Type Definition Schema Components

V0008:

The specification states:

```
3 If the {variety} is union, then all of the following must be true:
3.1 The {member type definitions} must all have {variety} of atomic or list…
```

MSXML6

A simple type with a **{variety}** value of `union` can include a simple type with a **{variety}** value of `union`.

2.1.8 [W3C-XSS] Section 4.2.3, References to schema components across namespaces

V0009:

The specification states:

```
Note: Since both the namespace and schemaLocation [attribute] are optional,
a bare <import/> information item is allowed. This simply allows unqualified
reference to foreign components with no target namespace without giving
any hints as to where to find them.
```

MSXML6

An unqualified reference results in an invalid schema document.

2.2 Clarifications

The following subsections identify clarifications to [W3C-XSS].

2.2.1 [W3C-XSS] Section 2.1, Overview of XML Schema

V0010:

The specification states:

```
[Definition:]  We refer to the augmented infoset which results from conformant processing as
defined in this specification as the post-schema-validation infoset, or PSVI.
```

MSXML6

No APIs are exposed for the post-schema-validation infoset (PSVI) so it cannot be tested. The internal implementation of PSVI may or may not conform to the specification. All features that depend on the PSVI should be tested separately.

2.2.2 [W3C-XSS] Section 3.8.4, Model Group Validation Rules

V0011:

The specification states:

```
Nothing in the above should be understood as ruling out groups whose
{particles} is empty: although no sequence can be ·valid· with respect to
such a group whose {compositor} is choice, the empty sequence is ·valid· with
respect to empty groups whose {compositor} is sequence or all.
```

MSXML6

An empty group that has a **{compositor}** property with a value of `choice` can be valid.

2.2.3 [W3C-XSS] Section 3.8.6, Constraints on Model Group Schema Components

V0012:

The specification states:

```
Schema Component Constraint: Element Declarations Consistent

If the {particles} contains, either directly, indirectly (that is, within the
{particles} of a contained model group, recursively) or ·implicitly· two or
more element declaration particles with the same {name} and {target namespace},
then all their type definitions must be the same top-level definition, that
is, all of the following must be true:
1 all their {type definition}s must have a non-·absent· {name}.
2 all their {type definition}s must have the same {name}.
3 all their {type definition}s must have the same {target namespace}.
```

MSXML6

If a model group implicitly contains an element declaration that has the same **{name}** and **{target namespace}** properties with a directly contained element declaration, it does not cause an error.

2.2.4 [W3C-XSS] Section 4.1, Layer 1: Summary of the Schema-validity Assessment Core

C0001:

The specification states:

```
Processors have the option to assemble (and perhaps to optimize or pre-compile) the entire
schema prior to the start of an ·assessment· episode, or to gather the schema lazily as
individual components are required. In all cases it is required that:

- The processor succeed in locating the ·schema components· transitively required to complete
an ·assessment· (note that components derived from ·schema documents· can be integrated with
components obtained through other means);
no definition or declaration changes once it has been established;
if the processor chooses to acquire declarations and definitions dynamically, that there be
no side effects of such dynamic acquisition that would cause the results of ·assessment· to
differ from that which would have been obtained from the same schema components acquired in
bulk.
```

MSXML6

The **SchemaCache** object does not acquire declarations and definitions dynamically.

C0002:

The specification states:

```
Note:  the ·assessment· core is defined in terms of schema components at the abstract level,
and no mention is made of the schema definition syntax (i.e. <schema>). Although many
processors will acquire schemas in this format, others may operate on compiled
representations, on a programmatic representation as exposed in some programming language,
etc.
```

MSXML6

The **SchemaCache** object does not operate on compiled representations or on a programmatic representation. **SchemaCache** acquires schemas only in **schema** document format.

2.3 Error Handling

There are no additional error handling considerations.

2.4 Security

There are no additional security considerations.

3 Change Tracking

This section identifies changes that were made to the [MS-XMLSS] 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