ISO/IEC 29500-3:201x

Office Open XML File Formats — Markup Compatibility and Extensibility

[Working DRAFT WD1 0.9]

2013-06-20

Comment [JH1]: General TO DO:
- Review all example text for proper use of styles when referring to elements, attributes, etc.
- Address mismatches and error handling

Broad changes:
- All non-breaking spaces replaced with spaces
- Manual line breaks replaced with regular line breaks in example markup

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 29500-3 was prepared by ISO/IEC JTC 1, Information technology, Subcommittee SC 34, Document description and processing languages.

This fourth edition cancels and replaces the third edition (ISO/IEC 29500-3:2012).

The major changes from the previous edition include:

1. Addition of a new attribute for declaring application-defined extension elements
2. Specification of the core semantics in one place
3. Removal of namespace subsumption
4. Expansion of examples, in particular, by providing output documents

The intended semantics remains the same as long as namespace subsumption is not used and the new attribute for declaring application-defined extension elements is not used.

Rationale, which will be removed from the final draft:

Why revision? The biggest reason is that interactions between MCE constructs were not clear enough. In particular, although application-defined extension elements suppress normal processing of every MCE construct, application-defined extension elements were not even mentioned in Clause 10, which defines the semantics of the MCE constructs. Other reasons include:

- Namespace subsumption was underspecified, unimplemented, and unused.

- There were no mechanisms for declaring application-defined extension elements.

Existing users of MCE are not affected as long as they do not use namespace subsumption and do not use (and even prohibit) the new attribute. Since OOXML Parts 1 and 4 do not use namespace subsumption, they should not be affected if they prohibit the use of the new attribute. (Note: CORs for 1 and 4 are expected for introducing this prohibition.)

Major changes in the third edition included:

- Removed all traces of the concept of *markup editor*
- Removed the attributes PreserveAttributes and PreserveElements

There were no major changes in the second edition.

ISO/IEC 29500 consists of the following parts, under the general title *Information technology — Document description and processing languages — Office Open XML File Formats*:

- *Part 1: Fundamentals and Markup Language Reference*
- *Part 2: Open Packaging Conventions*
- *Part 3: Markup Compatibility and Extensibility*
- *Part 4: Transitional Migration Features*

Annex A is for information only.

Introduction

Comment [rcj2]: Add fwd ptr to Annex A primer w.r.t getting started

ISO/IEC 29500 specifies a family of XML schemas, collectively called *Office Open XML*, that define the XML vocabularies for word-processing, spreadsheet, and presentation documents, as well as the packaging of documents that conform to these schemas.

The goal is to enable the implementation of the Office Open XML formats by the widest set of tools and platforms, fostering interoperability across office productivity applications and line-of-business systems, as well as to support and strengthen document archival and preservation, all in a way that is fully compatible with the existing corpus of Microsoft Office documents.

Information technology — Document description and processing languages — Office Open XML File Formats

Part 3:

Markup Compatibility and Extensibility

1. Scope

This Part of ISO/IEC 29500 describes a set of conventions that are used by Office Open XML documents to clearly mark elements and attributes introduced by future versions or extensions of Office Open XML documents, while providing a method by which consumers can obtain a baseline version of the Office Open XML document (i.e., a version without extensions) for interoperability.

Comment [rcj3]: Why is this OOXML-specific? Wasn't the original intent to allow Part 3 (and Part 2) for non-OOXML applications? Yes.
Owner: Rex will review this and draft replacement words.

2. Conformance

2.1 Introduction

The text in this Part of ISO/IEC 29500 is divided into *normative* and *informative* categories. Unless documented otherwise, any feature shall be implemented as specified by the normative text describing that feature in this Part of ISO/IEC 29500. Text marked informative (using the mechanisms described in §6) is for information purposes only. Unless stated otherwise, all text is normative.

Use of the word "shall" indicates required behavior.

Each Part of this multi-part standard has its own conformance clause. The term *conformance class* is used to disambiguate conformance within different Parts of this multi-part standard. This Part of ISO/IEC 29500 has only one conformance class, *MCE* (that is, Markup Compatibility and Extensibility). As such, conformance to that class implies conformance to the whole Part.

2.2 Document Conformance

A document has conformance class MCE if it satisfies the syntax constraints on elements and attributes defined in this Part of ISO/IEC 29500. Document conformance to this Part of ISO/IEC 29500 is purely syntactic.

2.3 Application Conformance

An application implementing this Part of ISO/IEC 29500 has conformance class MCE if any one of the following is true:

- The application is a markup consumer that does not reject any documents of conformance class MCE; or
- The application is a markup producer that is able to produce documents of conformance class MCE

Application conformance to this Part of ISO/IEC 29500 is purely syntactic.

[*Note*: Application conformance to this Part of ISO/IEC 29500 cannot be based on semantics, since the semantics depend on the choice of application-defined extension elements. *end note*]

~~3.~~2. Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 2382-1:1993, *Information technology — Vocabulary — Part 1: Fundamental terms*.

ISO/IEC 10646, *Information technology — Universal Coded Character Set (UCS)*.

ISO/IEC 19757-4:2006, *Information technology — Document Schema Definition Languages (DSDL) — Part 4: Namespace-based Validation Dispatching Language (NVDL)*.

RFC 3986 *Uniform Resource Identifier (URI): Generic Syntax,* The Internet Society, Berners-Lee, T., R. Fielding, and L. Masinter, 2005, http://www.ietf.org/rfc/rfc3986.txt.

RFC 4234 *Augmented BNF for Syntax Specifications: ABNF,* The Internet Society, Crocker, D., P. Overell, 2005, http://www.ietf.org/rfc/rfc4234.txt

The Unicode Consortium. The Unicode Standard, http://www.unicode.org/standard/standard.html.

XML, Tim Bray, Jean Paoli, Eve Maler, C. M. Sperberg-McQueen, and François Yergeau (editors). Extensible Markup Language (XML) 1.0, Fourth Edition. World Wide Web Consortium. 2006. http://www.w3.org/TR/2006/REC-xml-20060816/ [Implementers should be aware that a further correction of the normative reference to XML to refer to the 5th Edition will be necessary when the related Reference Specifications to which this International Standard also makes normative reference and which also depend upon XML, such as XSLT, XML Namespaces and XML Base, are all aligned with the 5th Edition.]

XML Base, Marsh, Jonathan. *XML Base*. World Wide Web Consortium. 2001. http://www.w3.org/TR/2009/REC-xmlbase-20090128/~~http://www.w3.org/TR/2001/REC-xmlbase-20010627/~~

XML Information Set, John Cowan and Richard Tobin (editors). *XML Information Set (Second Edition)*, 4 February 2004. World Wide Web Consortium. http://www.w3.org/TR/2004/REC-xml-infoset-20040204/

XML Namespaces, Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin (editors). *Namespaces in XML 1.0* (Third Edition), 8 December 2009. World Wide Web Consortium. http://www.w3.org/TR/2009/REC-xml-names-20091208/

XML Schema Part 0: Primer (Second Edition), W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-0/

XML Schema Part 1: Structures (Second Edition), W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-1/

Comment [rcj4]: Move to Bibliography as only alluded to in an informative annex.

And also add an entry to Bibliography for RELAX NG.

Also add 29500-1 and -4 to Bibliography.

Comment [rcj5]: Given that this spec no longer uses the term URI, this entry can be removed.

Comment [rcj6]: Not referenced.

XML Schema Part 2: Datatypes (Second Edition), W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-2/

Comment [rcj7]: Move to Bibliography as only alluded to in an informative annex.

~~4.~~3. Terms and Definitions

Comment [rcj8]: Review the newly added text elsewhere to make sure the terms in this clause get updated to reflect the new text.

Make sure numbering is correct

For the purposes of this document, the following terms and definitions apply~~:. Other terms are defined where they appear in *italics* typeface. Terms not explicitly defined in this Part of ISO/IEC 29500 are not to be presumed to refer implicitly to similar terms defined elsewhere.~~

~~Throughout this Part of ISO/IEC 29500, the terms *namespace declaration*, *namespace name*, *qualified name*, *expanded name*, *prefixed name*, *unprefixed name*, and *local name* shall have the meanings as defined in the W3C Recommendation, "Namespaces in XML."~~

3.1
alternate content
set of alternatives of XML markup and character data, of which no more than one shall be processed by a markup consumer based upon the set of XML namespaces understood by the markup consumer

3.2
byte
sequence of 8 bits treated as a unit

Comment [rcj9]: Remove as we don't use this term

3.3
compatibility-rule attribute
XML attribute described in this Part of ISO/IEC 29500 that expresses rules governing markup consumers' behavior when encountering XML elements and attributes from non-understood namespaces

Comment [rcj10]: Remove as we don't use this term

3.4
configuration
set of XML namespace names

Comment [rcj11]: Global change to "MCE configuration"

3.5
ignore
disregard the presence of an element or attribute, processing the markup as if that element or attribute did not exist

3.6
markup consumer
tool that can read and parse a markup document and further conforms to the requirements of a markup specification

Comment [rcj12]: Review the need for markup consumer, producer, and document. Current definitions don't seem quite right anyway. Objections raised as they refer to markup document, which in turn refers to a file format, the conformance of a consumer isn't relevant.

Owner: Murata-san

3.7
markup document
XML document that conforms to the requirements of a markup specification

Comment [rcj13]: See markup consumer

3.8
markup preprocessor
software module, designed for use in the implementation of markup consumers, that follows the rules of this Part of ISO/IEC 29500 to remove or replace all elements and attributes from the Markup Compatibility namespace, all elements and attributes from ignorable non-understood namespaces, and all elements and attributes from subsumed namespaces

3.9
markup producer
tool that can generate a markup document, and conforms to a markup specification

3.10
markup specification
XML-based format definition that incorporates all of the namespaces, elements, attributes, and requirements specified in this Part of ISO/IEC 29500

3.11
mismatch
constraint expressed in an input document that cannot be satisfied under the given configuration

3.12
namespace

3.12.1
ignorable namespace
XML namespace, identified in markup, whose elements and attributes ~~shall be~~are ignored ~~ignored by a markup consumer~~ if the namespace is not an understood namespace~~that does not understand that namespace~~

3.12.2
understood namespace
XML namespace that is included in a configuration~~containing any recognized XML elements or attributes~~

3.13
qualified attribute name
attribute's qualified name

3.14
qualified element name
element's qualified name

3.15
recognize
identify that an XML element, XML attribute, or attribute-value is defined in this Part of ISO/IEC 29500 or in the markup specification against which the containing XML document purports to be conformant

Comment [rcj14]: Make sure we consistently use "MCE processor" instead of this and other variants throughout the spec.

Comment [rcj15]: See markup consumer

Comment [rcj16]: Can this be removed or reformulated? Owner: Murata-san

3.16
term
namespace prefix followed by ":", followed either by a local name or by "*"

~~5.~~4. Notational Conventions

The following typographical conventions are used in ISO/IEC 29500:

1. The first occurrence of a new term is written in italics. [*Example*: The text in ISO/IEC 29500 is divided into *normative* and *informative* categories. *end example*]
2. The tag name of an XML element is written using a distinct style and typeface. [*Example*: The bookmarkStart and bookmarkEnd elements specify … *end example*]
3. The name of an XML attribute is written using a distinct style and typeface. [*Example*: The dropCap attribute specifies … *end example*]
4. The value of an XML attribute is written using a constant-width style. [*Example*: The attribute value of auto specifies … *end example*]
5. The qualified or unqualified name of a simple type, complex type, or base datatype is written using a distinct style and typeface.[*Example*: The possible values for this attribute are defined by the ST_HexColor simple type. *end example*]

~~6.~~5. General Description

This Part of ISO/IEC 29500 is divided into the following subdivisions:

1. Front matter (clauses 1–5);
2. Overview and introductory material (clause 6–7);
3. Main body (clauses 8–10);
4. Annexes

Examples are provided to illustrate possible forms of the constructions described. References are used to refer to related clauses. Notes are provided to give advice or guidance to implementers or programmers.

The following form the normative pieces of this Part of ISO/IEC 29500:

- Clauses 1–4, 5, and 7–9

The following make up the informative pieces of this Part of ISO/IEC 29500:

- Introduction
- Clauses 6 and 10
- All annexes
- All notes and examples

Comment [rcj17]: Spacing looks off above this

Except for whole clauses or annexes that are identified as being informative, informative text that is contained within normative text is indicated in the following ways:

1. [*Example:* code fragment, possibly with some narrative … *end example*]
2. [*Note:* narrative … *end note*]
3. [*Rationale:* narrative … *end rationale*]
4. [*Guidance*: narrative … *end guidance*]

Comment [rcj18]: Consider moving this to previous clause and making this clause informative. And implications on other Parts.

~~7.~~6. Overview

This clause is informative

This Part of ISO/IEC 29500 describes a set of XML elements and attributes whose purpose is to collectively enable producers to explicitly guide consumers in their handling of any XML elements and attributes not understood by the consumer.

These elements and attributes are intended to enable the creation of future versions of and extensions to ISO/IEC 29500, while enabling these desirable compatibility goals:

- A markup producer can produce markup documents that exploit new features defined by versions and extensions, yet remain interoperable with markup consumers that are unaware of those versions and extensions.
- For any such markup document, a markup consumer whose implementation is aware of the exploited versions and extensions can deliver functionality that is enhanced by the markup document's use of those versions and extensions.
- For any such markup document, the markup producer can enable and precisely control graceful degradation that might occur when the markup document is processed by a markup consumer that is unaware of the exploited versions and extensions.

These elements and attributes define particular types of compatibility and extension constructs, as summarized below:

Comment [rcj19]: Rework this so we don't appear to be introducing any new terms. Use plain English.
Owner: John H.

Don't use "recognize"

- Ignorable namespaces specify content that may be disregarded by markup consumers as if it did not exist. This allows markup producers to identify some markup as not core to the document content. Markup consumers that do not recognize the namespace and the capability it represents can disregard it without significant degradation of the document content.
- Must-process elements specify elements that must be processed even if they would otherwise be ignored. This allows markup producers to not lose content nested within ignored content when processed by markup consumers that do not understand the ignored namespace.
- Must-understand namespaces specify namespaces that must be understood by markup consumers in order to process the document. This allows markup producers to set minimum compatibility requirements for consumers.
- Alternate content regions specify alternative representations of document content. This allows markup producers to generate markup alternatives for markup consumers with differing sets of understood namespaces.
- Application-defined extension elements specify a method for defining extensibility points within the markup specification using markup compatibility and extensibility. This allows markup producers to

introduce new features scoped to particular nodes within the markup specification. Markup consumers can disregard these branches of additional capability.

End of informative text

~~8.~~7. Markup Compatibility Fundamentals

~~8.1~~7.1 Error Handling

Comment [rcj20]: Consider moving to the processing model clause

If an MCE processor detects that a document is non-conformant, the MCE processor should ~~signal~~indicate this non-conformance to the consuming application~~; afterwards, the processor may continue normal MCE processing, if possible~~.

~~9.~~8. Markup Compatibility Attributes and Elements

~~9.1~~8.1 Introduction

This subclause specifies the syntactic definitions of all the elements and attributes in the Markup Compatibility namespace.

The Markup Compatibility namespace shall be:

```
http://schemas.openxmlformats.org/markup-compatibility/2006
```

[*Guidance*: External DTD subsets should not specify default values for attributes in the Markup Compatibility namespace, as some non-validating XML processors do not use such default values. *end guidance*]

Attributes within the Markup Compatibility namespace may occur on any XML element, including Markup Compatibility elements.

~~9.2~~8.2 Ignorable Attribute

An Ignorable attribute shall be an attribute in the Markup Compatibility namespace with local name "Ignorable". Its value shall be a whitespace-delimited list of zero or more namespace prefixes, optionally having leading and/or trailing whitespace, where each namespace prefix identifies an ignorable namespace. For each namespace prefix in the list, there shall be an in-scope namespace to which that prefix is bound, and it shall not be the Markup Compatibility namespace. This in-scope namespace is said to be declared as ignorable by this Ignorable attribute.

[*Note*: By default, an ignored element is ignored in its entirety, including its attributes and its content. The processing of an ignored element's contents is enabled using the ProcessContent attribute. *end note*]

[*Example*:

```
<example xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006">
  <foo mce:Ignorable="i1"
    xmlns:i1="http://www.example.com/i1"
    xmlns:i2="http://www.example.com/i2">
    <bar mce:Ignorable="i2">…</bar>
  </foo>
</example>
```

The foo element and the bar element, which is a child of foo, each have an Ignorable attribute. The Ignorable attribute of foo specifies the prefix "i1", which is bound to the in-scope namespace

"http://www.example.com/i1". Thus, the Ignorable attribute of foo declares this namespace as ignorable. The Ignorable attribute of bar specifies the prefix "i2", which is bound to the in-scope namespace "http://www.example.com/i2". Thus, the Ignorable attribute of bar declares this namespace as ignorable. *end example*]

[*Example*:

```
<example xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006">
  <foo mce:Ignorable="i1"
    xmlns:i1="http://www.example.com/i1"
    xmlns:i2="http://www.example.com/i2">
    <bar mce:Ignorable="i2">…</bar>
    <bar mce:Ignorable="i1 i2">…</bar>
    <bar mce:Ignorable="i1alias i2"
xmlns:i1alias="http://www.example.com/i1">…</bar>
  </foo>
</example>
```

The Ignorable attribute of the first bar element declares the namespace "http://www.example.com/i2" as ignorable. The Ignorable attribute of the second bar element declares both "http://www.example.com/i1" and "http://www.example.com/i2" as ignorable, but the former is already declared by the Ignorable attribute of the parent foo element. The Ignorable attribute of the third bar element also declares these two namespaces as ignorable, although the namespace prefix is i1alias rather than i1. Therefore, although the lexical values are different, these three Ignorable attributes are equivalent as far as MCE processing is concerned. *end example*]

[*Example*:

```
<example xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006">
  <foo1 mce:Ignorable="i1">
    <foo2 xmlns:i1="http://www.example.com/i1">…</foo2>
  </foo1>
  <foo3 mce:Ignorable="i2">…</foo3>
  <foo4 xmlns:i2="http://www.example.com/i4"/>
</example>
```

This document is non-conformant for two reasons: First, the foo1 element has an Ignorable attribute, but the value i1 is not bound to an in-scope namespace. Second, the foo3 element also has an Ignorable attribute, but the value i2 is not bound to an in-scope namespace either. *end example*]

~~9.3~~8.3 ProcessContent Attribute

A ProcessContent attribute shall be an attribute in the Markup Compatibility namespace with local name "ProcessContent". Its value shall be a whitespace-delimited list of zero or more terms, optionally having leading and/or trailing whitespace. Each term shall be a namespace prefix followed by ":" followed either by a

local name or by "*".For each term in the list, there shall be an in-scope namespace to which the namespace-prefix part of the term is bound. This in-scope namespace shall not be the Markup Compatibility namespace, and shall be declared as ignorable by an Ignorable attribute at the same element or ancestor. The pair of this in-scope namespace and the local part or "*" in this term is said to be declared as a process content name pair by this ProcessContent attribute.

If (*n1*, *l1*) is the namespace-name and local-name pair of an element, that element matches a process content name pair (*n2*, *l2*) if

1. *n1* and *n2* are the same sequence of characters, and
2. Either
    a. *l1* and *l2* are the same sequence of characters, or
    b. *l2* is "*"

Markup producers shall not generate an element that has an xml:lang or xml:space attribute if that element is identified by a ProcessContent attribute value.

Comment [JH21]: Keep, change, remove? Move this to the processing model clause. Reformulate this as requirements on data rather than on producers. Owner: Murata-san

[*Example*:

```
<example xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006">
  <foo1 mce:Ignorable="i1"
    mce:ProcessContent="i1:bar1"
    xmlns:i1="http://www.example.com/i1"
    xmlns:i2="http://www.example.com/i2">
    <foo2 mce:Ignorable="i2"
      mce:ProcessContent="i2:*">…</foo2>
    <foo3 mce:ProcessContent="i1:bar2">…</foo3>
  </foo1>
</example>
```

The foo1, foo2, and foo3 elements have ProcessContent attributes. That of the foo1 element has a term "i1:bar1", where "i1" is a namespace prefix bound to an in-scope namespace "http://www.example.com/i1", which is declared as ignorable at this element. That of the foo2 element has a term "i2:*", where i2 is a namespace prefix bound to an in-scope namespace "http://www.example.com/i2", which is declared as ignorable at this element. That of the foo3 element has a term "i1:bar2", where i1 is a namespace prefix bound to an in-scope namespace "http://www.example.com/i1", which is declared as ignorable at the parent foo1 element. *end example*]

[*Example*:

```
<example xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006">
  <foo1 xmlns:i2="http://www.example.com/i2">
    <foo2 mce:ProcessContent="i2:*">…</foo2>
  </foo1>
</example>
```

The foo2 element has a ProcessContent attribute. The value is a term "i2:*", where i2 is a namespace prefix bound to an in-scope namespace "http://www.example.com/i2". However, this namespace is not declared as ignorable. As such, this example is non-conformant. *end example*]

[*Example*:

In the following example, extB:Blink is ignorable and is identified by the ProcessContent attribute because extA and extB share the same namespace name and therefore the expanded names match.

```
<Circles
  xmlns="http://schemas.openxmlformats.org/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:extA="http://www.example.com/Circles/extension"
  xmlns:extB="http://www.example.com/Circles/extension"
  mc:Ignorable="extB"
  mc:ProcessContent="extA:Blink" >
  <extB:Blink>
    <Circle Center="0,0" Radius="20" Color="Blue" />
  </extB:Blink>
</Circles>
```

Comment [rcj22]: Needs formatting

end example]

~~9.4~~8.4 MustUnderstand Attribute

A MustUnderstand attribute shall be an attribute in the Markup Compatibility namespace with local name “MustUnderstand”. Its value shall be a whitespace-delimited list of zero or more namespace prefixesoptionally having leading and/or trailing whitespace. For each namespace prefix in the list, there shall be an in-scope namespace name to which that prefix is bound, and this namespace shall not be the Markup Compatibility namespace. This in-scope namespace is said to be declared as a must-understand namespace by this MustUnderstand attribute.

[*Note*: §8.6 clarifies the rules for processing the MustUnderstand attribute when it is applied to a Choice or Fallback element, or when it is applied to a descendant element of one of those elements. *end note*]

Comment [JH23]: Keep? If so, update the section reference. Probably not since semantics are handled separately and explicitly cover combinations of MCE attributes/elements. Probably covered, but we need to check. If so, can delete this.
Owner: Murata-san.

[*Example*:

```
<example
  xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:e1="http://www.example.com/e1"
  mce:MustUnderstand="e1">
</example>
```

In this example, the root element has a MustUnderstand attribute. The value contains e1, which is bound to an in-scope namespace name "http://www.example.com/e1". *end example*]

[*Example*:

```
<example
  xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:e1="http://www.example.com/e1">
  <foo mce:MustUnderstand="e1 e2"/>
</example>
```

In this example, the MustUnderstand attribute of the element foo contains e1 and e2. Although e1 is bound to an in-scope namespace name "http://www.example.com/e1", e2 is not. As such, this document is non-conformant. *end example*]

~~9.5~~8.5 ExtensionElements Attribute

<<This is being proposed as a new feature>>

Comment [rcj24]: Agreed to rewrite this as a new attribute suspending MCE processing. Owner: Murata-san + Francis + Alex

An ExtensionElements attribute shall be an attribute in the Markup Compatibility namespace with local name “ExtensionElements”. The value of this attribute shall be a white-space delimited list of zero or more prefixed names, optionally having leading and/or trailing whitespace. For each prefixed name in the list, there shall be an in-scope namespace to which its namespace prefix is bound, and this namespace shall not be the Markup Compatibility namespace. The pair of this in-scope namespace and the local part of each prefixed name is said to be declared as an extension element name pair by this ExtensionElements attribute.

An element shall be an application-defined extension element if the pair of the namespace name and local name of this element is declared as an extension-element name pair by an ExtensionElements attribute of this element or of any ancestor element.

If the namespace name of the root element begins with "http://schemas.openxmlformats.org/", the following set of extension-element name pairs shall be assumed, and need not be declared by ExtensionElements attributes:

- (http://schemas.openxmlformats.org/drawingml/2006/chart, ext)
- (http://schemas.openxmlformats.org/drawingml/2006/chartDrawing, ext)
- (http://schemas.openxmlformats.org/drawingml/2006/main, ext)
- (http://schemas.openxmlformats.org/drawingml/2006/spreadsheetDrawing, ext)

- (http://schemas.openxmlformats.org/presentationml/2006/main, ext)
- (http://schemas.openxmlformats.org/spreadsheetml/2006/main, ext)

If the namespace name of the root element begins with "http://purl.oclc.org/ooxml/", the following set of extension-element name pairs shall be assumed, and need not be declared by ExtensionElements attributes:

- (http://purl.oclc.org/ooxml/drawingml/chart, ext)
- (http://purl.oclc.org/ooxml/drawingml/chartDrawing, ext)
- (http://purl.oclc.org/ooxml/drawingml/main, ext)
- (http://purl.oclc.org/ooxml/drawingml/spreadsheetDrawing, ext)
- (http://purl.oclc.org/ooxml/presentationml/main, ext)
- (http://purl.oclc.org/ooxml/spreadsheetml/main, ext)

[*Example*:

```
<example
  xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:e1="http://www.example.com/e1"
  mce:ExtensionElements="e1:foo">

  <e1:foo>…</e1:foo>
</example>
```

The root element has the ExtensionElements attribute. The attribute value is e1:foo, where e1 is the prefix for the namespace "http://www.example.com/e1". Thus, (http://www.example.com/e1, foo) is declared as an extension-element name pair. The child element e1:foo is an application-defined extension element as the namespace of this element is http://www.example.com/e1, and the local name of this element is foo. *end example*]

[*Example*:

```
<w:document
  xmlns:mce ="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:w ="http://schemas.openxmlformats.org/wordprocessingml/2006/main"
  xmlns:dchrt = "http://purl.oclc.org/ooxml/drawingml/chart"
  xmlns:cdr = "http://purl.oclc.org/ooxml/drawingml/chartDrawing"
  xmlns:a = "http://purl.oclc.org/ooxml/drawingml/main"
  xmlns:xdr =  "http://purl.oclc.org/ooxml/drawingml/spreadsheetDrawing"
  xmlns:p ="http://purl.oclc.org/ooxml/presentationml/main"
  xmlns:sml ="http://purl.oclc.org/ooxml/spreadsheetml/main"
  mce:ExtensionElements = "dchrt:ext cdr:ext a:ext xdr:ext p:ext sml:ext">
  …
</w:document>
```


The root element w:document has the ExtensionElements attribute and the attribute value is "dchrt:ext cdr:ext a:ext xdr:ext p:ext sml:ext". Thus, this attribute declares six pairs:

- (http://schemas.openxmlformats.org/drawingml/2006/chart, ext)
- (http://schemas.openxmlformats.org/drawingml/2006/chartDrawing, ext)
- (http://schemas.openxmlformats.org/drawingml/2006/main, ext)
- (http://schemas.openxmlformats.org/drawingml/2006/spreadsheetDrawing, ext)
- (http://schemas.openxmlformats.org/presentationml/2006/main, ext)
- (http://schemas.openxmlformats.org/spreadsheetml/2006/main, ext)

end example]

[*Example*:

```
<w:document
  xmlns:mce ="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:w ="http://schemas.openxmlformats.org/wordprocessingml/2006/main"
  xmlns:dchrt = "http://purl.oclc.org/ooxml/drawingml/chart"
  xmlns:cdr = "http://purl.oclc.org/ooxml/drawingml/chartDrawing"
  xmlns:a= "http://purl.oclc.org/ooxml/drawingml/main"
  xmlns:xdr =  "http://purl.oclc.org/ooxml/drawingml/spreadsheetDrawing"
  xmlns:p ="http://purl.oclc.org/ooxml/presentationml/main"
  xmlns:sml ="http://purl.oclc.org/ooxml/spreadsheetml/main">
  …
</w:document>
```

In this example, the root element does not have the ExtensionElements attribute. However, as the namespace of the root element is "http://schemas.openxmlformats.org/wordprocessingml/2006/main", the six pairs in the previous example are assumed to be declared. *end example*]

[NOTE: If a markup specification references the first edition of MCE, it is recommended to make a newer version of that markup specification reference the second edition of MCE but prohibit the explicit use of ExtensionElements. This prohibition ensures that existing programs for the current version of the markup specification continue to work for new documents conformant to the newer version.]

Comment [rcj25]: Recast this as a note updated once this subclause rewrite is complete.
Owner: Rex

[Should we put the ExtensionElements in a new namespace thus allowing MCE V1 implementations ignore this attribute?][Bellevue: Yes.]

Issue: Should we allow other mechanisms (e.g., APIs) for declaring application-defined extension elements. [Bellevue: Yes; please factor this into the rewrite of this subclause.]

~~9.6~~8.6 Alternate-Content Element

An AlternateContent element shall be an element in the Markup Compatibility namespace with local name "AlternateContent. An AlternateContent element shall not have unqualified attributes, but may have

qualified attributes. The namespace of each qualified attribute shall be either the Markup Compatibility namespace or a namespace declared as ignorable.

[DRAFTING NOTE: Are AlternateContent elements allowed to have attributes of ignorable and understood namespace names (in other words, should we consider configurations)? Or, should we allow namespaces for future versions of MCE only, and disallow configurations to contain such namespaces?]

An AlternateContent element shall contain one or more Choice child elements, optionally followed by a single Fallback child element. No other elements in the Markup Compatibility namespace may appear as child elements. Elements in other namespaces may appear as preceding, intervening, or trailing child elements, but the namespace of such a child element shall be declared as ignorable.

[DRAFTING NOTE: The same concern here. Are AlternateContent elements allowed to have elements of ignorable and understood namespace names (in other words, should we consider configurations)?]

[*Note*: The AlternateContent element can appear as the root element of a markup document. *end note*]

An AlternateContent element shall not have xml:lang or xml:space attributes.

[*Example*:

```
<example
  xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:e1="http://www.example.com/e1"
  xmlns:e2="http://www.example.com/e2">

  <mce:AlternateContent mce:MustUnderstand="e1">
    <mce:Choice Requires="e2">…</mce:Choice>
  </mce:AlternateContent>
</example>
```

In this example, the AlternateContent element has a MustUnderstand attribute and no other attributes. The AlternateContent element has a Choice element as a child but has no other child elements. ~~Assuming that the Choice element satisfies requirements shown in §**Error! Reference source not found.**, this document is conformant.~~ *end example*]

Comment [rcj26]: [Bellevue: Options
A.Allow foreign elements and attributes in ignorable namespaces only
B.Allow foreign elements and attributes in all namespaces, but recommend that those namespaces be ignorable
C.Allow foreign elements and attributes in all namespaces, but require they be MustUnderstand namespaces
]

Comment [rcj27]: [Bellevue: same as above]

[*Example*:

```
<example
  xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:i1="http://www.example.com/i1"
  xmlns:e1="http://www.example.com/e1">

  <mce:AlternateContent mce:Ignorable="i1" i1:foo="">
    <i1:bar/>
    <mce:Choice Requires="e1">…</mce:Choice>
    <i1:bar/>
  </mce:AlternateContent>
</example>
```

In this example, the AlternateContent element has an Ignorable attribute and no other attributes. The AlternateContent element has a Choice element and elements from another namespace as children. Because the other elements are declared as ignorable ~~and, assuming that the Choice element satisfies requirements shown in §**Error! Reference source not found.**,~~ this document is conformant. *end example*]

[EDITORIAL NOTE: Drop this example? Or, use a better namespace name for MCE V2?]

Comment [rcj28]: Bellevue:

Either delete this and the following example, or reformulate both based on the final text of the rewrite of "children of ignorable content".

[*Example*:

```
<example
  xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:i1="http://www.example.com/i1"
  xmlns:e1="http://www.example.com/e1">

  <mce:AlternateContent i1:foo="">
    <i1:bar/>
    <mce:Choice Requires="e1">…</mce:Choice>
    <i1:bar/>
  </mce:AlternateContent>
</example>
```

This example differs from the previous one in that the Ignorable attribute has been removed. Neither the i1:foo attribute nor the two i1:bar elements belong to ignorable namespaces, so this document is non-conformant. *end example*]

~~9.7~~8.7 Choice Element

A Choice element shall be an element in the Markup Compatibility namespace with local name "Choice". Parent elements of Choice elements shall be AlternateContent elements. A Choice element shall have an unqualified attribute with local name "Requires" and shall have no other unqualified attributes. The value of

the Requires attribute shall be a whitespace-delimited list of one or more namespace prefixes, optionally having leading and/or trailing whitespace.

[*Note*: With the exception of empty lists, the syntactical constraints associated with the Requires attribute are the same as those associated with the MustUnderstand attribute. *end note*]

A Choice element may have qualified attributes. The namespace of each qualified attribute shall be either the Markup Compatibility namespace or a namespace declared as ignorable.

[DRAFTING NOTE: Florian dissenting.]

Comment [rcj29]: [Bellevue: see previous drafting note.]

A Choice element shall not have xml:lang or xml:space attributes.

[*Example*:

```
<example
  xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:i1="http://www.example.com/i1"
  xmlns:e1="http://www.example.com/e1">

  <mce:AlternateContent mce:Ignorable="i1" >
    <mce:Choice Requires="e1" i1:foo="">…</mce:Choice>
  </mce:AlternateContent>
</example>
```

In this example, the Choice element specifies the i1:foo attribute. The namespace of this attribute is declared as ignorable at the parent AlternateContent element. This document is conformant, but would be non-conformant if the i1 namespace was not ignorable. [EDITORIAL NOTE: What happens when this namespace is understood?] *end example*]

Comment [rcj30]: Bellevue: The answer to this depends on our answer to the Drafting Note regarding foreign child attributes of MCE elements.

~~9.8~~8.8 Fallback Element

A Fallback element shall be an element in the Markup Compatibility namespace with local name “Fallback”. Parent elements of Fallback elements shall be AlternateContent elements.

A Fallback element shall not have unqualified attributes. A Fallback element may have qualified attributes. The namespace of each qualified attribute shall be either the Markup Compatibility namespace or a namespace declared as ignorable.

A Fallback element shall not have xml:lang or xml:space attributes.

[*Example*:

```
<example
  xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:i1="http://www.example.com/i1"
  xmlns:e1="http://www.example.com/e1">

  <mce:AlternateContent mce:Ignorable="i1" >
    <mce:Choice Requires="e1" >…</mce:Choice>
    <mce:Fallback i1:foo="">…</mce:Fallback>
  </mce:AlternateContent>
</example>
```

In this example, the Fallback element specifies the i1:foo attribute. The namespace of this attribute is declared as ignorable at the parent AlternateContent element. This document is conformant but would be non-conformant if the i1 namespace were not ignorable. [EDITORIAL NOTE: The same issue here.] *end example*]

~~10.~~9. Application-Defined Extension Elements

A markup specification using Markup Compatibility elements and attributes might define one or more specific extension elements in the namespaces it defines. *Extension elements* suspend Markup Compatibility processing within their content. Except as noted below, within the content of an extension element, markup consumers shall not treat elements and attributes from non-understood namespaces as Markup Compatibility errors. Similarly, under the same conditions, markup consumers shall disregard elements and attributes from the Markup Compatibility namespace.

A specification for an element nested somewhere within an extension-element might require a markup consumer to re-establish Markup Compatibility processing. Within the scope of such a nested element and its content, a markup consumer shall disregard all Markup Compatibility attributes that were encountered on elements outside of the element that re-establishes Markup Compatibility processing. Within the scope of such a nested element, a markup consumer might understand a set of namespaces that is different from the set of namespaces understood at the point in the markup where the extension element was encountered.

The following examples illustrate two uses of application-defined extension elements:

[*Example*: Example 9–1. An application-defined XML island

An extension element can be used to introduce an *island* of unprocessed XML whose markup is otherwise unconstrained by the application's specification. The specification of the island element can further require preservation of the contents of the island by markup processors. *end example*]

[*Example*: Example 9–2. An application-defined add-in element

Some markup specifications and markup consumers can use an extension element to implement an add-in model. In an add-in model, the specification for the contents of the extension element is separate from the specification for the extension element itself.

The specification for some particular nested content can include support for Markup Compatibility elements and attributes, while the specification for other nested content could omit such support. If the specification for the nested content does include support for Markup Compatibility elements and attributes, the Markup Compatibility processing state is reset temporarily for processing of the nested content. Any Ignorable attribute-value associated with an extension element or any of its ancestor elements is “forgotten” during the processing of content nested within that extension element. In an add-in model the set of namespaces assumed to be understood when processing descendant elements of an extension element is completely unrelated to the set of understood namespaces when that extension element itself is processed.

In this example, the "Circles" specification includes an extension AddIn element, allowing nested markup to be handled by a markup consumer that does not process Circle markup. The specification for the nested "TextFlow" markup does not provide for the processing of Markup Compatibility elements and attributes.

```
<Circles
  xmlns="http://schemas.openxmlformats.org/Circles/v1"
  xmlns:mc=http://schemas.openxmlformats.org/markup-
  compatibility/2006
  xmlns:v2="http://schemas.openxmlformats.org/Circles/v2"
  mc:Ignorable="v2 ">
  <Circle Center="0,0" Radius="20" Color="Blue"
    v2:Opacity="0.5" />
  <Circle Center="25,0" Radius="20" Color="Black"
    v2:Opacity="0.5" />
  <Circle Center="50,0" Radius="20" Color="Red"
    v2:Opacity="0.5" />
  <Circle Center="13,0" Radius="20" Color="Yellow"
    v2:Opacity="0.5" />
  <Circle Center="38,0" Radius="20" Color="Green"
    v2:Opacity="0.5" />
  <AddIn Center="25,10" Radius="10" CodeBase=
    "http://www.openxmlformats.org/code/TextFlowAddin.jar">
    <TextFlow
      xmlns="http://schemas.openxmlformats.org/TextFlow/v1">
      <!--
        Because the TextFlow specification does not make use
        of Markup Compatibility elements and attributes,
        the TextFlow processor would consider the presence
        of an mc:Ignorable attribute to be an error condition.
        Because the TextFlow specification is completely
        unaware of all versions of the Circles specification,
        the TextFlow processor would also consider the
        presence of a Circle or v2:Ellipse element to be an
        error condition.
      -->
      <Paragraph>How are <Bold>you</Bold>?</Paragraph>
    </TextFlow>
  </AddIn>
</Circles>
```

end example]

Comment [JH31]: TBD

~~11.~~10. Semantic Definitions and Reference Preprocessing Model

~~11.1~~10.1 Overview

For a given input document and configuration, this clause defines the output document that shall be created by the MCE processor. This clause further specifies the condition mismatches between a given input document and configuration that shall be signaled by the MCE processor; however, it does not specify exactly when such mismatches are to be signaled. If an MCE processor detects a mismatch, it shall signal this mismatch to the consuming application and it may continue normal MCE processing.

The MCE processor is initialized with the configuration representing the set of namespaces that are to be treated as understood.

This clause defines the semantics by a processing model, which has the following four steps:

[DRAFTING NOTE: Should the input and output of this processing model be Infosets as specified in W3C XML Information Set? If we use it, in-scope namespaces are attached to all elements (by propagation). But xml:base is not propagated in Infosets. Should rather use the RELAX NG data model (which propagates xml:base) or other data models?]

Comment [rcj32]: [Bellevue: Inclined to introduce explicit text for copying attributes in the XML namespace when their parent elements are unwrapped or AlternateContent, Choice or Fallback elements are replaced by the selected option.]

1. Step 1 defines which elements and attributes are marked as ignored or unwrapped. This definition takes into consideration Ignorable, ProcessContent, and ExtensionElements attributes, but does not take into consideration MustUnderstand attributes or AlternateContent, Choice, or Fallback elements. Elements or attributes inside application-defined extension elements are not marked as ignored or unwrapped.
2. Step 2 defines the semantics of alternate content blocks. It specifies which Choice or Fallback element of each AlternateContent element is selected. This definition takes into consideration ExtensionElements attributes as well as AlternateContent, Choice, and Fallback elements, but does not take into consideration Ignorable, ProcessContent, or MustUnderstand attributes. Choice or Fallback elements inside application-defined extension elements are not marked as selected.
3. Step 3 applies the results from Steps 1 and 2 to construct the output document. ~~defines the relationship between the input document and the output document constructed by the MCE processor, based on the definitions above.~~
4. ~~3.~~ Step 4 defines the semantics of MustUnderstand attributes and further defines detection of elements or attributes in namespaces that are neither ignorable nor understood. Neither MustUnderstand attributes nor elements or attributes in namespaces that are neither ignorable nor understood that exist inside application-defined extension elements are examined.

However, MCE processors are not required to carry out these four steps. MCE processors are conformant as long as output documents created and mismatches signaled from given input documents and configurations are the same as those created and signaled by the four steps.

[*Note*: Because Markup Compatibility processing is not performed inside extension elements, a document might still contain elements and attributes in the Markup Compatibility namespace after Markup Compatibility processing has been applied. *end note*]

~~11.2~~10.2 Step 1: Ignoring and Unwrapping

An element shall be marked as ignored if all of the following conditions are satisfied:

1. The namespace of this element is declared as ignorable by an Ignorable attribute of this element or of some ancestor element;
2. The namespace of this element is not included in the given configuration;
3. This element does not match any process-content name pairs declared by this element or some ancestor; and
4. This element is neither an application-defined extension element nor a descendant of an application-defined extension element.

An attribute shall be marked as ignored if all of the following conditions are satisfied:

1. The namespace of this attribute is declared as ignorable by an Ignorable attribute of the element having this attribute or of some ancestor element;
2. The namespace of this attribute is not included in the given configuration; and
3. This attribute does not belong to an application-defined extension element or a descendant of an application-defined extension element

An element shall be marked as unwrapped if all the following conditions are satisfied:

1. The namespace of this element is declared as ignorable by an Ignorable attribute of this element or of some ancestor element;
2. The namespace of this element is not included in the given configuration;
3. This element matches a process-content name pair declared by this element or some ancestor; and
4. This element is neither an application-defined extension element nor a descendant of an application-defined extension element.

[DRAFTING NOTE: Should we disallow elements matching a declared process content name pair to have xml:lang and xml:space (without disallowing xml:base, for example)? 10.1.2 now contains "Markup producers shall not generate an element that has an xml:lang or xml:space attribute if that element is identified by a ProcessContent attribute value. "]

Comment [rcj33]: Bellevue: See copying comment above.

[*Example*:

```
<example
  xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:e1="http://www.example.com/e1"
  mce:Ignorable="e1"
  mce:ProcessContent="e1:bar"
  mce:ExtensionElements="e1:baz"
  e1:foo="">

  <e1:foo><fooChild/></e1:foo>
  <e1:bar><barChild/></e1:bar>
  <e1:baz e1:baz=""><e1:bazChild/></e1:baz>
</example>
```

The namespace "http://www.example.com/e1" is declared as ignorable. A pair ("http://www.example.com/e1", bar) is declared as a process content name pair, while ("http://www.example.com/e1", baz) is declared as an extension element name pair. Suppose that a given configuration does not contain "http://www.example.com/e1". Then, the e1:foo attribute and the e1:foo element are marked as ignored, and the e1:bar element is marked as unwrapped. However, the e1:baz element is not marked as either ignored or unwrapped, as it is an application-defined extension element. Likewise, neither the e1:baz attribute nor the e1:bazChild element are marked. Neither fooChild nor barChild are marked as ignored or unwrapped although their parents are marked as ignored or unwrapped. *end example*]

~~11.3~~10.3 Step 2: Selecting Alternates

A Choice element shall be marked as selected if the following conditions are satisfied:

1. Each of the namespaces specified by the Requires attribute of this element is included in the given configuration;
2. No elder-sibling Choice element is marked as selected; and
3. The element is not a descendant of an application-defined extension element.

A Fallback element shall be marked as selected if the following conditions are satisfied:

1. No elder-sibling Choice element is marked as selected; and
2. The element is not a descendant of an application-defined extension element.

[*Example*:

```
<example
  xmlns:mce="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:n1="http://www.example.com/n1"
  xmlns:n2="http://www.example.com/n2"
  xmlns:n3="http://www.example.com/n3">

  <mce:AlternateContent>
    <mce:Choice Requires="n1 n2">                    <!-- Choice #1 -->
      <mce:AlternateContent>
        <mce:Choice Requires="n3">...</mce:Choice>     <!-- Choice #1-1 -->
        <mce:Fallback>...</mce:Fallback>               <!-- Fallback #1-1 -->
      </mce:AlternateContent>
    </mce:Choice>
    <mce:Choice Requires="n1">                       <!-- Choice #2 -->
      <mce:AlternateContent>
        <mce:Choice Requires="n3">...</mce:Choice>     <!-- Choice #2-1 -->
        <mce:Fallback>...</mce:Fallback>               <!-- Fallback #2-1 -->
      </mce:AlternateContent>
    </mce:Choice>
    <mce:Fallback>...</mce:Fallback>                 <!-- Fallback #1 -->
  </mce:AlternateContent>
</example>
```

The child of the root element is an AlternateContent element. Given that a configuration contains three namespaces, namely “http://www.example.com/n1”, “http://www.example.com/n2”, and “http://www.example.com/n3”, then Choice #1, Choice #1-1, and Choice #2-1 are marked as selected, and Choice #2, Fallback #1, Fallback #1-1, and Fallback #2-1 are not. Note that Choice #2-1, which is marked as selected, appears under Choice #2, which is not. *end example*]

[DRAFTING NOTE: Can elements marked as unwrapped specify xml:lang and xml:base? If they are prohibited, what about xml:base and other inherited attributes in markup vocabularies?]

Comment [rcj34]: See copying issue above.

[DRAFTING NOTE: Can elements marked as unwrapped specify @Ignorable, @ProcessContent, @MustUnderstand, or @ExtensionElements?]

Comment [rcj35]: Bellevue
Options:
A.Such attributes are mismatch errors and must be signalled
B.Such attributes are used, except for MustUnderstand, which is thrown away (status quo)
C. Propagate MustUnderstand
D. See note below in Step 3.

Option A looks best.

~~11.4~~10.4 Step 3: Combining Ignoring and Selecting

This step constructs an output document based on Steps 1 and 2. The output document is identical to the input document except that:

1. Each element marked as ignored, together with its attributes and contents, shall be removed.
2. Each element marked as unwrapped shall be replaced by its content. [*Note*: The attributes will be lost. *end note*.]

3. ~~Each MustUnderstand attribute at such an element shall be examined as specified in Step 4 before it is unwrapped.~~
4. Each attribute marked as ignored shall be removed.
5. Each of the Ignorable, ProcessContent, and ExtensionElements attributes shall be removed unless it belongs to an application-defined extension element or its descendant.
6. Each AlternateContent element shall be replaced by the content of the selected child Choice or Fallback element if there is such a selected child, and shall be deleted otherwise.
7. ~~Each MustUnderstand attribute at this AlternateContent element and the selected Choice or Fallback element shall be examined as specified in Step 4 before this AlternateContent is replaced.~~
8. If some child element of this AlternateContent element is not marked as ignored and is neither Choice nor Fallback element, a mismatch shall be signaled.

[*Note*: The content of a selected Choice or Fallback element does not appear in the output document if some ancestor of this element is ignored or some ancestor Choice or Fallback element is not selected. *end note*]

[DRAFTING NOTE: At least one example of applying Step 3 is needed.]

[DRAFTING NOTE: If we stick to the current semantics which allow @MustUnderstand attributes at unwrapped elements and ACBs, we cannot strikeout the sub-bullet under the 2nd bullet and the 1st sub-bullet under the 5th bullet.]

~~11.5~~10.5 Step 4: MustUnderstand and Non-Ignorable/Non-Understood Namespaces

Each MustUnderstand attribute is examined unless it belongs to an application-defined extension element or descendant of an application-defined extension element. If some of the namespaces declared by this attribute are not in the configuration, a mismatch shall be signaled. Each of the examined MustUnderstand attributes shall then be deleted.

Each element is examined unless it is an application-defined extension element or a descendant of an application-defined extension element. If the namespace of this element is not included in the configuration, a mismatch shall be signaled.

Each qualified attribute is examined unless it belongs to an application-defined extension element or a descendant of an application-defined extension element. If the namespace of this qualified attribute is not included in the configuration, a mismatch shall be signaled.

[*Note*: With the exception of those in application-defined extension elements, elements and attributes in the Markup Compatibility namespace do not appear in the output document. *end note*]

[DRAFTING NOTE: Should we mention namespace declarations at unwrapped elements, AlternateContent, Choice, and Fallback elements? WG4 is leaning toward infoset-like data models, which propagate namespace declarations.]

Comment [rcj36]:

Comment [rcj37]:

Comment [rcj38]: Agreed. Owner: Murata-san.

Comment [rcj39]: See the comment in the previous step.

Comment [rcj40]: Bellevue: Agree that this is reasonable.

~~11.6~~10.6 Justification of Ignorable Foreign Children of AlternateContent

This subclause is informative.

Handling extensions to AlternateContent is a specific case of the general question of how to handle future extensions to MCE. Child elements of AlternateContent are currently limited to Choice, Fallback or ignorable elements. The intent of allowing ignorable elements under an AlternateContent element is to provide for the possibility of future extensions to AlternateContent. An additive extension (i.e., a new element) as a child of AlternateContent declared as ignorable allows a version 1 MCE processor to not fail when encountering that new element. A version 2 MCE processor would understand both the v1 and v2 MCE namespaces. The presence of a non-ignorable v2 MCE element under AlternateContent other than Choice or Fallback would cause v1 MCE processors to fail with an MCE error.

This implies that we ought to be mindful of such compatibility issues if we ever take up future extensions to MCE in general. AlternateContent is a special case that is called out in the current text of this Part because it involves an element and child elements. Other MCE constructs are attributes and thus do not have such considerations. If we do extend MCE in the future, the use of v1 MCE constructs to achieve that would naturally prevent backward compatibility problems for v1 MCE processors. The only category of future changes this would preclude are changes to existing v1 MCE constructs without changing their namespace. However, it would be straightforward and reasonable to create such a changed element (or, indeed, a new attribute) in a new namespace. For example, a new version of AlternateContent could be created in a v2 MCE namespace. While v2 MCE constructs would allow new and useful functionality, it would be incumbent on a producer using them in a document to be mindful of how the document will be handled by a v1 MCE processor.

It was recognized that there is some debate among XML experts whether extensions to existing namespaces should use the existing namespace or a new namespace. The use of a new namespace in conjunction with MCE constructs allows for extensions to MCE without causing backward-compatibility problems.

Comment [JH41]: Needs wordsmithing – this was an e-mail John sent during the London 2012 meeting to capture some discussion
Owner: John

[DRAFTING NOTE: Florian dissenting. Murata is sympathetic.]

End of informative text.

Annex A.
(informative)
Primer

This annex is informative.

Comment [rcj42]: Need to add a conceptual overview for each subclause.
Owner: John

A.1 Example: Ignorable Attribute

Input document:

```
<Circles xmlns="http://www.example.com/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:v2="http://www.example.com/Circles/v2"
  xmlns:v3="http://www.example.com/Circles/v3"
  mc:Ignorable="v2 v3">

  <Circle Center="0,0" Radius="20" Color="Blue"
    v2:Opacity="0.5" v3:Luminance="13"/>
<Circles>
```

Three namespaces in this document, namely “http://www.example.com/Circles/v1”, “http://www.example.com/Circles/v2”, and “http://www.example.com/Circles/v3” capture three versions of a markup specification. Version 1 introduces Circle elements of the namespace for version 1. Version 2 introduces the Opacity attribute of the namespace for version 2. Version 3 introduces the Luminance attribute of the namespace for version 3. In this document, both the namespace for version 2 and that for version 2 are declared as ignorable.

First, suppose that the configuration contains the namespaces for versions 1, 2, and 3. Then, the output document contains the Opacity and Luminance attributes.

Output document:

```
<Circles xmlns="http://www.example.com/Circles/v1"
  xmlns:v2="http://www.example.com/Circles/v2"
  xmlns:v3="http://www.example.com/Circles/v3">

  <Circle Center="0,0" Radius="20" Color="Blue"
      v2:Opacity="0.5"
      v3:Luminance="13"/>
</Circles>
```

Second, suppose that the configuration contains the namespaces for versions 1 and 2 but not the one for version 3. Then, the output document contains the Opacity attribute but does not contain the Luminance attributes.

Output document:

```
<Circles xmlns="http://www.example.com/Circles/v1"
  xmlns:v2="http://www.example.com/Circles/v2">

  <Circle Center="0,0" Radius="20" Color="Blue"
      v2:Opacity="0.5"/>
</Circles>
```

Third, suppose that the configuration contains the namespace for version 1 but not those for versions 2 or 3. In this case, the output document contains neither the Opacity attributes nor the Luminance attributes.

Output document:

```
<Circles xmlns="http://www.example.com/Circles/v1">
  <Circle Center="0,0" Radius="20" Color="Blue"/>
</Circles>
```

A.2 Example: Ignorable and ProcessContent Attributes

Input document:

```
<Circles
  xmlns="http://www.example.com/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:v2="http://www.example.com/Circles/v2"
  mc:Ignorable="v2"
  mc:ProcessContent="v2:Blink">

  <v2:Watermark Opacity="v0.1">
    <Circle Center="0,0" Radius="20" Color="Blue"/>


  </v2:Watermark>
  <v2:Blink>
    <Circle Center="13,0" Radius="20" Color="Yellow"/>

  </v2:Blink>
</Circles>
```

Two namespaces in this document, namely “http://www.example.com/Circles/v1” and “http://www.example.com/Circles/v2”, represent two versions of a markup specification. Version 1 introduces Circles and Circle elements of the namespace for version 1. Version 2 introduces Watermark and Blink elements of the namespace for version 2. The namespace for version 1 is declared as ignorable and the Blink element matches a process content name pair (“http://www.example.com/Circles/v2”, Blink) declared at the root element.

First, suppose that a configuration contains the namespaces for Versions 1 and 2. Then, the output document contains all elements in the input document.

Output document:

```
<Circles
  xmlns="http://www.example.com/Circles/v1"
  xmlns:v2="http://www.example.com/Circles/v2">

  <v2:Watermark Opacity="v0.1">
    <Circle Center="0,0" Radius="20" Color="Blue"/>
  </v2:Watermark>
  <v2:Blink>
    <Circle Center="13,0" Radius="20" Color="Yellow"/>
  </v2:Blink>
</Circles>
```

Second, suppose that a configuration contains the namespace for version 1 but not the one for version 2. Then, the output document does not contain the Watermark and Blink elements, which are of the namespace for version 2. However, the content of the Blink element is retained, since this element matches a declared process content name pair.

Output document:

```
<Circles
  xmlns="http://www.example.com/Circles/v1">

  <Circle Center="13,0" Radius="20" Color="Yellow"/>

</Circles>
```

A.3 Example: Non-Ignorable and Non-Understood Namespace

Input document:

```
<Circles
  xmlns="http://www.example.com/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
  xmlns:v2="http://www.example.com/Circles/v2">

  <Circle Center="0,0" Radius="20" Color="Blue"

    v2:Opacity="0.5" />
</Circles>
```

Two namespaces in this document, namely “http://www.example.com/Circles/v1” and “http://www.example.com/Circles/v2”, represent two versions of a markup specification. Version 1 introduces Circles and Circle elements of the namespace for version 1. Version 2 introduces an Opacity attribute of the namespace for version 2.

First, suppose that a configuration contains the namespaces for versions 1 and 2. Then, the output document is identical to the input document, with the possible exception of omitting the declaration of the Markup Compatibility namespace.

Second, suppose that a configuration contains the namespace for version 1 but not the one for version 2. Then, the MCE processor will report a mismatch error when the Opacity attribute is examined in Step 4.

A.4 Example: MustUnderstand Attribute

Input document:

```
<Circles
  xmlns="http://www.example.com/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:v2="http://www.example.com/Circles/v2"
  mc:MustUnderstand="v2">

  <Circle Center="0,0" Radius="20" Color="Blue"

    v2:Opacity="0.5" />
</Circles>
```

This document is similar to the previous example. The only difference is the addition of the MustUnderstand attribute at the root element.

The MCE processor behaves the same, except that the mismatch error is reported when the root element is examined in Step 4.

A.5 Example: AlternateContent Element

Input document:

```
<Circles
  xmlns="http://www.example.com/Circles/v1"
```

```
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:v2="http://www.example.com/Circles/v2"
  xmlns:v3="http://www.example.com/Circles/v3"
  mc:Ignorable="v2 v3">

  <mc:AlternateContent>
    <mc:Choice Requires="v3">
      <v3:Circle Center="0,0" Radius="20" Color="Blue"
        Opacity="0.5" Luminance="13"/>
    </mc:Choice>
    <mc:Fallback>
      <LuminanceFilter Luminance="13">
         <Circle Center="0,0" Radius="20" Color="Blue"
           v2:Opacity="0.5"/>
       </LuminanceFilter>
     </mc:Fallback>
  </mc:AlternateContent>
</Circles>
```

Three namespaces in this document, namely "http://www.example.com/Circles/v1", "http://www.example.com/Circles/v2", and "http://www.example.com/Circles/v3" capture three versions of a markup specification. Version 1 introduces LuminanceFilter and Circle elements of the namespace for version 1. Version 2 introduces the Opacity attribute of the namespace for version 2. Version 3 introduces Circle elements of the namespace for version 3. Both the namespace for version 2 and that for version 2 are declared as ignorable.

First, suppose that the configuration contains the namespaces for versions 1, 2, and 3. Then, since the Choice element is selected, the output document contains Circle elements of the namespace for version 3 but does not contain the LuminanceFilter element.

Output document:

```
<Circles
  xmlns="http://www.example.com/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:v2="http://www.example.com/Circles/v2"
  xmlns:v3="http://www.example.com/Circles/v3"
  mc:Ignorable="v2 v3">

  <v3:Circle Center="0,0" Radius="20" Color="Blue"
    Opacity="0.5" Luminance="13"/>
</Circles>
```

Second, suppose that the configuration contains the namespaces for versions 1 and 2 but not the one for version 3. Then, since the Fallback element is selected, the output document contains the LuminanceFilter and Circle elements of the namespace for version 1 but does not contain Circle elements of that for version 3. The Opacity attributes are not removed, since the configuration contains the namespace for version 2.

Output document:

```
<Circles
  xmlns="http://www.example.com/Circles/v1"
  xmlns:v2="http://www.example.com/Circles/v2">
    <LuminanceFilter Luminance="13">
      <Circle Center="0,0" Radius="20" Color="Blue"
        v2:Opacity="0.5"/>
    </LuminanceFilter>
</Circles>
```

Third, suppose that the configuration contains the namespace for version 1 but not those for versions 2 or 3. Then, since the Fallback element is selected, the output document contains the LuminanceFilter element and Circle elements of the namespace for version 1 but does not contain Circle elements of that for version 3. Furthermore, since the configuration does not contain the namespace for version 2, the Opacity attributes are removed.

Output document:

```
<Circles
  xmlns="http://www.example.com/Circles/v1">
    <LuminanceFilter Luminance="13">
      <Circle Center="0,0" Radius="20" Color="Blue"/>
    </LuminanceFilter>
</Circles>
```

A.6 Example: Application-Defined Extension Elements

TBD

End of informative text.

Annex B.
(informative)
Validation Using NVDL

Comment [JH43]: This should be reviewed to ensure that the script is still valid
Owner: Murata-san

This annex is informative.

B.1 Introduction

Namespace-based Validation Dispatching Language (NVDL) allows documents to be decomposed into validation candidates, each of which can be validated independently.

NVDL can be used for validation against the normative requirements of this Part of ISO/IEC 29500. It can also be used for validation against the combination of Office Open XML documents (including the elements and attributes defined in this Part of ISO/IEC 29500) and any extensions.

B.2 Example of Validation Against Requirements of this Part of ISO/IEC 29500

A markup document can satisfy requirements of this ~~Annex~~ Part without being an Office Open XML document. The following NVDL script examines whether a given document correctly uses the attributes and elements as defined by this Part of ISO/IEC 29500.

This NVDL script first extracts elements and attributes in the Markup Compatibility namespace, and then validates them against the appropriate RELAX NG schemas.

Note that AlternateContent, Choice and Fallback elements are allowed to have foreign elements and attributes.

```
<?xml version="1.0" encoding="UTF-8"?>
<rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0">
  <namespace match="attributes" ns="http://schemas.openxmlformats.org/markup-
    compatibility/2006">
    <validate schemaType="application/relax-ng-compact-syntax">
      <schema>
        namespace mc="http://schemas.openxmlformats.org/markup-
         compatibility/2006"
        nsList = list { xsd:NCName* }
        qnameList = list { (xsd:QName | xsd:string {pattern = "\i\c*:\*" })*}
        start = element * {
          attribute mc:Ignorable { nsList }?,
```

```
          attribute mc:ProcessContent { qnameList }?,
          attribute mc:MustUnderstand { nsList }?
        }
      </schema>
    </validate>
  </namespace>
  <namespace match="elements" ns="http://schemas.openxmlformats.org/markup-
  compatibility/2006">
    <validate schemaType="application/relax-ng-compact-syntax">
      <schema>
        default namespace ="http://schemas.openxmlformats.org/markup-
          compatibility/2006"
        nsList = list { xsd:NCName* }
        qnameList = list { (xsd:QName | xsd:string {pattern = "\i\c*:\*" })*}
        start = element AlternateContent {choice+,fallback?}
        choice = element Choice {attribute Requires { nsList }, text}
        fallback = element Fallback {text}
      </schema>
    </validate>
  </namespace>
  <namespace ns="" match="attributes">
    <attach/>
  </namespace>
  <anyNamespace match="elements attributes">
    <allow/>
  </anyNamespace>
</rules>
```

The two RELAX NG schemas embedded in the above NVDL script can be rewritten in the analogous XML Schema form.

B.3 Example of Validation ~~Against~~ Using an NVDL Script~~the Combination of Office Open XML and Extensions~~

An extension of Office Open XML Transitional specified using the mechanisms defined in this Part of ISO/IEC 29500 can be captured by an NVDL script that invokes the Office Open XML Transitional schema and schemas for the extension. [*Note*: This NVDL script handles the conformance class "Transitional". A similar NVDL script for the conformance class "Strict" can be created by replacing each transitional namespace by a corresponding strict namespace and removing <namespace> elements for transitional-only features such as VML. *end note*]

The following schema allows two extensions. They have the namespaces http://www.example.com/myExtensionWithFallback and http://www.example.com/myExtensionWithoutFallback. The first extension is accompanied with a parent

AlternateContent element and a sibling Fallback element, while the second one can appear anywhere in the document without AlternateContent or Fallback elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0" startMode="top">
    <mode name="top">
        <namespace
          ns="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
            <validate schema="wml.xsd" useMode="nested"/>
        </namespace>
    </mode>
    <mode name="nested">
        <namespace match="attributes elements"
            ns="http://schemas.openxmlformats.org/drawingml/2006/*">
            <attach/>
        </namespace>
        <namespace match="attributes elements"
            ns="http://schemas.openxmlformats.org/officeDocument/2006/*">
            <attach/>
        </namespace>
        <namespace match="attributes elements"
          ns="http://schemas.openxmlformats.org/package/2006/*">
            <attach/>
        </namespace>
        <namespace match="attributes elements"
            ns="http://schemas.openxmlformats.org/presentationml/2006/main">
            <attach/>
        </namespace>
        <namespace match="attributes elements"
            ns="http://schemas.openxmlformats.org/schemaLibrary/2006/main">
            <attach/>
        </namespace>
        <namespace match="attributes elements"
            ns="http://schemas.openxmlformats.org/spreadsheetml/2006/7/main">
            <attach/>
        </namespace>
        <namespace match="attributes elements"
          ns="urn:schemas-microsoft-com:*">
            <attach/>
        </namespace>
        <namespace match="attributes"
            ns="http://schemas.openxmlformats.org/markup-compatibility/2006">
            <validate schemaType="application/relax-ng-compact-syntax">
                <schema>
```

```
            namespace mc =
            "http://schemas.openxmlformats.org/markup-
              compatibility/2006"
            nsList = list { xsd:NCName* }
            qnameList = list { (xsd:QName | xsd:string {pattern =
              "\i\c*:\*" })*}
            start = element * {
              attribute mc:Ignorable { nsList }?,
              attribute mc:ProcessContent { qnameList }?,
              attribute mc:MustUnderstand { nsList }?
            }
          </schema>
    </validate>
</namespace>
<namespace match="elements"
  ns="http://schemas.openxmlformats.org/markup-compatibility/2006">
    <validate schemaType="application/relax-ng-compact-syntax">
        <schema>
            default namespace =
            "http://schemas.openxmlformats.org/markup-
              compatibility/2006"
            nsList = list { xsd:NCName* }
            qnameList = list { (xsd:QName | xsd:string {pattern =
              "\i\c*:\*" })*}
            start = element AlternateContent {choice,fallback}
            choice = element Choice {attribute Requires { nsList },
              text}
            fallback = element Fallback {text}
        </schema>
        <mode>
            <anyNamespace>
                <allow/>
            </anyNamespace>
        </mode>
        <context path="Choice">
            <mode>
                <namespace
                  ns="http://www.example.com/myExtenstionWithFallback">
                    <validate schema="myExtensionWithFallback.rng">
                        <mode>
                            <anyNamespace>
                                <attach/>
                            </anyNamespace>
```

```
                                </mode>
                            </validate>
                        </namespace>
                    </mode>
                </context>
            </validate>
            <unwrap>
                <mode>
                    <anyNamespace>
                        <allow/>
                    </anyNamespace>
                </mode>
                <context path="Fallback">
                    <mode>
                        <anyNamespace>
                            <attach/>
                        </anyNamespace>
                    </mode>
                </context>
            </unwrap>
        </namespace>
        <namespace ns="http://www.example.com/myExtensionWithoutFallback">
            <validate schema="myExtensionWithoutFallback.rng">
                <mode>
                    <anyNamespace>
                        <attach/>
                    </anyNamespace>
                </mode>
            </validate>
        </namespace>
    </mode>
</rules>
```

End of informative text.