17.5 Custom Markup

Within a WordprocessingML document, it is often necessary for specific documents to contain semantic information beyond the presentation information specified by ECMA-376. [*Example*: An invoice document might wish to specify that a particular sentence of text is a customer name, in order for that information to be easily extracted from the document without the need to parse the text using regular expression matching or similar. *end example*]

For these scenarios, multiple facilities are provided for the insertion and round-tripping of ~~customer defined~~extra-standard semantics within a WordprocessingML document. There are three distinct forms in which ~~customer-defined~~extra-standard semantics can be inserted into a WordprocessingML document, each with their own specific intended usage:

- Smart tags
- Custom XML markup
- Structured document tags (content controls)

The elements and attributes which define each of these forms is described in the following clauses.

17.5.1 Custom XML and Smart Tags

The first ~~example~~ form of ~~customer-defined~~extra-standard semantics that can be embedded in a WordprocessingML document are smart tags. ~~Customers~~ Implementations can establish sets of smart tags that allow semantic labels to be added around an arbitrary run or set of runs within a document to provide information about the type of data contained within.

[*Example*: Consider the following text in a WordprocessingML document, with a smart tag around the stock
symbol 'CNTS':
This is a stock symbol: CNTS
This text would translate to the following WordprocessingML markup:

```
<w:p w:rsidR="00672474" w:rsidRDefault="00672474">
<w:r>
<w:t xml:space="preserve">This is a stock symbol: </w:t>
</w:r>
<w:smartTag w:uri="http://www.example.com"
w:element="stockticker">
<w:r>
<w:t>CNTS</w:t>
</w:r>
</w:smartTag>
</w:p>
```

As shown above, the smart tag is delimited by the smartTag element, which surrounds the run (or runs) which contain the text which is part of the smart tag. *end example*]

The smart tag ~~itself~~ element in a document~~carries~~ has two required ~~pieces of information~~attributes~~, which together contain the customer semantics for this smart tag~~:

- The first of these is the namespace for this smart tag (contained in the uri attribute). ~~This allows the smart tag to specify a URI which should identifies the namespace of this smart tag to a consumer. It is intended to be used to specify a family of smart tags to which this one belongs.~~ [*Example*: In the sample above, the smart tag belongs to the `http://www.example.com` namespace. *end example*]
- The second of these is the ~~element~~ classification ~~name~~ for this smart tag (contained in the element attribute). This attribute, in combination with the namespace uri, specifies a classification which uniquely identifies this smart tag within its family and again available to a consumer~~This allows the smart tag to specify a name which identifies this type of smart tag~~

~~within its namespace and again available to a consumer. It is intended to be used to specify a unique name for this type of smart tag.~~ [*Example*: In the sample above, the smart tag specifies that its data is ~~of~~ classified as a~~style~~ `stockticker`. *End example*]

The next example of ~~customer-defined~~ extra-standard semantics which can be embedded in a WordprocessingML document is custom XML markup. Custom XML markup allows the application of the XML elements defined in any schema syntax (XML Schema, NVDL, etc.) to be applied to the contents of a WordprocessingML document in one of two locations: around a paragraph or set of paragraphs (at the block level); or around an arbitrary run or set of runs within a document (at the inline level) to provide semantics to that content within the context and structures defined by the associated schema definition.

The distinction between custom XML markup and smart tags is that custom XML markup is based on a ~~specified~~ schema~~.~~, which may be specified using the attachedSchema element (§17.15.1.5). As a result, the custom XML elements can be validated against the schema. Also, as shown below, custom XML markup can be used at the block-level as well as on the inline (run) level.

[*Example*: Consider a simple XML Schema which defines two elements: a root element of invoice, and a child element of customerName - the first defining that this file's contents are an invoice, and the second specifying that the enclosed text as a customer's name:

This output would translate to the following WordprocessingML markup:

```
<w:customXml w:uri="http://www.example.com/2006/invoice" w:element="invoice">
<w:p>
<w:r>
<w:t>This is an invoice.</w:t>
</w:r>
</w:p>
<w:p>
<w:r>
<w:t xml:space="preserve">And this is a customer name: </w:t>
</w:r>
<w:customXml w:uri="http://www.example.com/2006/invoice"
w:element="customerName">
<w:r>
<w:t>Tristan Davis</w:t>
</w:r>
</w:customXml>
</w:p>
</w:customXml>
```

As shown above, each of the XML elements from the ~~customer-~~supplied XML schema is represented within the document output as a customXml element. *end example*]

Similar to the smart tag example above, a custom XML element in a document has two required attributes.

- The first is the uri attribute, whose contents specify the namespace of the custom XML element in the document. In the example above, the elements each belong to the `http://www.example.com/2006/invoice` namespace.
- The second is the element attribute, whose contents specify the name of the custom XML element at this location in the document. In the example above, the root element is called invoice and the child element is called customerName.

As well as the required information specified above, custom XML elements can also specify any number of attributes (as specified in the associated XML Schema) on the element. To add this information, the

customXmlPr (properties on the custom XML element) specify one or more attr elements. [*Example*: Using the example above, we can add a type attribute to the customerName element as follows:

```
<w:customXml w:uri="http://www.example.com/2006/invoice"
w:element="customerName">
<w:customXmlPr>
<w:attr w:uri="http://www.example.com/2006/invoice" w:name="type"
w:val="individual"/>
</w:customXmlPr>
<w:r>
<w:t>Tristan Davis</w:t>
</w:r>
</w:customXml>
```

The resulting XML, as seen above, simply adds an attr element which specifies the attribute for the custom XML element. *end example*].~~...~~

It is a consumer defined operation to extract XML documents from custom XML markup. The result of extracting XML from custom XML markup shall always return well-formed XML. In addition, custom XML markup with Open XML, if added according to the specification, shall never create a non-conformant Open XML document. However, the result of extracting XML from custom XML markup may return an invalid document according to the custom XML schema.

[*Guidance*: The uri attribute on custom XML elements or attributes can be used to specify XML prefixes when extracting XML documents from custom XML markup. If the uri attribute is specified for custom XML elements or attributes then a consumer can choose to add a prefix to the output XML elements or attributes based on the value of the uri attribute. If the uri attribute is omitted for custom XML elements or attributes then a consumer can choose to not write a prefix to the output XML elements or attributes. *End guidance*]

Part 1, §17.5.2, "Structured Document Tags", p. 549, will be updated as follows:

The final form of ~~customer defined~~ extra-standard semantics which can be embedded in a WordprocessingML document are structured document tags (SDTs).

As shown above, smart tags and custom XML markup each provide a facility for embedding ~~customer defined~~extra-standard semantics into the document: smart tags, via the ability to provide a basic namespace/name for a run or set of runs within a documents; and custom XML markup, via the ability to tag the document with XML elements and attributes specified by any XML Schema file.

However, each of these techniques, while they each provide a way to add the desired semantic information, does not provide a way to affect the presentation or interaction within the document. To bridge these two worlds, structured document tags allow both the specification of extra-standard~~customer~~ semantics as well as the ability to influence the presentation of that data in the document.

This means that the implementation~~customer~~ can define the semantics and context of the tag, but can then use a rich set of pre-defined properties to define its behavior and appearance within the WordprocessingML document's presentation.

Part 1, §M.1.6, "Custom Markup", p. 5087, will be updated as follows:

~~Within a WordprocessingML document, it is often necessary for specific documents to contain semantic information beyond the presentation information specified by this Office Open XML specification. For example, an invoice document might wish to specify that a particular sentence of text is a customer name, in order for that information to be easily extracted from the document without the need to parse the text using regular expression matching or similar. For those cases, multiple facilities are provided for the insertion and round-tripping of customer defined semantics within a WordprocessingML document.~~

There are three distinct forms in which ~~customer defined~~ extra-standard semantics can be inserted into a WordprocessingML document, each with their own specific intended usage: …

Part 1, §M.1.6.1, "Smart Tags", p. 5087, will be updated as follows:

The first form~~example~~ of ~~customer defined~~ extra-standard semantics ~~which~~that can be embedded in a WordprocessingML document are smart tags. Smart tags allow semantic information to be added around an arbitrary run or set of runs within a document to provide information about the kind of data contained within.

Part 1, §M.1.6.2, "Custom XML Markup", p. 5089, will be updated as follows:

The next form ~~example~~ of ~~customer defined~~ extra-standard semantics which can be embedded in a WordprocessingML document is custom XML markup. …

…

A producer can embed a custom XML element around or with block-level or run-level content in a WordprocessingML document in order to embed the structure of the ~~customer defined~~ extra-standard XML Schema within the WordprocessingML content. This allows 'tagging' of specific regions of a document with the semantics from this schema, while ensuring that the resulting file can be validated to the WordprocessingML schemas.

A consumer can read this custom XML markup and provide additional functionality around this ~~customer defined~~ extra-standard XML markup, which might or might not be specific to that particular XML namespace. Examples of this functionality include: the ability to add/remove this XML markup via a user interface, ability to provide actions to operating in the context of this namespace, etc.

Part 1, §M.1.6.3, "Structured Document Tags", p. 5091, will be updated as follows:

The final form~~example~~ of ~~customer defined~~ extra-standard semantics which can be embedded in a WordprocessingML document is the structured document tag (SDT).

Part 1, §M.3.1.2.8, "Customer Data", p. 5304, will be updated as follows:

There is a set of utilities that facilitate the storage of customer XML data within the file format. Although a topic for a separate paper, essentially, this functionality comes down to the ability

to store ~~customer defined~~ extra-standard XML in the file format in a way that it can be easily queried, modified and/or surfaced in the presentation. Suffice it to say, the data is stored in a separate part within the package, and hence the utility pairs the object using it with the part within the package.