16 Output

```
<!-- Category: top-level-element -->
<xsl:output
  method = "xml" | "html" | "text" | qname-but-not-ncname
  version = nmtoken
  encoding = string
  omit-xml-declaration = "yes" | "no"
  standalone = "yes" | "no"
  doctype-public = string
  doctype-system = string
  cdata-section-elements = qnames
  indent = "yes" | "no"
  media-type = string />
```

An XSLT processor may output the result tree as a sequence of bytes, although it is not required to be able to do so (see [**17 Conformance**]). The `xsl:output` element allows stylesheet authors to specify how they wish the result tree to be output. If an XSLT processor outputs the result tree, it should do so as specified by the `xsl:output` element; however, it is not required to do so.

The `xsl:output` element is only allowed as a top-level element.

The `method` attribute on `xsl:output` identifies the overall method that should be used for outputting the result tree. The value must be a QName. If the QName does not have a prefix, then it identifies a method specified in this document and must be one of `xml`, `html` or `text`. If the QName has a prefix, then the QName is expanded into an expanded-name as described in [**2.4 Qualified Names**]; the expanded-name identifies the output method; the behavior in this case is not specified by this document.

The default for the `method` attribute is chosen as follows. If

- the root node of the result tree has an element child,
- the expanded-name of the first element child of the root node (i.e. the document element) of the result tree has local part `html` (in any combination of upper and lower case) and a null namespace URI, and
- any text nodes preceding the first element child of the root node of the result tree contain only whitespace characters,

then the default output method is `html`; otherwise, the default output method is `xml`. The default output method should be used if there are no `xsl:output` elements or if none of the `xsl:output` elements specifies a value for the `method` attribute.

The other attributes on `xsl:output` provide parameters for the output method. The following attributes are allowed:

- `version` specifies the version of the output method
- `indent` specifies whether the XSLT processor may add additional whitespace when outputting the result tree; the value must be `yes` or `no`
- `encoding` specifies the preferred character encoding that the XSLT processor should use to encode sequences of characters as sequences of bytes; the value of the attribute should be treated case-insensitively; the value must contain only characters in the range #x21 to #x7E (i.e. printable ASCII characters); the value should either be a `charset` registered with the Internet Assigned Numbers Authority [IANA], [RFC2278] or start with `X-`
- `media-type` specifies the media type (MIME content type) of the data that results from

outputting the result tree; the `charset` parameter should not be specified explicitly; instead, when the top-level media type is `text`, a `charset` parameter should be added according to the character encoding actually used by the output method

- `doctype-system` specifies the system identifier to be used in the document type declaration
- `doctype-public` specifies the public identifier to be used in the document type declaration
- `omit-xml-declaration` specifies whether the XSLT processor should output an XML declaration; the value must be `yes` or `no`
- `standalone` specifies whether the XSLT processor should output a standalone document declaration; the value must be `yes` or `no`
- `cdata-section-elements` specifies a list of the names of elements whose text node children should be output using CDATA sections

The detailed semantics of each attribute will be described separately for each output method for which it is applicable. If the semantics of an attribute are not described for an output method, then it is not applicable to that output method.

A stylesheet may contain multiple `xsl:output` elements and may include or import stylesheets that also contain `xsl:output` elements. All the `xsl:output` elements occurring in a stylesheet are merged into a single effective `xsl:output` element. For the `cdata-section-elements` attribute, the effective value is the union of the specified values. For other attributes, the effective value is the specified value with the highest import precedence. It is an error if there is more than one such value for an attribute. An XSLT processor may signal the error; if it does not signal the error, if should recover by using the value that occurs last in the stylesheet. The values of attributes are defaulted after the `xsl:output` elements have been merged; different output methods may have different default values for an attribute.

16.1 XML Output Method

The `xml` output method outputs the result tree as a well-formed XML external general parsed entity. If the root node of the result tree has a single element node child and no text node children, then the entity should also be a well-formed XML document entity. When the entity is referenced within a trivial XML document wrapper like this

```
<!DOCTYPE doc [
<!ENTITY e SYSTEM "entity-URI">
]>
<doc>&e;</doc>
```

where *`entity-URI`* is a URI for the entity, then the wrapper document as a whole should be a well-formed XML document conforming to the XML Namespaces Recommendation [XML Names]. In addition, the output should be such that if a new tree was constructed by parsing the wrapper as an XML document as specified in [**3 Data Model**], and then removing the document element, making its children instead be children of the root node, then the new tree would be the same as the result tree, with the following possible exceptions:

- The order of attributes in the two trees may be different.
- The new tree may contain namespace nodes that were not present in the result tree.

> **NOTE:**An XSLT processor may need to add namespace declarations in the course of outputting the result tree as XML.

If the XSLT processor generated a document type declaration because of the `doctype-system`

attribute, then the above requirements apply to the entity with the generated document type declaration removed.

The `version` attribute specifies the version of XML to be used for outputting the result tree. If the XSLT processor does not support this version of XML, it should use a version of XML that it does support. The version output in the XML declaration (if an XML declaration is output) should correspond to the version of XML that the processor used for outputting the result tree. The value of the `version` attribute should match the VersionNum production of the XML Recommendation [XML]. The default value is `1.0`.

The `encoding` attribute specifies the preferred encoding to use for outputting the result tree. XSLT processors are required to respect values of `UTF-8` and `UTF-16`. For other values, if the XSLT processor does not support the specified encoding it may signal an error; if it does not signal an error it should use `UTF-8` or `UTF-16` instead. The XSLT processor must not use an encoding whose name does not match the EncName production of the XML Recommendation [XML]. If no `encoding` attribute is specified, then the XSLT processor should use either `UTF-8` or `UTF-16`. It is possible that the result tree will contain a character that cannot be represented in the encoding that the XSLT processor is using for output. In this case, if the character occurs in a context where XML recognizes character references (i.e. in the value of an attribute node or text node), then the character should be output as a character reference; otherwise (for example if the character occurs in the name of an element) the XSLT processor should signal an error.

If the `indent` attribute has the value `yes`, then the `xml` output method may output whitespace in addition to the whitespace in the result tree (possibly based on whitespace stripped from either the source document or the stylesheet) in order to indent the result nicely; if the `indent` attribute has the value `no`, it should not output any additional whitespace. The default value is `no`. The `xml` output method should use an algorithm to output additional whitespace that ensures that the result if whitespace were to be stripped from the output using the process described in **[3.4 Whitespace Stripping]** with the set of whitespace-preserving elements consisting of just `xsl:text` would be the same when additional whitespace is output as when additional whitespace is not output.

> **NOTE:**It is usually not safe to use `indent="yes"` with document types that include element types with mixed content.

The `cdata-section-elements` attribute contains a whitespace-separated list of QNames. Each QName is expanded into an expanded-name using the namespace declarations in effect on the `xsl:output` element in which the QName occurs; if there is a default namespace, it is used for QNames that do not have a prefix. The expansion is performed before the merging of multiple `xsl:output` elements into a single effective `xsl:output` element. If the expanded-name of the parent of a text node is a member of the list, then the text node should be output as a CDATA section. For example,

```
<xsl:output cdata-section-elements="example"/>
```

would cause a literal result element written in the stylesheet as

```
<example>&lt;foo></example>
```

or as

```
<example><![CDATA[<foo>]]></example>
```

to be output as

```
<example><![CDATA[<foo>]]></example>
```

If the text node contains the sequence of characters `]]>`, then the currently open CDATA section

should be closed following the]] and a new CDATA section opened before the >. For example, a literal result element written in the stylesheet as

```
<example>]]&gt;</example>
```

would be output as

```
<example><![CDATA[]]]><![CDATA[>]]></example>
```

If the text node contains a character that is not representable in the character encoding being used to output the result tree, then the currently open CDATA section should be closed before the character, the character should be output using a character reference or entity reference, and a new CDATA section should be opened for any further characters in the text node.

CDATA sections should not be used except for text nodes that the cdata-section-elements attribute explicitly specifies should be output using CDATA sections.

The xml output method should output an XML declaration unless the omit-xml-declaration attribute has the value yes. The XML declaration should include both version information and an encoding declaration. If the standalone attribute is specified, it should include a standalone document declaration with the same value as the value as the value of the standalone attribute. Otherwise, it should not include a standalone document declaration; this ensures that it is both a XML declaration (allowed at the beginning of a document entity) and a text declaration (allowed at the beginning of an external general parsed entity).

If the doctype-system attribute is specified, the xml output method should output a document type declaration immediately before the first element. The name following <!DOCTYPE should be the name of the first element. If doctype-public attribute is also specified, then the xml output method should output PUBLIC followed by the public identifier and then the system identifier; otherwise, it should output SYSTEM followed by the system identifier. The internal subset should be empty. The doctype-public attribute should be ignored unless the doctype-system attribute is specified.

The media-type attribute is applicable for the xml output method. The default value for the media-type attribute is text/xml.

16.2 HTML Output Method

The html output method outputs the result tree as HTML; for example,

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="html"/>

<xsl:template match="/">
  <html>
   <xsl:apply-templates/>
  </html>
</xsl:template>

...

</xsl:stylesheet>
```

The version attribute indicates the version of the HTML. The default value is 4.0, which specifies that the result should be output as HTML conforming to the HTML 4.0 Recommendation [HTML].

The html output method should not output an element differently from the xml output method unless the expanded-name of the element has a null namespace URI; an element whose

expanded-name has a non-null namespace URI should be output as XML. If the expanded-name of the element has a null namespace URI, but the local part of the expanded-name is not recognized as the name of an HTML element, the element should output in the same way as a non-empty, inline element such as `span`.

The `html` output method should not output an end-tag for empty elements. For HTML 4.0, the empty elements are `area`, `base`, `basefont`, `br`, `col`, `frame`, `hr`, `img`, `input`, `isindex`, `link`, `meta` and `param`. For example, an element written as `<br/>` or `<br></br>` in the stylesheet should be output as `<br>`.

The `html` output method should recognize the names of HTML elements regardless of case. For example, elements named `br`, `BR` or `Br` should all be recognized as the HTML `br` element and output without an end-tag.

The `html` output method should not perform escaping for the content of the `script` and `style` elements. For example, a literal result element written in the stylesheet as

```
<script>if (a &lt; b) foo()</script>
```

or

```
<script><![CDATA[if (a < b) foo()]]></script>
```

should be output as

```
<script>if (a < b) foo()</script>
```

The `html` output method should not escape `<` characters occurring in attribute values.

If the `indent` attribute has the value `yes`, then the `html` output method may add or remove whitespace as it outputs the result tree, so long as it does not change how an HTML user agent would render the output. The default value is `yes`.

The `html` output method should escape non-ASCII characters in URI attribute values using the method recommended in Section B.2.1 of the HTML 4.0 Recommendation.

The `html` output method may output a character using a character entity reference, if one is defined for it in the version of HTML that the output method is using.

The `html` output method should terminate processing instructions with `>` rather than `?>`.

The `html` output method should output boolean attributes (that is attributes with only a single allowed value that is equal to the name of the attribute) in minimized form. For example, a start-tag written in the stylesheet as

```
<OPTION selected="selected">
```

should be output as

```
<OPTION selected>
```

The `html` output method should not escape a `&` character occurring in an attribute value immediately followed by a `{` character (see Section B.7.1 of the HTML 4.0 Recommendation). For example, a start-tag written in the stylesheet as

```
<BODY bgcolor='&amp;{{randomrbg}};'>
```

should be output as

```
<BODY bgcolor='&{randomrbg};'>
```

The `encoding` attribute specifies the preferred encoding to be used. If there is a `HEAD` element, then the `html` output method should add a `META` element immediately after the start-tag of the `HEAD` element specifying the character encoding actually used. For example,

```
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=EUC-JP">
...
```

It is possible that the result tree will contain a character that cannot be represented in the encoding that the XSLT processor is using for output. In this case, if the character occurs in a context where HTML recognizes character references, then the character should be output as a character entity reference or decimal numeric character reference; otherwise (for example, in a `script` or `style` element or in a comment), the XSLT processor should signal an error.

If the `doctype-public` or `doctype-system` attributes are specified, then the `html` output method should output a document type declaration immediately before the first element. The name following `<!DOCTYPE` should be `HTML` or `html`. If the `doctype-public` attribute is specified, then the output method should output `PUBLIC` followed by the specified public identifier; if the `doctype-system` attribute is also specified, it should also output the specified system identifier following the public identifier. If the `doctype-system` attribute is specified but the `doctype-public` attribute is not specified, then the output method should output `SYSTEM` followed by the specified system identifier.

The `media-type` attribute is applicable for the `html` output method. The default value is `text/html`.

16.3 Text Output Method

The `text` output method outputs the result tree by outputting the string-value of every text node in the result tree in document order without any escaping.

The `media-type` attribute is applicable for the `text` output method. The default value for the `media-type` attribute is `text/plain`.

The `encoding` attribute identifies the encoding that the `text` output method should use to convert sequences of characters to sequences of bytes. The default is system-dependent. If the result tree contains a character that cannot be represented in the encoding that the XSLT processor is using for output, the XSLT processor should signal an error.

16.4 Disabling Output Escaping

Normally, the `xml` output method escapes & and < (and possibly other characters) when outputting text nodes. This ensures that the output is well-formed XML. However, it is sometimes convenient to be able to produce output that is almost, but not quite well-formed XML; for example, the output may include ill-formed sections which are intended to be transformed into well-formed XML by a subsequent non-XML aware process. For this reason, XSLT provides a mechanism for disabling output escaping. An `xsl:value-of` or `xsl:text` element may have a `disable-output-escaping` attribute; the allowed values are `yes` or `no`; the default is `no`; if the value is `yes`, then a text node generated by instantiating the `xsl:value-of` or `xsl:text` element should be output without any escaping. For example,

```
<xsl:text disable-output-escaping="yes">&lt;</xsl:text>
```

should generate the single character <.

It is an error for output escaping to be disabled for a text node that is used for something other than a text node in the result tree. Thus, it is an error to disable output escaping for an

than a text node in the result tree. Thus, it is an error to disable output escaping for an `xsl:value-of` or `xsl:text` element that is used to generate the string-value of a comment, processing instruction or attribute node; it is also an error to convert a result tree fragment to a number or a string if the result tree fragment contains a text node for which escaping was disabled. In both cases, an XSLT processor may signal the error; if it does not signal the error, it must recover by ignoring the `disable-output-escaping` attribute.

The `disable-output-escaping` attribute may be used with the `html` output method as well as with the `xml` output method. The `text` output method ignores the `disable-output-escaping` attribute, since it does not perform any output escaping.

An XSLT processor will only be able to disable output escaping if it controls how the result tree is output. This may not always be the case. For example, the result tree may be used as the source tree for another XSLT transformation instead of being output. An XSLT processor is not required to support disabling output escaping. If an `xsl:value-of` or `xsl:text` specifies that output escaping should be disabled and the XSLT processor does not support this, the XSLT processor may signal an error; if it does not signal an error, it must recover by not disabling output escaping.

If output escaping is disabled for a character that is not representable in the encoding that the XSLT processor is using for output, then the XSLT processor may signal an error; if it does not signal an error, it must recover by not disabling output escaping.

Since disabling output escaping may not work with all XSLT processors and can result in XML that is not well-formed, it should be used only when there is no alternative.