RXSLT Examples

Sam Wilmott
sam@wilmott.ca
www.wilmott.ca

August 3, 2006

The following examples are all the examples in the XSLT 1.0 specification, each followed by the corresponding RXSLT program or program fragment, and by the output of the RXSLT-to-XSLT translator in three forms: preserving all space (non-indented), wrapping all text in <xsl:text> elements (indented), and wrapping in <xsl:text> only when necessary (better indented). (Where the latter two give the same output, which is often the case, the "better indented" output is omitted.) The examples are as in the XSLT 1.0 spec except where some small modification has to be made to make it complete.

This file was produced automatically by running it through the RXSLT compiler. As a result, all XSLT 1.0 outputs are encoded as full stylesheets even where the XSLT 1.0 spec example is just a code fragment.

2.3 Literal Result Element as Stylesheet

XML-encoded XSLT:

```
<html xsl:version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns="http://www.w3.org/TR/xhtml1/strict">
    <head>
      <title>Expense Report Summary</title>
    </head>
    <body>
      <p>Total Amount: <xsl:value-of select="expense-report/total"/></p>
    </body>
</html>
```

RXSLT:

```
<html xmlns="http://www.w3.org/TR/xhtml1/strict">
  <head>
    <title>Expense Report Summary</title>
  </head>
  <body>
    <p>Total Amount: {value-of expense-report/total}</p>
  </body>
</html>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
  <head>
    <title>Expense Report Summary</title>
  </head>
  <body>
    <p>Total Amount: <xsl:value-of select="expense-report/total"/></p>
  </body>
</html></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html xmlns="http://www.w3.org/TR/xhtml1/strict">
  <head>
    <title>Expense Report Summary</title>
  </head>
  <body>
    <p>Total Amount: <xsl:value-of select="expense-report/total"/></p>
  </body>
</html>
  </xsl:template>
</xsl:stylesheet>
```

2.5 Forwards-Compatible Processsing

This is not really supported by other than by XML-encoding the non-XSLT 1.0 feature.
XML-encoded XSLT:

```
<xsl:stylesheet version="1.1"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:choose>
      <xsl:when test='system-property("version") >= 1.1'>
        <xsl:exciting-new-1.1-feature/>
      </xsl:when>
      <xsl:otherwise>
        <html>
          <head>
            <title>XSLT 1.1 required</title>
          </head>
          <body>
            <p>Sorry, this stylesheet requires XSLT 1.1.</p>
          </body>
        </html>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

RXSLT:

```
xslt1.1
template /
  choose
    when system-property("version") >= 1.1
      <xsl:exciting-new-1.1-feature/>
    otherwise
      <html>
        <head>
          <title>XSLT 1.1 required</title>
        </head>
        <body>
          <p>Sorry, this stylesheet requires XSLT 1.1.</p>
        </body>
      </html>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.1" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
        <head>
          <title>XSLT 1.1 required</title>
        </head>
        <body>
          <p>Sorry, this stylesheet requires XSLT 1.1.</p>
        </body>
      </html></xsl:otherwise></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.1" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/choose">
    <xsl:when test='system-property("version")>=1.1'>
      <xsl:exciting-new-1.1-feature/>
    </xsl:when>
    <xsl:otherwise>
      <html>
        <head>
          <title>XSLT 1.1 required</title>
        </head>
        <body>
          <p>Sorry, this stylesheet requires XSLT 1.1.</p>
        </body>
      </html>
    </xsl:otherwise>
  </xsl:template>
</xsl:stylesheet>
```

2.6.2 Stylesheet Import

XML-encoded XSLT:

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="article.xsl"/>
  <xsl:import href="bigfont.xsl"/>
  <xsl:attribute-set name="note-style">
    <xsl:attribute name="font-style">italic</xsl:attribute>
  </xsl:attribute-set>
</xsl:stylesheet>
```

RXSLT:

```
import article.xsl
import bigfont.xsl
attribute-set note-style
  attribute font-style : "italic"
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="article.xsl"/>
  <xsl:import href="bigfont.xsl"/>
  <xsl:attribute-set name="note-style">
    <xsl:attribute name="font-style">
```

```
      <xsl:text>italic</xsl:text>
    </xsl:attribute>
  </xsl:attribute-set>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="article.xsl"/>
  <xsl:import href="bigfont.xsl"/>
  <xsl:attribute-set name="note-style">
    <xsl:attribute name="font-style">italic</xsl:attribute>
  </xsl:attribute-set>
</xsl:stylesheet>
```

2.7 Embedding Stylesheets

This feature is currently not supported, but could be supported by wrapping a whole RXSLT program in "style" tags in the manner that Javascript programs are embedded in HTML documents.

5.3 Defining Template Rules

XML-encoded XSLT:

```
<xsl:template match="emph">
  <fo:inline-sequence font-weight="bold">
    <xsl:apply-templates/>
  </fo:inline-sequence>
</xsl:template>
```

RXSLT:

```
template emph
  <fo:inline-sequence font-weight="bold">
    {apply-templates}
  </fo:inline-sequence>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:apply-templates/>
  </fo:inline-sequence></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="emph">
    <fo:inline-sequence font-weight="bold">
    <xsl:apply-templates/>
  </fo:inline-sequence>
  </xsl:template>
</xsl:stylesheet>
```

5.4 Applying Template Rules (first example)

XML-encoded XSLT:

```
<xsl:template match="chapter">
  <fo:block>
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

RXSLT:

```
template chapter
  <fo:block>
    {apply-templates}
  </fo:block>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:apply-templates/>
  </fo:block></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="chapter">
    <fo:block>
    <xsl:apply-templates/>
  </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

5.4 Applying Template Rules (second example)

XML-encoded XSLT:

```
<xsl:template match="author-group">
  <fo:inline-sequence>
    <xsl:apply-templates select="author"/>
  </fo:inline-sequence>
</xsl:template>
```

RXSLT:

```
template author-group
  <fo:inline-sequence>
    {apply-templates with-nodes author}
  </fo:inline-sequence>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:apply-templates select="author"/>
  </fo:inline-sequence></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="author-group">
    <fo:inline-sequence>
    <xsl:apply-templates select="author"/>
  </fo:inline-sequence>
  </xsl:template>
</xsl:stylesheet>
```

5.4 Applying Template Rules (third example)

XML-encoded XSLT:

```
<xsl:template match="author-group">
  <fo:inline-sequence>
    <xsl:apply-templates select="author/given-name"/>
  </fo:inline-sequence>
</xsl:template>
```

RXSLT:

```
template author-group
  <fo:inline-sequence>
    {apply-templates with-nodes author/given-name}
  </fo:inline-sequence>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:apply-templates select="author/given-name"/>
  </fo:inline-sequence></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="author-group">
    <fo:inline-sequence>
    <xsl:apply-templates select="author/given-name"/>
  </fo:inline-sequence>
  </xsl:template>
</xsl:stylesheet>
```

5.4 Applying Template Rules (fourth example)

XML-encoded XSLT:

```
<xsl:template match="book">
  <fo:block>
    <xsl:apply-templates select=".//heading"/>
  </fo:block>
</xsl:template>
```

RXSLT:

```
template book
  <fo:block>
    {apply-templates with-nodes .//heading}
  </fo:block>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:apply-templates select=".//heading"/>
  </fo:block></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="book">
    <fo:block>
    <xsl:apply-templates select=".//heading"/>
  </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

5.4 Applying Template Rules (fifth example)

XML-encoded XSLT:

```
<xsl:template match="employee">
  <fo:block>
    Employee <xsl:apply-templates select="name"/> belongs to group
    <xsl:apply-templates select="ancestor::department/group"/>
  </fo:block>
</xsl:template>
```

RXSLT:

```
template employee
  <fo:block>
    Employee {apply-templates with-nodes name} belongs to group
    {apply-templates with-nodes ancestor::department/group}
  </fo:block>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    Employee <xsl:apply-templates select="name"/> belongs to group
    <xsl:apply-templates select="ancestor::department/group"/>
  </fo:block></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="employee">
    <fo:block>
    Employee <xsl:apply-templates select="name"/> belongs to group
    <xsl:apply-templates select="ancestor::department/group"/>
  </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

5.4 Applying Template Rules (sixth example)

XML-encoded XSLT:

```
<xsl:template match="product">
  <table>
    <xsl:apply-templates select="sales/domestic"/>
  </table>
  <table>
    <xsl:apply-templates select="sales/foreign"/>
  </table>
</xsl:template>
```

RXSLT:

```
template product
  <table>
    {apply-templates with-nodes sales/domestic}
  </table>
  <table>
    {apply-templates with-nodes sales/foreign}
  </table>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:apply-templates select="sales/domestic"/>
  </table><table>
    <xsl:apply-templates select="sales/foreign"/>
  </table></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="product">
    <table>
    <xsl:apply-templates select="sales/domestic"/>
  </table>
    <table>
    <xsl:apply-templates select="sales/foreign"/>
  </table>
  </xsl:template>
</xsl:stylesheet>
```

5.4 Applying Template Rules (seventh example)

XML-encoded XSLT:

```
<xsl:template match="doc">
  <xsl:apply-templates select=".//div"/>
</xsl:template>
```

RXSLT:

```
template doc
  apply-templates with-nodes .//div
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="doc">
    <xsl:apply-templates select=".//div"/>
  </xsl:template>
</xsl:stylesheet>
```

5.4 Applying Template Rules (eighth example – the invalid one)

XML-encoded XSLT:

```
<xsl:template match="foo">
  <xsl:apply-templates select="."/>
</xsl:template>
```

RXSLT:

```
template foo
  apply-templates with-nodes .
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="foo">
    <xsl:apply-templates select="."/>
  </xsl:template>
</xsl:stylesheet>
```

5.6 Overriding Template Rules

file "doc.xsl":
XML-encoded XSLT:

```
<xsl:template match="example">
  <pre><xsl:apply-templates/></pre>
</xsl:template>
```

RXSLT:

```
template example
  <pre>{apply-templates}</pre>
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="example">
    <pre><xsl:apply-templates/></pre>
  </xsl:template>
</xsl:stylesheet>
```

importing stylesheet:
XML-encoded XSLT:

```
<xsl:import href="doc.xsl"/>
<xsl:template match="example">
  <div style="border: solid red">
     <xsl:apply-imports/>
  </div>
</xsl:template>
```

RXSLT:

```
import "doc.xsl"
template example
  <div style="border: solid red">
     {apply-imports}
  </div>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
     <xsl:apply-imports/>
  </div></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="doc.xsl"/>
  <xsl:template match="example">
    <div style="border: solid red">
     <xsl:apply-imports/>
  </div>
  </xsl:template>
</xsl:stylesheet>
```

5.8 Built-in Template Rules (element and root nodes)

XML-encoded XSLT:

```
<xsl:template match="*|/">
  <xsl:apply-templates/>
</xsl:template>
```

RXSLT:

```
template * | /
  apply-templates
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="*|/apply-templates"/>
</xsl:stylesheet>
```

5.8 Built-in Template Rules (for each mode)

XML-encoded XSLT:

```
<xsl:template match="*|/" mode="m">
  <xsl:apply-templates/>
</xsl:template>
```

RXSLT:

```
mode m
template * | /
  apply-templates
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="*|/apply-templates" mode="m"/>
</xsl:stylesheet>
```

5.8 Built-in Template Rules (text and attribute nodes)

XML-encoded XSLT:

```
<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

RXSLT:

```
template text() | @*
  value-of .
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="text()|@*">
    <xsl:value-of select="."/>
  </xsl:template>
</xsl:stylesheet>
```

5.8 Built-in Template Rules (processing instructions and comments)

XML-encoded XSLT:

```
<xsl:template match="processing-instruction() | comment()"/>
```

RXSLT:

```
template processing-instruction() | comment()
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="processing-instruction()|comment()"/>
</xsl:stylesheet>
```

7.1.1 Literal Result Elements

XML-encoded XSLT:

```
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:axsl="http://www.w3.org/1999/XSL/TransformAlias">
<xsl:namespace-alias stylesheet-prefix="axsl" result-prefix="xsl"/>
<xsl:template match="/">
  <axsl:stylesheet>
    <xsl:apply-templates/>
  </axsl:stylesheet>
</xsl:template>
<xsl:template match="block">
  <axsl:template match="{.}">
    <fo:block><axsl:apply-templates/></fo:block>
  </axsl:template>
</xsl:template>
</xsl:stylesheet>
```

RXSLT:

```
xslt1.0
  xmlns fo="http://www.w3.org/1999/XSL/Format"
  xmlns axsl="http://www.w3.org/1999/XSL/TransformAlias"
stylesheet-prefix axsl
result-prefix xsl
template /
  <axsl:stylesheet>
    {apply-templates}
  </axsl:stylesheet>
template block
  <axsl:template match="{.}">
     <fo:block><axsl:apply-templates/></fo:block>
  </axsl:template>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:axsl="http://www
    <xsl:apply-templates/>
  </axsl:stylesheet></xsl:template><xsl:template match="block"><axsl:template match="{.}">
     <fo:block><axsl:apply-templates/></fo:block>
  </axsl:template></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:axsl="http://www
  <xsl:namespace-alias stylesheet-prefix="axsl" result-prefix="xsl"/>
  <xsl:template match="/">
    <axsl:stylesheet>
    <xsl:apply-templates/>
  </axsl:stylesheet>
  </xsl:template>
  <xsl:template match="block">
    <axsl:template match="{.}">
     <fo:block><axsl:apply-templates/></fo:block>
  </axsl:template>
  </xsl:template>
</xsl:stylesheet>
```

7.1.3 Creating Attributes with xsl:attribute

XML-encoded XSLT:

```
<xsl:attribute name="a">x
y</xsl:attribute>
```

RXSLT:

```
attribute a : "x
y"
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
y</xsl:attribute></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

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

```
      <xsl:text>x
y</xsl:text>
    </xsl:attribute>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:attribute name="a">x
y</xsl:attribute>
  </xsl:template>
</xsl:stylesheet>
```

7.1.4 Named Attribute Sets

XML-encoded XSLT:

```
<xsl:template match="chapter/heading">
  <fo:block quadding="start" xsl:use-attribute-sets="title-style">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
<xsl:attribute-set name="title-style">
  <xsl:attribute name="font-size">12pt</xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
</xsl:attribute-set>
```

RXSLT:

```
template chapter/heading
  <fo:block quadding="start" {title-style}>
    {apply-templates}
  </fo:block>
attribute-set title-style
  attribute font-size : "12pt"
  attribute font-weight : "bold"
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:apply-templates/>
  </fo:block></xsl:template><xsl:attribute-set name="title-style"><xsl:attribute name="font-size">
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="chapter/heading">
    <fo:block quadding="start"  xsl:use-attribute-sets="title-style">
    <xsl:apply-templates/>
  </fo:block>
  </xsl:template>
  <xsl:attribute-set name="title-style">
    <xsl:attribute name="font-size">
      <xsl:text>12pt</xsl:text>
    </xsl:attribute>
    <xsl:attribute name="font-weight">
      <xsl:text>bold</xsl:text>
    </xsl:attribute>
  </xsl:attribute-set>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="chapter/heading">
    <fo:block quadding="start"  xsl:use-attribute-sets="title-style">
    <xsl:apply-templates/>
  </fo:block>
  </xsl:template>
  <xsl:attribute-set name="title-style">
    <xsl:attribute name="font-size">12pt</xsl:attribute>
    <xsl:attribute name="font-weight">bold</xsl:attribute>
  </xsl:attribute-set>
</xsl:stylesheet>
```

7.3 Creating Processing Instructions

XML-encoded XSLT:

```
<xsl:processing-instruction name="xml-stylesheet">href="book.css" type="text/css"</xsl:processing-
```

RXSLT:

```
processing-instruction xml-stylesheet : 'href="book.css" type="text/css"'
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:processing-instruction name="xml-stylesheet">
      <xsl:text>href=&quot;book.css&quot; type=&quot;text/css&quot;</xsl:text>
    </xsl:processing-instruction>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:processing-instruction name="xml-stylesheet">href=&quot;book.css&quot; type=&quot;text/cs
  </xsl:template>
</xsl:stylesheet>
```

7.4 Creating Comments

XML-encoded XSLT:

```
<xsl:comment>This file is automatically generated. Do not edit!</xsl:comment>
```

RXSLT:

```
comment: "This file is automatically generated. Do not edit!"
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:comment>
      <xsl:text>This file is automatically generated. Do not edit!</xsl:text>
    </xsl:comment>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:comment>This file is automatically generated. Do not edit!</xsl:comment>
  </xsl:template>
</xsl:stylesheet>
```

7.5 Copying (first example)

XML-encoded XSLT:

```
<xsl:template match="@*|node()">
  <xsl:copy>
    <xsl:apply-templates select="@*|node()"/>
  </xsl:copy>
</xsl:template>
```

RXSLT:

```
template @* | node()
  copy
    apply-templates with-nodes @* | node()
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

7.5 Copying (second example – definition)

XML-encoded XSLT:

```
<xsl:template name="apply-templates-copy-lang">
  <xsl:for-each select="@xml:lang">
    <xsl:copy/>
  </xsl:for-each>
  <xsl:apply-templates/>
</xsl:template>
```

RXSLT:

```
named-template apply-templates-copy-lang
 for-each @xml:lang
   copy
 apply-templates
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="apply-templates-copy-lang">
    <xsl:for-each select="@xml:lang">
      <xsl:copy/>
    </xsl:for-each>
    <xsl:apply-templates/>
  </xsl:template>
</xsl:stylesheet>
```

7.5 Copying (second example – use)

XML-encoded XSLT:

```
<xsl:call-template name="apply-templates-copy-lang">
```

RXSLT:

```
call-template apply-templates-copy-lang
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:call-template name="apply-templates-copy-lang"/>
  </xsl:template>
</xsl:stylesheet>
```

7.6.1 Generating Text with xsl:value-of (first example)

XML-encoded XSLT:

```
<xsl:template match="person">
  <p>
   <xsl:value-of select="@given-name"/>
   <xsl:text> </xsl:text>
   <xsl:value-of select="@family-name"/>
  </p>
</xsl:template>
```

RXSLT:

```
template person
  <p>{value-of @given-name; " "; value-of @family-name}</p>
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="person">
    <p><xsl:value-of select="@given-name"/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="@family-name"/></p>
  </xsl:template>
</xsl:stylesheet>
```

7.6.1 Generating Text with xsl:value-of (second example)

XML-encoded XSLT:

```
<xsl:template match="person">
  <p>
   <xsl:value-of select="given-name"/>
   <xsl:text> </xsl:text>
   <xsl:value-of select="family-name"/>
  </p>
</xsl:template>
```

RXSLT:

```
template person
  <p>{value-of given-name; " "; value-of family-name}</p>
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="person">
    <p><xsl:value-of select="given-name"/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="family-name"/></p>
  </xsl:template>
</xsl:stylesheet>
```

7.6.1 Generating Text with xsl:value-of (third example)

XML-encoded XSLT:

```
<xsl:template match="procedure">
  <fo:block>
    <xsl:value-of select="ancestor-or-self::*[@security][1]/@security"/>
  </fo:block>
  <xsl:apply-templates/>
</xsl:template>
```

RXSLT:

```
template procedure
  <fo:block>
    {value-of ancestor-or-self::*[@security][1]/@security}
  </fo:block>
  apply-templates
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:value-of select="ancestor-or-self::*[@security][1]/@security"/>
  </fo:block><xsl:apply-templates/></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="procedure">
    <fo:block>
    <xsl:value-of select="ancestor-or-self::*[@security][1]/@security"/>
  </fo:block>
    <xsl:apply-templates/>
  </xsl:template>
</xsl:stylesheet>
```

7.6.2 Attribute Value Templates (first example)

XML-encoded XSLT:

```
<xsl:variable name="image-dir">/images</xsl:variable>
<xsl:template match="photograph">
<img src="{$image-dir}/{href}" width="{size/@width}"/>
</xsl:template>
```

RXSLT:

```
variable image-dir
  "/images"
template photograph
  <img src="{$image-dir}/{href}" width="{size/@width}"/>
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:variable name="image-dir">
    <xsl:text>/images</xsl:text>
  </xsl:variable>
  <xsl:template match="photograph">
    <img src="{$image-dir}/{href}" width="{size/@width}"/>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:variable name="image-dir">/images</xsl:variable>
  <xsl:template match="photograph">
    <img src="{$image-dir}/{href}" width="{size/@width}"/>
  </xsl:template>
</xsl:stylesheet>
```

7.7 Numbering (numbering a sorted list)

XML-encoded XSLT:

```
<xsl:template match="items">
  <xsl:for-each select="item">
    <xsl:sort select="."/>
    <p>
      <xsl:number value="position()" format="1. "/>
      <xsl:value-of select="."/>
    </p>
  </xsl:for-each>
</xsl:template>
```

RXSLT:

```
template items
  for-each item
    sort using .
    <p>
      {number value=position() format="1. "
       value-of .}
    </p>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
      <xsl:number value="position()" format="1. "/><xsl:value-of select="."/>
    </p></xsl:for-each></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="items">
    <xsl:for-each select="item">
      <xsl:sort select="."/>
      <p>
      <xsl:number value="position()" format="1. "/>
        <xsl:value-of select="."/>
    </p>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

7.7 Numbering (numbering items in an ordered list)

XML-encoded XSLT:

```
<xsl:template match="ol/item"><fo:block>
    <xsl:number/><xsl:text>. </xsl:text><xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

RXSLT:

```
template ol/item
  <fo:block>
    {number; ". "; apply-templates}
  </fo:block>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:number/>. <xsl:apply-templates/>
  </fo:block></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="ol/item">
    <fo:block>
    <xsl:number/>
      <xsl:text>. </xsl:text>
      <xsl:apply-templates/>
  </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="ol/item">
    <fo:block>
    <xsl:number/>. <xsl:apply-templates/>
  </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

7.7 Numbering (numbering titles)

XML-encoded XSLT:

```
<xsl:template match="title">
  <fo:block>
    <xsl:number level="multiple"
                count="chapter|section|subsection"
                format="1.1 "/>
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
<xsl:template match="appendix//title" priority="1">
  <fo:block>
     <xsl:number level="multiple"
                 count="appendix|section|subsection"
                 format="A.1 "/>
     <xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

RXSLT:

```
template title
  <fo:block>
     {multiple-number count=chapter|section|subsection format="1.1 "
      apply-templates}
  </fo:block>
1 template appendix//title
  <fo:block>
     {multiple-number count=appendix|section|subsection format="A.1 "
      apply-templates}
  </fo:block>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
     <xsl:number level="multiple" count="chapter|section|subsection" format="1.1 "/><xsl:apply-ter
  </fo:block></xsl:template><xsl:template match="appendix//title" priority="1"><fo:block>
     <xsl:number level="multiple" count="appendix|section|subsection" format="A.1 "/><xsl:apply-te
  </fo:block></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="title">
    <fo:block>
     <xsl:number level="multiple" count="chapter|section|subsection" format="1.1 "/>
      <xsl:apply-templates/>
  </fo:block>
  </xsl:template>
  <xsl:template match="appendix//title" priority="1">
    <fo:block>
     <xsl:number level="multiple" count="appendix|section|subsection" format="A.1 "/>
      <xsl:apply-templates/>
  </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

7.7 Numbering (numbering notes)

XML-encoded XSLT:

```
<xsl:template match="note">
  <fo:block>
     <xsl:number from="chapter" format="(1) "/>
     <xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

RXSLT:

```
template note
  <fo:block>
     {number from=chapter format="(1) "
      apply-templates}
  </fo:block>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
     <xsl:number from="chapter" format="(1) "/><xsl:apply-templates/>
  </fo:block></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="note">
    <fo:block>
     <xsl:number from="chapter" format="(1) "/>
      <xsl:apply-templates/>
  </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

7.7 Numbering (numbering H4 elements)

XML-encoded XSLT:

```
<xsl:template match="H4">
 <fo:block>
   <xsl:number from="H1" count="H2"/>
   <xsl:text>.</xsl:text>
   <xsl:number from="H2" count="H3"/>
   <xsl:text>.</xsl:text>
   <xsl:number from="H3" count="H4"/>
   <xsl:text> </xsl:text>
   <xsl:apply-templates/>
 </fo:block>
</xsl:template>
```

RXSLT:

```
template H4
 <fo:block>
   {number from=H1 count=H2
    "."
    number from=H2 count=H3
    "."
    number from=H3 count=H4
    " "
    apply-templates}
 </fo:block>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
   <xsl:number from="H1" count="H2"/>.<xsl:number from="H2" count="H3"/>.<xsl:number from="H3" cou
 </fo:block></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="H4">
    <fo:block>
   <xsl:number from="H1" count="H2"/>
      <xsl:text>.</xsl:text>
      <xsl:number from="H2" count="H3"/>
      <xsl:text>.</xsl:text>
      <xsl:number from="H3" count="H4"/>
      <xsl:text> </xsl:text>
      <xsl:apply-templates/>
 </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="H4">
    <fo:block>
   <xsl:number from="H1" count="H2"/>.<xsl:number from="H2" count="H3"/>.<xsl:number from="H3" cou
      <xsl:text> </xsl:text>
      <xsl:apply-templates/>
 </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

8. Repetition

XML-encoded XSLT:

```
<xsl:template match="/">
  <html>
    <head>
      <title>Customers</title>
    </head>
    <body>
      <table>
        <tbody>
          <xsl:for-each select="customers/customer">
            <tr>
              <th>
                <xsl:apply-templates select="name"/>
              </th>
              <xsl:for-each select="order">
                <td>
                  <xsl:apply-templates>
                </td>
              </xsl:for-each>
            </tr>
          </xsl:for-each>
        </tbody>
      </table>
    </body>
  </html>
</xsl:template>
```

RXSLT:

```
template /
  <html>
    <head>
      <title>Customers</title>
    </head>
    <body>
      <table>
        <tbody>
          {for-each customers/customer
            <tr>
              <th>
                {apply-templates with-nodes name}
              </th>
              {for-each order
                <td>
                  {apply-templates}
                </td>
              }
            </tr>
          }
        </tbody>
      </table>
    </body>
  </html>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <head>
      <title>Customers</title>
    </head>
    <body>
      <table>
        <tbody>
          <xsl:for-each select="customers/customer"><tr>
              <th>
                <xsl:apply-templates select="name"/>
              </th>
              <xsl:for-each select="order"><td>
                  <xsl:apply-templates/>
                </td></xsl:for-each>
            </tr></xsl:for-each>
        </tbody>
      </table>
    </body>
  </html></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
    <head>
      <title>Customers</title>
    </head>
    <body>
      <table>
        <tbody>
          <xsl:for-each select="customers/customer">
        <tr>
              <th>
                <xsl:apply-templates select="name"/>
              </th>
              <xsl:for-each select="order">
            <td>
                  <xsl:apply-templates/>
                </td>
          </xsl:for-each>
            </tr>
      </xsl:for-each>
        </tbody>
      </table>
    </body>
  </html>
  </xsl:template>
</xsl:stylesheet>
```

9.1 Conditional Processing with xsl:if (first example)

XML-encoded XSLT:

```
<xsl:template match="namelist/name">
  <xsl:apply-templates>
  <xsl:if select="not(position()=last())">, </xsl:if>
</xsl:template>
```

RXSLT:

```
template namelist/name
  apply-templates
  if not(position()=last()): ", "
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="namelist/name">
    <xsl:apply-templates/>
    <xsl:if test="not(position()=last())">
      <xsl:text>, </xsl:text>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="namelist/name">
    <xsl:apply-templates/>
    <xsl:if test="not(position()=last())">, </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

9.1 Conditional Processing with xsl:if (second example)

XML-encoded XSLT:

```
<xsl:template match="item">
  <tr>
    <xsl:if select="position() mod 2 = 0">
      <xsl:attribute name="bgcolor">yellow</xsl:attribute>
    </xsl:if>
    <xsl:apply-templates/>
  </tr>
</xsl:template>
```

RXSLT:

```
template item
  <tr>
    {
     if position() mod 2 = 0
       attribute bgcolor
         "yellow"
     apply-templates}
  </tr>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:if test="position() mod 2=0"><xsl:attribute name="bgcolor">yellow</xsl:attribute></xsl:i
  </tr></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="item">
    <tr>
    <xsl:if test="position() mod 2=0">
        <xsl:attribute name="bgcolor">
          <xsl:text>yellow</xsl:text>
        </xsl:attribute>
      </xsl:if>
      <xsl:apply-templates/>
  </tr>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="item">
    <tr>
    <xsl:if test="position() mod 2=0">
        <xsl:attribute name="bgcolor">yellow</xsl:attribute>
      </xsl:if>
      <xsl:apply-templates/>
  </tr>
  </xsl:template>
</xsl:stylesheet>
```

9.2 Conditional Processing with xsl:choose

XML-encoded XSLT:

```
<xsl:template match="orderedlist/listitem">
  <fo:list-item indent-start='2pi'>
    <fo:list-item-label>
      <xsl:variable name="level"
                    select="count(ancestor::orderedlist) mod 3"/>
      <xsl:choose>
        <xsl:when test="$level=1">
          <xsl:number format="i"/>
        </xsl:when>
        <xsl:when test="$level=2">
          <xsl:number format="a"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:number format="1"/>
        </xsl:otherwise>
      </xsl:choose>.
    </fo:list-item-label>
    <fo:list-item-body>
      <xsl:apply-templates/>
    </fo:list-item-body>
  </fo:list-item>
</xsl:template>
```

RXSLT:

```
template orderedlist/listitem
  <fo:list-item indent-start='2pi'>
    <fo:list-item-label>
      {variable level = count(ancestor::orderedlist) mod 3
```

```
      choose
        when $level=1
          number format=i
        when $level=2
          number format=a
        otherwise
          number format=1
      ". "}
    </fo:list-item-label>
    <fo:list-item-body>
      {apply-templates}
    </fo:list-item-body>
  </fo:list-item>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <fo:list-item-label>
      <xsl:variable name="level" select="count(ancestor::orderedlist) mod 3"/><xsl:choose><xsl:whe
    </fo:list-item-label>
    <fo:list-item-body>
      <xsl:apply-templates/>
    </fo:list-item-body>
  </fo:list-item></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="orderedlist/listitem">
    <fo:list-item indent-start='2pi'>
    <fo:list-item-label>
      <xsl:variable name="level" select="count(ancestor::orderedlist) mod 3"/>
      <xsl:choose>
        <xsl:when test="$level=1">
          <xsl:number format="i"/>
        </xsl:when>
        <xsl:when test="$level=2">
          <xsl:number format="a"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:number format="1"/>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:text>. </xsl:text>
    </fo:list-item-label>
    <fo:list-item-body>
      <xsl:apply-templates/>
    </fo:list-item-body>
  </fo:list-item>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="orderedlist/listitem">
    <fo:list-item indent-start='2pi'>
    <fo:list-item-label>
      <xsl:variable name="level" select="count(ancestor::orderedlist) mod 3"/>
```

```
      <xsl:choose>
        <xsl:when test="$level=1">
          <xsl:number format="i"/>
        </xsl:when>
        <xsl:when test="$level=2">
          <xsl:number format="a"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:number format="1"/>
        </xsl:otherwise>
      </xsl:choose>.
    </fo:list-item-label>
    <fo:list-item-body>
      <xsl:apply-templates/>
    </fo:list-item-body>
  </fo:list-item>
  </xsl:template>
</xsl:stylesheet>
```

10. Sorting

XML-encoded XSLT:

```
<xsl:template match="employees">
  <ul>
    <xsl:apply-templates select="employee">
      <xsl:sort select="name/family"/>
      <xsl:sort select="name/given"/>
    </xsl:apply-templates>
  </ul>
</xsl:template>
<xsl:template match="employee">
  <li>
    <xsl:value-of select="name/given"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="name/family"/>
  </li>
</xsl:template>
```

RXSLT:

```
template employees
  <ul>
    {apply-templates with-nodes employee
       sort using name/family
       sort using name/given
    }
  </ul>
template employee
  <li>
    {value-of name/given; " "; value-of name/family}
  </li>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:apply-templates select="employee"><xsl:sort select="name/family"/><xsl:sort select="name/
  </ul></xsl:template><xsl:template match="employee"><li>
    <xsl:value-of select="name/given"/> <xsl:value-of select="name/family"/>
  </li></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="employees">
    <ul>
    <xsl:apply-templates select="employee">
        <xsl:sort select="name/family"/>
        <xsl:sort select="name/given"/>
      </xsl:apply-templates>
  </ul>
  </xsl:template>
  <xsl:template match="employee">
    <li>
    <xsl:value-of select="name/given"/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="name/family"/>
  </li>
  </xsl:template>
</xsl:stylesheet>
```

11.2 Values of Variables and Parameters (various examples)

XML-encoded XSLT:

```
<xsl:variable name="x"/>
<xsl:variable name="x" select="''"/>
<xsl:variable name="n" select="2"/>
<xsl:value-of select="item[$n]"/>
<xsl:variable name="n">2</xsl:variable>
<xsl:value-of select="item[position()=$n]"/>
<xsl:param name="x" select="/.."/>
```

RXSLT:

```
variable x
variable x = ''
variable n = 2
value-of item[$n]
variable n : "2"
value-of item[position()=$n]
param x = /..
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:variable name="x"/>
  <xsl:variable name="x" select="''"/>
  <xsl:variable name="n" select="2"/>
  <xsl:template match="/">
    <xsl:value-of select="item[$n]"/>
    <xsl:variable name="n">
      <xsl:text>2</xsl:text>
    </xsl:variable>
    <xsl:value-of select="item[position()=$n]"/>
    <xsl:param name="x" select="/.."/>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:variable name="x"/>
  <xsl:variable name="x" select="''"/>
  <xsl:variable name="n" select="2"/>
  <xsl:template match="/">
    <xsl:value-of select="item[$n]"/>
    <xsl:variable name="n">2</xsl:variable>
    <xsl:value-of select="item[position()=$n]"/>
    <xsl:param name="x" select="/.."/>
  </xsl:template>
</xsl:stylesheet>
```

11.4 Top-level Variables and Parameters

XML-encoded XSLT:

```
<xsl:variable name="para-font-size">12pt</xsl:variable>
<xsl:template match="para">
 <fo:block font-size="{$para-font-size}">
   <xsl:apply-templates/>
 </fo:block>
</xsl:template>
```

RXSLT:

```
variable para-font-size : "12pt"
template para
 <fo:block font-size="{$para-font-size}">
   {apply-templates}
 </fo:block>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
   <xsl:apply-templates/>
 </fo:block></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:variable name="para-font-size">
    <xsl:text>12pt</xsl:text>
  </xsl:variable>
  <xsl:template match="para">
    <fo:block font-size="{$para-font-size}">
   <xsl:apply-templates/>
 </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:variable name="para-font-size">12pt</xsl:variable>
  <xsl:template match="para">
    <fo:block font-size="{$para-font-size}">
   <xsl:apply-templates/>
 </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

11.5 Variables and Parameters within Templates (the allowed version)

XML-encoded XSLT:

```
<xsl:param name="x" select="1"/>
<xsl:template name="foo">
<xsl:variable name="x" select="2"/>
</xsl:template>
```

RXSLT:

```
param x = 1
named-template foo
  variable x = 2
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="x" select="1"/>
  <xsl:template name="foo">
    <xsl:variable name="x" select="2"/>
  </xsl:template>
</xsl:stylesheet>
```

11.6 Passing Parameters to Templates

XML-encoded XSLT:

```
<xsl:template name="numbered-block">
  <xsl:param name="format">1. </xsl:param>
  <fo:block>
    <xsl:number format="{$format}"/>
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
<xsl:template match="ol//ol/li">
  <xsl:call-template name="numbered-block">
    <xsl:with-param name="format">a. </xsl:with-param>
  </xsl:call-template>
</xsl:template>
```

RXSLT:

```
named-template numbered-block
  param format : "1. "
  <fo:block>
    {number format="{$format}"
     apply-templates}
  </fo:block>
template ol//ol/li
  call-template numbered-block
    with-param format : "a. "
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <xsl:number format="{$format}"/><xsl:apply-templates/>
  </fo:block></xsl:template><xsl:template match="ol//ol/li"><xsl:call-template name="numbered-bloc
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="numbered-block">
    <xsl:param name="format">
      <xsl:text>1. </xsl:text>
    </xsl:param>
    <fo:block>
    <xsl:number format="{$format}"/>
      <xsl:apply-templates/>
  </fo:block>
  </xsl:template>
  <xsl:template match="ol//ol/li">
    <xsl:call-template name="numbered-block">
      <xsl:with-param name="format">
        <xsl:text>a. </xsl:text>
      </xsl:with-param>
    </xsl:call-template>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="numbered-block">
    <xsl:param name="format">1. </xsl:param>
    <fo:block>
    <xsl:number format="{$format}"/>
      <xsl:apply-templates/>
  </fo:block>
  </xsl:template>
  <xsl:template match="ol//ol/li">
    <xsl:call-template name="numbered-block">
      <xsl:with-param name="format">a. </xsl:with-param>
    </xsl:call-template>
  </xsl:template>
</xsl:stylesheet>
```

12.2 Keys (div element with attribute id)

XML-encoded XSLT:

```
<xsl:key name="idkey" match="div" use="@id">
```

RXSLT:

```
key idkey match div use @id
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:key name="idkey" match="div" use="@id">
</xsl:stylesheet>
```

12.2 Keys (prototype element)

XML-encoded XSLT:

```
<xsl:key name="func" match="prototype" use="@name">
<xsl:template match="function"><b>
    <a href="#{generate-id(key('func',.))}">
      <xsl:apply-templates/>
    </a>
  </b>
</xsl:template>
<xsl:template match="prototype">
<p><a name="{generate-id()}">
<b>Function: </b>
...
</a></p>
</xsl:template>
```

RXSLT:

```
key func match prototype use @name
template function
  <b>
    <a href="#{generate-id(key('func',.))}">
      {apply-templates}
    </a>
  </b>
template prototype
  <p><a name="{generate-id()}">
  <b>Function: </b>
  ...
  </a></p>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
    <a href="#{generate-id(key('func',.))}">
      <xsl:apply-templates/>
    </a>
  </b></xsl:template><xsl:template match="prototype"><p><a name="{generate-id()}">
  <b>Function: </b>
  ...
  </a></p></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:key name="func" match="prototype" use="@name">
  <xsl:template match="function">
    <b>
    <a href="#{generate-id(key('func',.))}">
      <xsl:apply-templates/>
    </a>
  </b>
  </xsl:template>
  <xsl:template match="prototype">
    <p><a name="{generate-id()}">
  <b>Function: </b>
  ...
  </a></p>
  </xsl:template>
</xsl:stylesheet>
```

12.2 Keys (from another document)

XML-encoded XSLT:

```
<xsl:key name="bib" match="entry" use="@name">
<xsl:template match="bibref">
  <xsl:variable name="name" select="."/>
  <xsl:for-each select="document('bib.xml')">
    <xsl:apply-templates select="key('bib',$name)"/>
  </xsl:for-each>
</xsl:template>
```

RXSLT:

```
key bib match entry use @name
template bibref
  variable name = .
  for-each document('bib.xml')
    apply-templates with-nodes key('bib',$name)
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:key name="bib" match="entry" use="@name">
  <xsl:template match="bibref">
    <xsl:variable name="name" select="."/>
    <xsl:for-each select="document('bib.xml')">
      <xsl:apply-templates select="key('bib',$name)"/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

12.4 Miscellaneous Additional Functions

XML-encoded XSLT:

```
<xsl:value-of select="current()"/>
<xsl:value-of select="."/>
<xsl:apply-templates select="//glossary/item[@name=current()/@ref]"/>
<xsl:apply-templates select="//glossary/item[@name=./@ref]"/>
<xsl:apply-templates select="//glossary/item[@name=@ref]"/>
```

RXSLT:

```
value-of current()
value-of .
apply-templates with-nodes //glossary/item[@name=current()/@ref]
apply-templates with-nodes //glossary/item[@name=./@ref]
apply-templates with-nodes //glossary/item[@name=@ref]
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:value-of select="current()"/>
    <xsl:value-of select="."/>
    <xsl:apply-templates select="//glossary/item[@name=current()/@ref]"/>
    <xsl:apply-templates select="//glossary/item[@name=./@ref]"/>
    <xsl:apply-templates select="//glossary/item[@name=@ref]"/>
  </xsl:template>
</xsl:stylesheet>
```

13 Messages

XML-encoded XSLT:

```
<xsl:param name="lang" select="en"/>
<xsl:variable name="messages"
  select="document(concat('resources/', $lang, '.xml'))/messages"/>
<xsl:template name="localized-message">
  <xsl:param name="name"/>
  <xsl:message>
    <xsl:value-of select="$messages/message[@name=$name]"/>
  </xsl:message>
</xsl:template>
<xsl:template name="problem">
  <xsl:call-template name="localized-message">
    <xsl:with-param name="name">problem</xsl:with-param>
  </xsl:call-template>
</xsl:template>
```

RXSLT:

```
param lang = en
variable messages = document(concat('resources/', $lang, '.xml'))/messages
named-template localized-message
  param name
  message : value-of $messages/message[@name=$name]
named-template problem
  call-template localized-message
    with-param name : "problem"
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="lang" select="en"/>
  <xsl:variable name="messages" select="document(concat('resources/', $lang,'.xml'))/messages"/>
  <xsl:template name="localized-message">
    <xsl:param name="name"/>
    <xsl:message>
      <xsl:value-of select="$messages/message[@name=$name]"/>
    </xsl:message>
  </xsl:template>
  <xsl:template name="problem">
    <xsl:call-template name="localized-message">
      <xsl:with-param name="name">
        <xsl:text>problem</xsl:text>
```

```
      </xsl:with-param>
    </xsl:call-template>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="lang" select="en"/>
  <xsl:variable name="messages" select="document(concat('resources/', $lang,'.xml'))/messages"/>
  <xsl:template name="localized-message">
    <xsl:param name="name"/>
    <xsl:message>
      <xsl:value-of select="$messages/message[@name=$name]"/>
    </xsl:message>
  </xsl:template>
  <xsl:template name="problem">
    <xsl:call-template name="localized-message">
      <xsl:with-param name="name">problem</xsl:with-param>
    </xsl:call-template>
  </xsl:template>
</xsl:stylesheet>
```

16.1 XML Output Method

XML-encoded XSLT:

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

RXSLT:

```
output cdata-section-elements=example
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output cdata-section-elements="example"/>
</xsl:stylesheet>
```

16.2 HTML Output Method

XML-encoded XSLT:

```
<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>
```

RXSLT:

```
output method=html
template /
  <html>
   {apply-templates}
  </html>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
   <xsl:apply-templates/>
  </html></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<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>
```

16.4 Disabling Output Escaping

XML-encoded XSLT:

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

RXSLT:

```
unescaped "&lt;"
```

Output from RXSLT compiler (non-indented):

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

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:text disable-output-escaping="yes">&lt;</xsl:text>
  </xsl:template>
</xsl:stylesheet>
```

D.1 Document Example

XML-encoded XSLT:

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/TR/xhtml1/strict">

<xsl:strip-space elements="doc chapter section"/>
<xsl:output
   method="xml"
   indent="yes"
   encoding="iso-8859-1"
/>

<xsl:template match="doc">
```

```
  <html>
    <head>
      <title>
        <xsl:value-of select="title"/>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="doc/title">
  <h1>
    <xsl:apply-templates/>
  </h1>
</xsl:template>

<xsl:template match="chapter/title">
  <h2>
    <xsl:apply-templates/>
  </h2>
</xsl:template>

<xsl:template match="section/title">
  <h3>
    <xsl:apply-templates/>
  </h3>
</xsl:template>

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

<xsl:template match="note">
  <p class="note">
    <b>NOTE: </b>
    <xsl:apply-templates/>
  </p>
</xsl:template>

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

</xsl:stylesheet>
```

RXSLT:

```
xslt1.0 xmlns = "http://www.w3.org/TR/xhtml1/strict"

strip-space doc, chapter, section
output method=xml
       indent=yes
```

```
       encoding=iso-8859-1

template doc
 <html>
   <head>
     <title>
       {value-of title}
     </title>
   </head>
   <body>
     {apply-templates}
   </body>
 </html>

template doc/title
  <h1>
    {apply-templates}
  </h1>

template chapter/title
  <h2>
    {apply-templates}
  </h2>

template section/title
  <h3>
    {apply-templates}
  </h3>

template para
  <p>
    {apply-templates}
  </p>

template note
  <p class="note">
    <b>NOTE: </b>
    {apply-templates}
  </p>

template emph
  <em>
    {apply-templates}
  </em>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns="http://www.w3.org/TR/xhtml1/strict" xmlns:xsl="http://www.w3
   <head>
     <title>
       <xsl:value-of select="title"/>
     </title>
   </head>
   <body>
     <xsl:apply-templates/>
   </body>
 </html></xsl:template><xsl:template match="doc/title"><h1>
    <xsl:apply-templates/>
```

```
  </h1></xsl:template><xsl:template match="chapter/title"><h2>
    <xsl:apply-templates/>
  </h2></xsl:template><xsl:template match="section/title"><h3>
    <xsl:apply-templates/>
  </h3></xsl:template><xsl:template match="para"><p>
    <xsl:apply-templates/>
  </p></xsl:template><xsl:template match="note"><p class="note">
    <b>NOTE: </b>
    <xsl:apply-templates/>
  </p></xsl:template><xsl:template match="emph"><em>
    <xsl:apply-templates/>
  </em></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns="http://www.w3.org/TR/xhtml1/strict" xmlns:xsl="http://www.w3
  <xsl:strip-space elements="doc chapter section">
  <xsl:output method="xml" indent="yes" encoding="iso-8859-1"/>
  <xsl:template match="doc">
    <html>
   <head>
     <title>
       <xsl:value-of select="title"/>
     </title>
   </head>
   <body>
     <xsl:apply-templates/>
   </body>
 </html>
  </xsl:template>
  <xsl:template match="doc/title">
    <h1>
    <xsl:apply-templates/>
  </h1>
  </xsl:template>
  <xsl:template match="chapter/title">
    <h2>
    <xsl:apply-templates/>
  </h2>
  </xsl:template>
  <xsl:template match="section/title">
    <h3>
    <xsl:apply-templates/>
  </h3>
  </xsl:template>
  <xsl:template match="para">
    <p>
    <xsl:apply-templates/>
  </p>
  </xsl:template>
  <xsl:template match="note">
    <p class="note">
    <b>NOTE: </b>
    <xsl:apply-templates/>
  </p>
  </xsl:template>
  <xsl:template match="emph">
    <em>
```

```
    <xsl:apply-templates/>
  </em>
  </xsl:template>
</xsl:stylesheet>
```

D.2 Data Example (HTML outputter)

XML-encoded XSLT:

```
<html xsl:version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      lang="en">
    <head>
        <title>Sales Results By Division</title>
    </head>
    <body>
        <table border="1">
            <tr>
                <th>Division</th>
                <th>Revenue</th>
                <th>Growth</th>
                <th>Bonus</th>
            </tr>
            <xsl:for-each select="sales/division">
                <xsl:sort select="revenue"
                          data-type="number"
                          order="descending"/>
                <tr>
                    <td>
                        <em><xsl:value-of select="@id"/></em>
                    </td>
                    <td>
                        <xsl:value-of select="revenue"/>
                    </td>
                    <td>
                        <xsl:if select="growth &lt; 0">
                             <xsl:attribute name="style">
                                 <xsl:text>color:red</xsl:text>
                             </xsl:attribute>
                        </xsl:if>
                        <xsl:value-of select="growth"/>
                    </td>
                    <td>
                        <xsl:value-of select="bonus"/>
                    </td>
                </tr>
            </xsl:for-each>
        </table>
    </body>
</html>
```

RXSLT:

```
<html lang="en">
  <head>
      <title>Sales Results By Division</title>
  </head>
  <body>
      <table border="1">
```

```
          <tr>
              <th>Division</th>
              <th>Revenue</th>
              <th>Growth</th>
              <th>Bonus</th>
          </tr>
          {
            for-each sales/division
              # order the result by revenue
              sort using revenue
                         data-type="number"
                         order="descending"
              <tr>
                  <td>
                      <em>{value-of @id}</em>
                  </td>
                  <td>
                      {value-of revenue}
                  </td>
                  <td>
                      { # highlight negative growth in red
                        if growth < 0
                           attribute style : "color:red"
                        value-of growth
                      }
                  </td>
                  <td>
                      {value-of bonus}
                  </td>
              </tr>
          }
      </table>
  </body>
</html>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xml:space="preserve
  <head>
      <title>Sales Results By Division</title>
  </head>
  <body>
      <table border="1">
          <tr>
              <th>Division</th>
              <th>Revenue</th>
              <th>Growth</th>
              <th>Bonus</th>
          </tr>
          <xsl:for-each select="sales/division"><xsl:sort select="revenue" data-type="number" orde
                  <td>
                      <em><xsl:value-of select="@id"/></em>
                  </td>
                  <td>
                      <xsl:value-of select="revenue"/>
                  </td>
                  <td>
                      <xsl:if test="growth<0"><xsl:attribute name="style">color:red</xsl:attribute
```

```
                </td>
                <td>
                    <xsl:value-of select="bonus"/>
                </td>
            </tr></xsl:for-each>
      </table>
  </body>
</html></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html lang="en">
  <head>
      <title>Sales Results By Division</title>
  </head>
  <body>
      <table border="1">
          <tr>
              <th>Division</th>
              <th>Revenue</th>
              <th>Growth</th>
              <th>Bonus</th>
          </tr>
          <xsl:for-each select="sales/division">
        <xsl:sort select="revenue" data-type="number" order="descending"/>
        <tr>
                <td>
                    <em><xsl:value-of select="@id"/></em>
                </td>
                <td>
                    <xsl:value-of select="revenue"/>
                </td>
                <td>
                    <xsl:if test="growth<0">
            <xsl:attribute name="style">
              <xsl:text>color:red</xsl:text>
            </xsl:attribute>
          </xsl:if>
          <xsl:value-of select="growth"/>
                </td>
                <td>
                    <xsl:value-of select="bonus"/>
                </td>
            </tr>
      </xsl:for-each>
      </table>
  </body>
</html>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html lang="en">
  <head>
```

```
      <title>Sales Results By Division</title>
  </head>
  <body>
      <table border="1">
          <tr>
              <th>Division</th>
              <th>Revenue</th>
              <th>Growth</th>
              <th>Bonus</th>
          </tr>
          <xsl:for-each select="sales/division">
        <xsl:sort select="revenue" data-type="number" order="descending"/>
        <tr>
                  <td>
                      <em><xsl:value-of select="@id"/></em>
                  </td>
                  <td>
                      <xsl:value-of select="revenue"/>
                  </td>
                  <td>
                      <xsl:if test="growth<0">
            <xsl:attribute name="style">color:red</xsl:attribute>
          </xsl:if>
          <xsl:value-of select="growth"/>
                  </td>
                  <td>
                      <xsl:value-of select="bonus"/>
                  </td>
              </tr>
      </xsl:for-each>
      </table>
  </body>
</html>
  </xsl:template>
</xsl:stylesheet>
```

D.2 Data Example (SVG outputter)

XML-encoded XSLT:

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/Graphics/SVG/SVG-19990812.dtd">

<xsl:output method="xml" indent="yes" media-type="image/svg"/>

<xsl:template match="/">

<svg width = "3in" height="3in">
    <g style = "stroke: #000000">
        <!-- draw the axes -->
        <line x1="0" x2="150" y1="150" y2="150"/>
        <line x1="0" x2="0" y1="0" y2="150"/>
        <text x="0" y="10">Revenue</text>
        <text x="150" y="165">Division</text>
        <xsl:for-each select="sales/division">
            <!-- define some useful variables -->
```

```
            <!-- the bar's x position -->
            <xsl:variable name="pos" select="(position()*40)-30"/>

            <!-- the bar's height -->
            <xsl:variable name="height" select="revenue*10"/>

            <!-- the rectangle -->
            <rect x="{$pos}" y="{150-$height}"
                  width="20" height="{$height}"/>

            <!-- the text label -->
            <text x="{$pos}" y="165">
                <xsl:value-of select="@id"/>
            </text>

            <!-- the bar value -->
            <text x="{$pos}" y="{145-$height}">
                <xsl:value-of select="revenue"/>
            </text>
        </xsl:for-each>
    </g>
</svg>

</xsl:template>
</xsl:stylesheet>
```

RXSLT:

```
xslt1.0 xmlns="http://www.w3.org/Graphics/SVG/SVG-19990812.dtd"

output method=xml indent=yes media-type="image/svg"

template /
  <svg width = "3in" height="3in">
      <g style = "stroke: #000000">
          <!-- draw the axes -->
          <line x1="0" x2="150" y1="150" y2="150"/>
          <line x1="0" x2="0" y1="0" y2="150"/>
          <text x="0" y="10">Revenue</text>
          <text x="150" y="165">Division</text>
          {
            for-each sales/division
              # define some useful variables

              # the bar's x position
              variable pos = (position()*40)-30

              # the bar's height
              variable height = revenue*10

              # the rectangle
              <rect x="{$pos}" y="{150-$height}"
                    width="20" height="{$height}"/>

              # the text label
              <text x="{$pos}" y="165">
                  <xsl:value-of select="@id"/>
              </text>
```

```
            # the bar value
            <text x="{$pos}" y="{145-$height}">
                {value-of revenue}
            </text>
        }
    </g>
  </svg>
```

Output from RXSLT compiler (non-indented):

```
<xsl:stylesheet version="1.0" xmlns="http://www.w3.org/Graphics/SVG/SVG-19990812.dtd" xmlns:xsl="h
      <g style = "stroke: #000000">
          <!-- draw the axes -->
          <line x1="0" x2="150" y1="150" y2="150"/>
          <line x1="0" x2="0" y1="0" y2="150"/>
          <text x="0" y="10">Revenue</text>
          <text x="150" y="165">Division</text>
          <xsl:for-each select="sales/division"><xsl:variable name="pos" select="(position()*40)"/
                    width="20" height="{$height}"/><text x="{$pos}" y="165">
                <xsl:value-of select="@id"/>
            </text><text x="{$pos}" y="{145-$height}">
                <xsl:value-of select="revenue"/>
            </text></xsl:for-each>
      </g>
  </svg></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns="http://www.w3.org/Graphics/SVG/SVG-19990812.dtd" xmlns:xsl="h
  <xsl:output method="xml" indent="yes" media-type="image/svg"/>
  <xsl:template match="/">
    <svg width = "3in" height="3in">
      <g style = "stroke: #000000">
          <!-- draw the axes -->
          <line x1="0" x2="150" y1="150" y2="150"/>
          <line x1="0" x2="0" y1="0" y2="150"/>
          <text x="0" y="10">Revenue</text>
          <text x="150" y="165">Division</text>
          <xsl:for-each select="sales/division">
        <xsl:variable name="pos" select="(position()*40)"/>
        <xsl:variable name="height" select="revenue*10"/>
        <rect x="{$pos}" y="{150-$height}"
                    width="20" height="{$height}"/>
        <text x="{$pos}" y="165">
                  <xsl:value-of select="@id"/>
              </text>
        <text x="{$pos}" y="{145-$height}">
                  <xsl:value-of select="revenue"/>
              </text>
      </xsl:for-each>
      </g>
  </svg>
  </xsl:template>
</xsl:stylesheet>
```

D.2 Data Example (VRML outputter)

XML-encoded XSLT:

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

<!-- generate text output as mime type model/vrml, using default charset -->
<xsl:output method="text" encoding="UTF-8" media-type="model/vrml"/>
<xsl:template match="/">#VRML V2.0 utf8

# externproto definition of a single bar element
EXTERNPROTO bar [
  field SFInt32 x
  field SFInt32 y
  field SFInt32 z
  field SFString name
  ]
  "http://www.vrml.org/WorkingGroups/dbwork/barProto.wrl"

# inline containing the graph axes
Inline {
        url "http://www.vrml.org/WorkingGroups/dbwork/barAxes.wrl"
        }
                <xsl:for-each select="sales/division">
bar {
        x <xsl:value-of select="revenue">
        y <xsl:value-of select="growth">
        z <xsl:value-of select="bonus">
        name "<xsl:value-of select="@id">"
        }
                </xsl:for-each>

        </xsl:template>

</xsl:stylesheet>
```

RXSLT:

```
# generate text output as mime type model/vrml, using default charset
output method=text encoding=UTF-8 media-type="model/vrml"

template / : '#VRML V2.0 utf8

# externproto definition of a single bar element
EXTERNPROTO bar [
  field SFInt32 x
  field SFInt32 y
  field SFInt32 z
  field SFString name
  ]
  "http://www.vrml.org/WorkingGroups/dbwork/barProto.wrl"

# inline containing the graph axes
Inline {
        url "http://www.vrml.org/WorkingGroups/dbwork/barAxes.wrl"
        }
'
  for-each sales/division
    'bar {
        x '; value-of revenue; '
        y '; value-of growth; '
```

```
        z '; value-of bonus; '
        name "'; value-of @id; '"
        }}
'
```

Output from RXSLT compiler (non-indented):

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

# externproto definition of a single bar element
EXTERNPROTO bar [
  field SFInt32 x
  field SFInt32 y
  field SFInt32 z
  field SFString name
  ]
  &amp;quot;http://www.vrml.org/WorkingGroups/dbwork/barProto.wrl&amp;quot;

# inline containing the graph axes
Inline {
        url &amp;quot;http://www.vrml.org/WorkingGroups/dbwork/barAxes.wrl&amp;quot;
        }
<xsl:for-each select="sales/division">bar {
        x <xsl:value-of select="revenue"/>
        y <xsl:value-of select="growth"/>
        z <xsl:value-of select="bonus"/>
        name &amp;quot;<xsl:value-of select="@id"/>&amp;quot;
        }}
</xsl:for-each></xsl:template></xsl:stylesheet>
```

Output from RXSLT compiler (indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text" encoding="UTF-8" media-type="model/vrml"/>
  <xsl:template match="/">
    <xsl:text>#VRML V2.0 utf8

# externproto definition of a single bar element
EXTERNPROTO bar [
  field SFInt32 x
  field SFInt32 y
  field SFInt32 z
  field SFString name
  ]
  &amp;quot;http://www.vrml.org/WorkingGroups/dbwork/barProto.wrl&amp;quot;

# inline containing the graph axes
Inline {
        url &amp;quot;http://www.vrml.org/WorkingGroups/dbwork/barAxes.wrl&amp;quot;
        }
</xsl:text>
    <xsl:for-each select="sales/division">
      <xsl:text>bar {
        x </xsl:text>
      <xsl:value-of select="revenue"/>
      <xsl:text>
        y </xsl:text>
      <xsl:value-of select="growth"/>
      <xsl:text>
```

```
        z </xsl:text>
      <xsl:value-of select="bonus"/>
      <xsl:text>
        name &quot;</xsl:text>
      <xsl:value-of select="@id"/>
      <xsl:text>&quot;
        }}
</xsl:text>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Output from RXSLT compiler (better indented):

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text" encoding="UTF-8" media-type="model/vrml"/>
  <xsl:template match="/">#VRML V2.0 utf8

# externproto definition of a single bar element
EXTERNPROTO bar [
  field SFInt32 x
  field SFInt32 y
  field SFInt32 z
  field SFString name
  ]
  &quot;http://www.vrml.org/WorkingGroups/dbwork/barProto.wrl&quot;

# inline containing the graph axes
Inline {
        url &quot;http://www.vrml.org/WorkingGroups/dbwork/barAxes.wrl&quot;
        }
<xsl:for-each select="sales/division">bar {
        x <xsl:value-of select="revenue"/>
        y <xsl:value-of select="growth"/>
        z <xsl:value-of select="bonus"/>
        name &quot;<xsl:value-of select="@id"/>&quot;
        }}
</xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```