(12) UK Patent Application (19) GB (11) 2 382 174 (13) A

(43) Date of A Publication 21.05.2003

(21) Application No 0127722.7

(22) Date of Filing 20.11.2001

(71) Applicant(s)
Hewlett-Packard Company
(Incorporated in USA - Delaware)
3000 Hanover Street, Palo Alto,
California 94304, United States of America

(72) Inventor(s)
Fabio Giannetti

(74) Agent and/or Address for Service
Richard Anthony Lawrence
Hewlett-Packard Limited, IP Section,
Filton Road, Stoke Gifford, BRISTOL,
BS34 8QZ, United Kingdom

(51) INT CL[7]
G06F 17/30 17/21

(52) UK CL (Edition V)
G4A AMX AUDB

(56) Documents Cited
WO 2001/090873 A1 **WO 2001/063481 A2**
WO 2000/023912 A1 **WO 2000/003332 A1**
US 6101513 A
http://www.oasis-open.org/cover/rml.html, "The XML Voer Pages. Relational Markup Language (RML)", R Cover, Last modified 2 June 2001

(58) Field of Search
UK CL (Edition T) **G4A** AMX AUDB
INT CL[7] **G06F** 17/21 17/30
Other: **Online: WPI, EPODOC, PAJ, INSPEC, XPESP, IBM TDB, COMPUTER, Selected Internet sites**

(54) Abstract Title
Data formatting in a platform independent manner

(57) A method of generating data suitable for transmission to at least one data-receiving device, said method comprising the following steps: specifying said data in a first and a second portion, said first portion being substantially independent of any formatting, and said second portion containing said formatting for said first portion specified in a platform independent manner; transforming said second portion, using a first transform, to generate a platform dependent portion containing said formatting specified in a platform dependent manner; and combining said first portion with said platform dependent portion using a second transform to generate said data suitable for transmission to said at least one data-receiving device. This method may be suitable for generating data suitable for transmission to a variety of platforms including WML, HTML, XSL-FO, etc. based devices, for use, for example, with the World Wide Web.

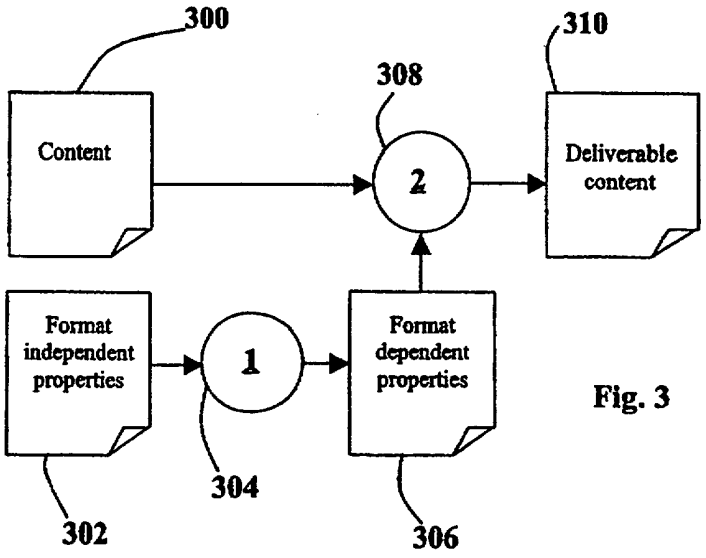

Fig. 3

GB 2 382 174 A

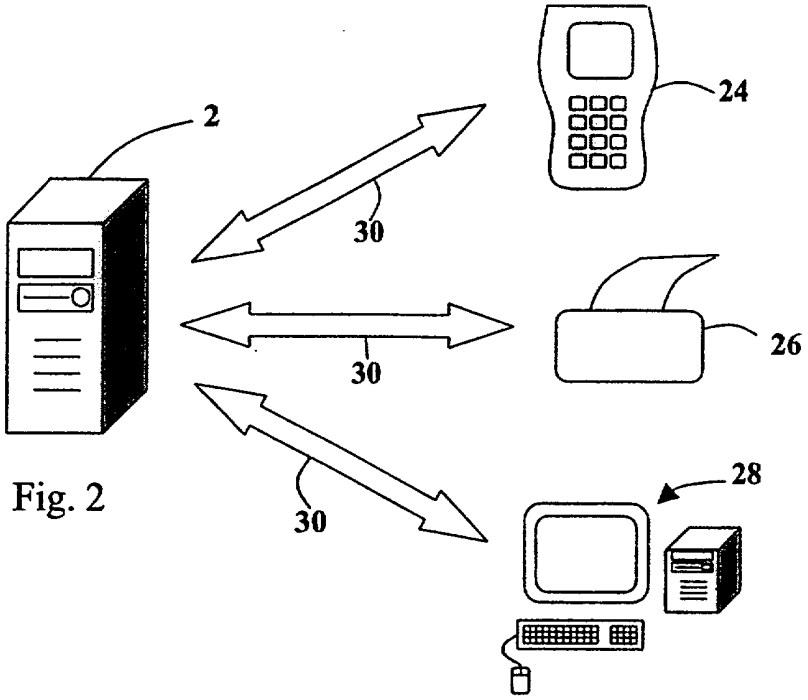

Fig. 2

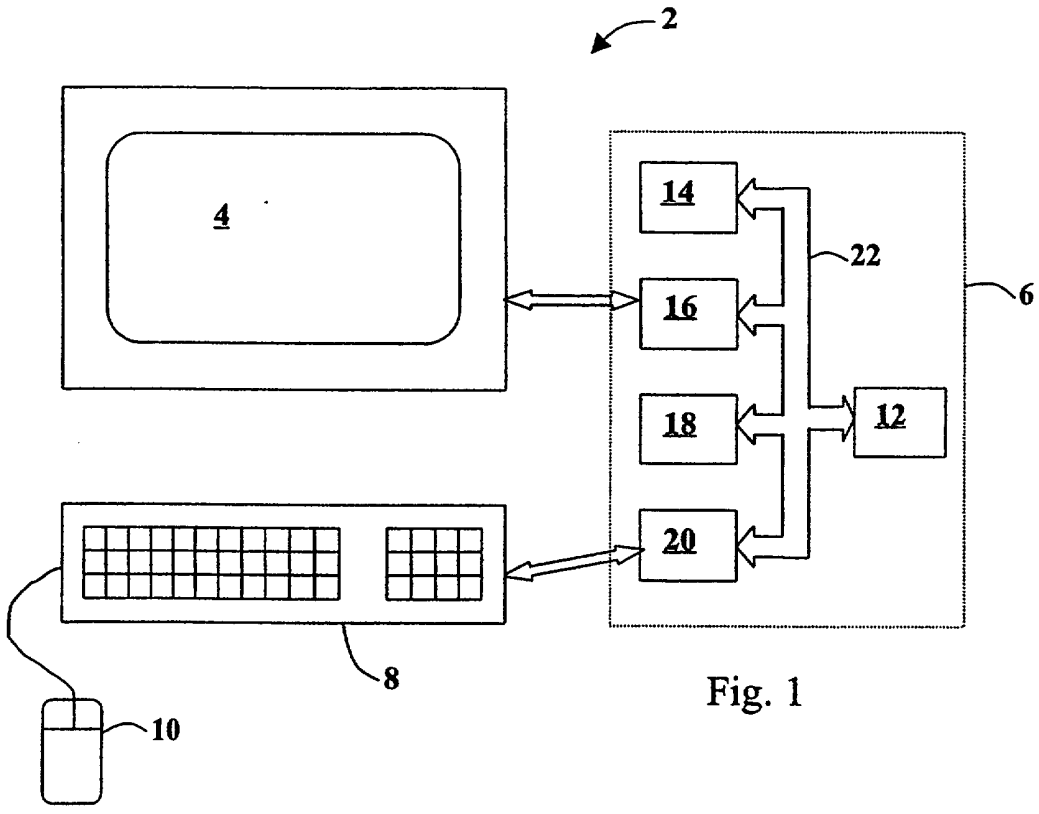

Fig. 1

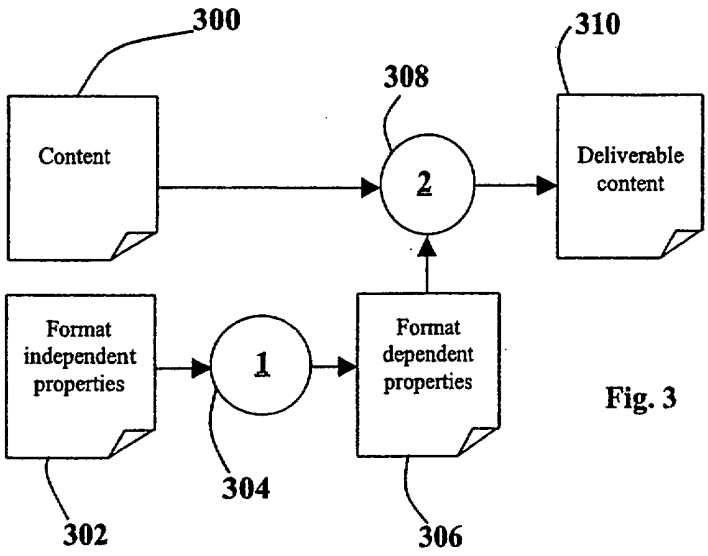
300
310
308
Content
2
Deliverable content
Format independent properties
1
Format dependent properties
304
302
306

Fig. 3

IMPROVEMENTS RELATING TO DATA PROCESSING

This invention relates to a method of generating data, an improved apparatus for performing such generation, together with an improved data structure facilitating said method.

Many documents are now published electronically, and can be intended for publication on a number of different mediums. For example the same document can be published upon a variety of devices including any of the following: a web page, a WAP (wireless application protocol) telephone, a web enabled television, a personal digital assistant (PDA) (whether landscape, or portrait versions), a printer, etc. Each of these devices has considerably different display capabilities and as such the same document cannot be displayed on each of the devices without modification.

Data can be presented in any number of ways and still convey the same information to a reader. Thus, the content of a set of data is separate from the presentation of the document. Page description languages such as HTML are well known and allow a user to write a content document specifying how they wish the data to be display when the content document is rendered. However, this single content document therefore contains both the data content and also its format.

It is an object of the present invention to overcome, or at least reduce, the problems with the prior art.

According to a first aspect of the invention there is provided a method of generating data suitable for transmission to at least one data-receiving device, said method comprising the following steps:

i. specifying said data in a first and a second portion, said first portion being substantially independent of any formatting, and said second portion containing said formatting for said first portion specified in a platform independent manner;

ii. transforming said second portion, using a first transform, to generate a platform dependent portion containing said formatting specified in a platform dependent manner;

iii. combining said first portion with said platform dependent portion using a second transform to generate said data suitable for transmission to said at least one data-receiving device.

An advantage of such a method is that it allows a single first portion to be written, which will be can be combined with a plurality of second portions to make it suitable for transmitting to a number of different data-receiving devices. This is especially advantageous for first portions that are lengthy, and it is desired to display said first portion in substantially the same style throughout. (The skilled person will appreciate that the style comprises the formatting for the data at any one point; for example the font that is used, the justification, the line spacing, etc.) In such circumstances it is likely that the first portion will be substantially longer than the second portion and that therefore, much storage space will be saved because only a single copy of the first portion will be required, rather than a copy for each platform to which it is desired to transmit, as in the prior art. The method will still be advantageous in instances where there is less of a marked difference in the size of the first and second portions due to the space saving and time saved in being able to generate suitable for transmission to a number of platforms from a single first portion.

It will be appreciated that a transform may be any process that combines two or more items.

Preferably, the first portion comprises a portion of text, preferably containing at least one marker allowing said text to be identified. Ideally, said marker identifies a paragraph, or other section, such that paragraph, or section, specific formatting can be applied to said paragraph, or section. An advantage of such an arrangement is that it allows a variety of different formats to be applied to the data contained in the first portion. There may be a plurality of markers contained in the first portion.

Conveniently, the second portion contains at least one of the following items of formatting information: font type, font size, font colour, font weight (i.e. whether the font is to be displayed in italic, bold, normal, , etc.), justification, line spacing, character spacing, or any other item of formatting information.

Preferably, the first and second portions are held in separate files. Such an arrangement is convenient because it makes finding and editing the data more convenient.

Alternatively, it would be possible for the first and second portions to be held in the same file.

Preferably, the first transform accesses the file in which the second portion is stored and generates a third file. Further, the second transform may access both the file in which the first portion is stored and the third file in order to generate said data suitable for transmissions to said at least one data-receiving device, which may in turn be stored in a fourth file.

Conveniently, the first portion and/or the second portion are written in adevice independent language. In the most preferred embodiment the first and second portions are written in XML. An advantage of using XML is that it is a platform independent language, which is provided with a dedicated transform language.

The first and/or the second transform may be written in XSL. Use of XSL is particularly advantageous if the first and /or the second portions are written in XML due to the close ties between XSL and XML.

The method may comprise holding the data on a server and arranging said server to perform the method upon receipt of a request for said data. The request may be from a data-receiving device for data to be sent thereto, or may be from a first data-receiving device requesting that data should be sent to a second.

The method may comprise generating the said data suitable for transmission to said at least one data-receiving device when a request for said data is received. As such a fourth file containing said data suitable for transmission to said at least one data-receiving device may not be created, or one may be created when the request is received. Such an arrangement is advantageous because it reduces the amount of storage space that is required to store the data.

However, in an alternative embodiment said data suitable for transmission to said at least one data-receiving device may be generated in advance and stored for transmission to a data-receiving device. Therefore, should a fourth file containing the data suitable for transmission to said at least on data-receiving device be created it may be held on the server. Such an arrangement is itself advantageous because it is less intensive and

therefore, will require less processing power and may therefore run on less powerful hardware.

The method may be capable of generating said data suitable for transmission to any of the following data-receiving devices: a WAP enabled telephone, a web enabled television, a printer, a browser (for example MICROSOFT (R.T.M.) EXPLORER, or NETSCAPE NAVIGATOR (R.T.M.)), a PDA, etc.

According to a second aspect of the invention there is provided a computer readable medium holding a program arranged to run the method of the first aspect of the invention.

A computer readable medium may comprise any one of the following: a floppy disk, a CDROM, a DVD ROM/RAM, a ZIP disk, LS120 disk, any other suitable physical format, a transmitted signal, an internet download, etc.

According to a third aspect of the invention there is provided a data structure accessible by a processing apparatus for processing and subsequent transmission to a data-receiving device, comprising in combination:

a first portion containing data that it is desired to send to said data-receiving devices, held in a platform independent form, and being substantially free of any formatting information;

a second portion specifying how the first portion should be displayed on said data-receiving device, said second portion containing said formatting information for said first portion specified in a plaform independent manner.

Preferably, the first and/or second portions is written in a mark-up language. The mark-up language may be any language defined by one of the following: XML, SGML, or any other suitable mark-up language specification.

Conveniently, the first and second portions are held as separate files. The skilled person will appreciate that the first and second portions could be held as separate portions within the same file, but this is likely to be less convenient.

According to a fourth aspect of the invention there is provided a processing apparatus arranged to hold data intended for transmission to at least one data-receiving device, said data being held in at least a first, data-receiving device independent portion substantially independent of any formatting, and a second portion containing said formatting for said first portion specifying how said first portion should be displayed on a data-receiving device, said apparatus comprising processing circuitry including a transmitter and receiver, the receiver arranged to receive a data request and pass said request to said processing circuitry, on receipt of said data request said processing circuitry being arranged to combine an appropriate second portion for said data-receiving device to which data is to be sent with said first portion to generate a data-receiving device specific portion, and further being arranged to send said data-receiving device specific portion to said transmitter for transmission to said data-receiving device.

There now follows by way of example only a detailed description of the present invention with reference to the accompanying drawings of which:

Figure 1 schematically shows the architecture of a computer capable of acting as a server for this invention;

Figure 2 schematically shows how a document can be sent to a number of different devices; and

Figure 3 schematically shows the processes of the present invention.

This particular invention is applicable to distribute data electronically, and in particular via the World Wide Web, or in short the web. Such technology is well known. Generally the data to be distributed is held on a processing apparatus, or server 2, as shown in Figure 1, and can be requested by any number of devices that are capable of communicating with the server 2. Indeed, a first device can make a request for data to be sent to a second device.

In this embodiment the processing apparatus, or server 2, comprises a display 4, processing circuitry 6, a keyboard 8, and mouse 10. The processing circuitry 6 further comprises a processing unit 12, a hard drive 14, a video driver 16, memory 18 (RAM and ROM) and an I/O subsystem 20 which all communicate with one another, as is known in the art, via a system bus 22. The processing unit 12 comprises an INTEL PENTIUM series processor, running at typically between 900MHz and 1.7GHz.

As is known in the art the ROM portion of the memory 18 contains the Basic Input Output System (BIOS) that controls basic hardware functionality. The RAM portion of memory 18 is a volatile memory used to hold instructions that are being executed, such as program code, etc. The hard drive 14 is used as mass storage for programs and other data.

Other devices such as CDROMS, DVD ROMS, network cards, etc. could be coupled to the system bus 22 and allow for storage of data, communication with other computers over a network, etc.

The server 2 could have the architecture known as a PC, originally based on the IBM specification, but could equally have other architectures. The server could may be an APPLE, or may be a RISC system, and may run a variety of operating systems (perhaps HP-UX, LINUX, UNIX, MICROSOFT NT, AIX™, or the like).

In this embodiment data, in this case data is held on the server 2, which stores the data and distributes it on request to a requesting data-receiving device. The requesting data-receiving device can be any device that is capable of communicating with the server 2. When the server 2 receives a request from a data-receiving device or another device it will forward the requested data onto the appropriate data-receiving device, after generating data suitable for transmission to said at least one data-receiving device as described hereinafter. (The server 2 may generate the data suitable for transmission to said at least one data-receiving device in advance, or when the request for data is received). This arrangement is schematically represented in Figure 2, which shows a WAP enabled telephone 24, a printer 26 and a PC 28 in communication with the server via a network connection 30.

Data suitable for transmission to said at least one data-receiving device may be capable of being displayed on a variety of devices. For example a PC 28 may be programmed so that it can receive and correctly process data presented in WML format, which would generally be used for WAP enabled telephones. As such, reference to platform is intended to cover any device that is capable of receiving said data suitable for transmission to said at least one data-receiving device in any one particular format.

The code shown in appendix I is written in XML (eXtensible Mark-up Language), but could be equally stored using other suitable mark-up languages. XML requires pairs of tags to be placed within a document. Theses tags do not specify how the information should be presented, but specify the content of the information between the pairs of tags. The skilled person will fully understand XML, but a full description can be found at http://www.w3.org, and the brief description below will aid his/her understanding.

The skilled person will appreciate how an XML document is structured: written in words, or data sub-items, which are collected into data sub-item groups. The data sub-item groups can comprise sentences, paragraphs, or simply collections of words. The data sub-item groups, or even just data sub-items, are placed between pairs of tags.

The tags appear as follows: <variable>, and </variable>, with variable being any word, or character string acceptable according to the XML recommendation. Further, each data sub item group can be itself broken down into a number of sub-items. This structure is convenient and allows for easy manipulation and searching of the complete data item.

The code shown in appendices II and IV is written as an XSL transformation (XSLT). The skilled person will appreciate that *XSL* is a language for expressing stylesheets consisting of three parts: i. the XSL language for expressing XSL transformation of XML documents; ii. an XPATH language used by the XSL language for referring to parts of a document; and a vocabulary for specifying formatting semantics.

An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary. A fuller

description of XSL can be viewed on the web site of the World Wide Web consortium (http://www.w3.org).

Figure 3 shows one embodiment of how the present invention can be realised. A content document 300, or first portion of data, is written such that it contains the data content and does not contain any formatting information. The content document 300 contains a number of markers that identify paragraphs therein, and is held in a first file on the server 2.

A second, format independent properties, document 302, or second portion of data, contains the desired formatting for the content document written in a format (or device) independent manner. This format independent properties document 302 is held in a second file on the server 2. An example of such a document is provided in appendix I, and in this embodiment is written in XML. This format independent properties document 302 specifies the following parameters: fontfamily, i.e. the font in which the content will appear (serif); fontsize i.e. the size of the font in which the content will appear (medium); the font style, in this case normal; the weight of the font, in this case normal; the colour of the font, in this case black; the alignment of the font, in this case justified; and the spacing of the font, in this case 2em (a relative dimension that a renderer will calculate according to other parameters).

It will be apparent to the skilled person that any other property of a font could be specified by the format independent properties document 302. For example whether or not the font is italic, in bold, underlined, etc. could all be specified.

Further, although the format independent properties document shown in appendix II shows only one particular set of formatting, it would be possible for it to contain a number of different sets. As such, different

paragraphs, or other sections, within the content document 300 could have a different set of formatting applied thereto. The markers provided in the content document 300 allow the correct set of formatting to be applied to the correct paragraph, or other section.

A first transform process 302 processes the format independent properties document 304 to generate a format dependent properties document 306 therefrom, which is stored in a third file on the server 2. This format dependent properties document 306 is device dependent and a separate document must be generated for each device, or set of devices, on which it is desired to display information. In the present example the format independent properties document 302 is transformed into three separate format dependent property documents 306. Appendix II shows the following documents: an XSL stylesheet suitable for transforming into a WML document for transmission to a WAP enabled telephone 24, an XSL stylesheet suitable for transforming into an HTML document for transmission to a browser running on a computer 28; and an XSL stylesheet suitable for transforming into a XSL-FO document suitable to be rendered into a page description language and to be printed on a printer 26. Appendix III shows the results of these transform processes.

Once the format dependent properties document 306 has been generated for a particular platform it must be combined with the content document 300 before a deliverable content document 310 can be obtained, suitable for transmission to a data-receiving device.

This combining of the format dependent properties document 306 with the content document 300 is performed by a second process 308 that is defined by an appropriate XSL stylesheet. The resulting deliverable content is held in a fourth file on the server 2. Examples of such stylesheets are shown in appendix IV. Again, because in this embodiment

a document is being generated for a WAP telephone 24, a printer 26, and a browser running on a PC 28 three separate style sheets are required, and an example of each of these is shown in appendix IV.

The second process applies an appropriate set of formatting as defined in the format dependent properties document 306 with the appropriate paragraph, or section, defined by the markers in the content document 300.

Looking at the XSL stylesheet that generates the XSL code in appendix II, a portion 400 switches on the <FontFamily> tags provided in the format independent properties document 302 to generate a line of code 500 in the HTML format dependent properties document 306 specifying that the "serif" font should be used.

Next a portion 402 of the stylesheet for the first process switches on the <FontSize> tags within the XML format independent properties document 302 to determine the size of the font that should be used to display the document. The result of this portion can be seen at 502 in appendix III, in the XSL code for the format dependent properties document 306.

A portion of the code 404 within the XSL stylesheet determines the colour of the text in which the text will be displayed by using the <colour> tag within the format independent properties document 302, and generates the line 504 in appendix III. Thus, the colour that will be specified in the HTML deliverable content document 310 when it is generated will be black.

Further, a portion of the code 406 switches on the <FontWeight> tag within the format independent properties document 302 to generate a line

within the format dependent properties document 306 specifying whether the text should be bold, italic, underlined, etc. Because the format independent properties document 302 specifies that the FontWeight should be normal, no line is generated in the format dependent properties document 306.

Once the format dependent properties document 306 has been generated, as described above, it is combined with the content document 300, by the style sheet 308 shown in appendix IV. Looking at the XSL stylesheet to generate the HTML version, the line 550 imports the file "CurrentTextElementHTMLPropertyInstance.XSL", which is the format dependent properties document 306 generated by the first transform process.

Although not discussed in detail, the WML, and XSL-FO versions of the code shown in the appendices function in a similar manner. The skilled person will also appreciate that other transforms, and languages are possible, and that the three page description languages used herein are provided merely as examples.

In its broadest concept the invention may be considered as a method of generating data suitable for transmission to at least one data receiving device said method comprising the following steps: specifying the data in at least a first and a second portion, said first portion containing said data and being data-receiving device independent, and said second portion specifying how said first portion should be displayed on a data-receiving device; and combining said first and second portions into a data-receiving device specific document before generating said device specific data suitable for transmission to said data-receiving device.

APPENDIX I

Example format independent properties document 302

```
<?xml version="1.0" ?>
<TextElement>
  <PropertySets>
    <FontSet>
      <FontFamily>serif</FontFamily>
      <FontSize>medium</FontSize>
      <FontStyle>normal</FontStyle>
      <FontWeight>normal</FontWeight>
    </FontSet>
    <ColourSet>
      <Colour>black</Colour>
    </ColourSet>
    <AlignmentSet>
      <Align>justify</Align>
    </AlignmentSet>
    <SpacingSet>
      <SpaceBefore>2em</SpaceBefore >
    </SpacingSet>
  </PropertySets>
</TextElement>
```

Appendix II

XSL stylesheets for first transform process 304

HTML version

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
  <xsl:import href="ColourPropertySet.xsl"/>
  <xsl:template match="FontSet" >
    <xsl:element name="xsl:template" >
      <xsl:attribute name= "match" >Content </xsl:attribute >
    <xsl:apply-templates select="//FontSet/FontFamily"/>
    </xsl:element >
  </xsl:template >
  <xsl:template match="FontSet/FontFamily" >
    <xsl:element name="FONT" >
      <xsl:attribute name="face" >
      <xsl:value-of select="."/>
      </xsl:attribute >
      <xsl:apply-templates select="//FontSet/FontSize"/>
    </xsl:element >
  </xsl:template >
  <xsl:template match="FontSet/FontSize" >
    <xsl:variable name="fontsize" >
      <xsl:value-of select="."/>
    </xsl:variable >
    <xsl:element name="FONT" >
      <xsl:attribute name="size" >
        <xsl:choose >
          <xsl:when test="$fontsize='xx-large'" > +6 </xsl:when >
          <xsl:when test="$fontsize='x-large'" > +4 </xsl:when >
          <xsl:when test="$fontsize='large'" > +2 </xsl:when >
          <xsl:when test="$fontsize='normal'" > +0 </xsl:when >
          <xsl:when test="$fontsize='small'" > -2 </xsl:when >
          <xsl:when test="$fontsize='x-small'" > -4 </xsl:when >
          <xsl:when test="$fontsize='xx-small'" > -6 </xsl:when >
          <xsl:otherwise />
        </xsl:choose >
      </xsl:attribute >
      <xsl:apply-templates select="//ColourSet/Colour"/>
    </xsl:element >
  </xsl:template >
  <xsl:template match="FontSet/FontStyle" >
  <xsl:variable name="style" >
    <xsl:value-of select="."/>
  </xsl:variable >
  <xsl:choose >
    <xsl:when test= "$style='italic'" >
      <xsl:element name="I" >
        <xsl:apply-templates select= "//FontSet/FontWeight"/>
      </xsl:element >
    </xsl:when >
```

Selects font in which, text will be displayed. In this case "serif"

400

Selects size of font in which, text will be displayed. In this case "+4"

402

Selects colour in which, text will be displayed. In this case "black"

404

```
        <xsl:when test= "$style='oblique'" >
          <xsl:element name="I" >
            <xsl:apply-templates select="//FontSet/FontWeight"/ >
          </xsl:element >
        </xsl:when >
        <xsl:when test= "$style='normal'" >
        <xsl:apply-templates select= "//FontSet/FontWeight"/ >
        </xsl:when >
        <xsl:otherwise/ >
      </xsl:choose >
    </xsl:template >
    <xsl:template match="FontSet/FontWeight" >
      <xsl:variable name="weight" select="."/ >
      <xsl:choose >
        <xsl:when test="$weight='bold'" >
          <xsl:element name="B" >
            <xsl:element name= "xsl:apply-templates"/ >
          </xsl:element >
        </xsl:when >
        <xsl:otherwise >
          <xsl:element name="xsl:apply-templates"/ >
        </xsl:otherwise >
      </xsl:choose >
    </xsl:template >
</xsl:stylesheet >
```

Selects font weight in which, text will be displayed. In this case "normal"

406

WML version

```
xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<xsl:template match ="FontSet" >
    <xsl:element name="xsl:template" >
        <xsl:attribute name= "match" >Content</xsl:attribute >
        <xsl:apply-templates select="// FontSet/FontFamily" / >
    </xsl:element >
</xsl:template >
<xsl:template match= "FontSet/FontFamily" >
    <xsl:apply-templates select= "//FontSet/FontSize" / >
</xsl :template >
<xsl:template match=" FontSet/FontSize" >
    <xsl:variable name= "fontsize" >
        <xsl:value-of select="."/ >
    </xsl:variable >
    <xsl:choose >
        <xsl:when test= "$fontsize='xx-large'" >
          <xsl:element name="big" >
            <xsl:apply-templates select= "//FontSet/FontStyle" / >
          </xsl:element >
        </xsl:when >
        <xsl:when test= "$fontsize='x-large"' >
          <xsl:element name="big" >
            <xsl:apply-templates select= "//FontSet/FontStyle" / >
          </xsl:element >
        </xsl:when >
```

```
<xsl:when test= "$fontsize='large"' >
  <xsl:element name="big" >
    <xsl:apply-templates select= "//FontSet/FontStyle" /> _ </xsl:element>
</xsl:when >
<xsl:when test= "$fontsize='medium"' >
    <xsl:apply-templates select= "//FontSet/FontStyle" />
</xsl:when >
<xsl:when test= "$fontsize='small"' >
  <xsl:element name="small" >
    <xsl:apply-templates select= "//FontSet/FontStyle" />
  </xsl:element >
</xsl:when >
<xsl:when test= "$fontsize='x-small1"' >
  <xsl:element name="small" >
    <xsl:apply-templates select= "//FontSet/FontStyle" />
  </xsl:element >
</xsl :when >
<xsl:when test= "$fontsize='xx-small"' >
  <xsl:element name="small" >
    <xsl:apply-templates select= "//FontSet/FontStyle" />
  </xsl : element >
</xsl:when >
<xsl:otherwise />
</xsl:choose >
</xsl:template >
  <xsl:template match=" FontSet/FontStyle" >
      <xsl:variable name="style" >
          <xsl:value-of select="." />
      </xsl:variable >
      <xsl:choose >
          <xsl:when test= "$style='italic"' >
              <xsl:element name="i" >
                  <xsl:apply-templates select="// FontSet/FontWeight" />
              </xsl:element >
          </xsl:when >
          <xsl:when test= "$style='oblique"' >
              <xsl:element name="i" >
                  <xsl:apply-templates select= "//FontSet/FontWeight" />
              </xsl:element >
          </xsl:when >
          <xsl:when test= "$style='normal'" >
              <xsl:apply-templates select= "//FontSet/FontWeight" />
          </xsl:when >
      <xsl:otherwise />
  </xsl:choose >
</xsl:template >
<xsl:template match= "FontSet/FontWeight" >
  <xsl:variable name="weight" select="." />
  <xsl:choose >
      <xsl:when test= "$weight='bold"' >
          <xsl:element name="b" >
              <xsl:element name= "xsl:apply-templates" />
          </xsl:element >
      </xsl:when >
      <xsl:when test= "$weight='normal"' >
```

```
                <xsl:element name = "xsl:apply-templates" />
            </xsl:when>
          <xsl:otherwise />
        </xsl:choose>
    </xsl:template>
</xsl:stylesheet>
```

xsl - fo version

```
xsl:stylesheet version = "1.0"
 xmlns:xsl = "http://www.w3.org/1999/XSL/Transform" >
<xsl:template match = " FontSet" >
    <xsl:apply-templates />
</xsl :template >
<xsl:template match = " FontSet/FontFamily" >
    <xsl:element name = "xsl:attribute" >
       <xsl:attribute name = "name" > font-family </xsl:attribute >
       <xsl:apply-templates />
    </xsl:element >
    </xsl:template >
    <xsl:template match = " FontSet/FontSize" >
       <xsl:element name = "xsl:attribute" >
          <xsl:attribute name = "name" > font-size </xsl:attribute >
          <xsl:apply-templates />
        </xsl:element >
    </xsl:template >
    <xsl:template match = " FontSet/FontStyle" >
        <xsl:element name = "xsl:attribute" >
          <xsl:attribute name = "name" > font-style </xsl:attribute >
          <xsl:apply-templates />
       </xsl:element >
    </xsl:template >
    <xsl:template match = " FontWeight" >
        <xsl:element name = "xsl:attribute" >
          <xsl:attribute name = "name" > font-weight </xsl:attribute >
          <xsl:apply-templates />
       </xsl:element >
    </xsl:template >
</xsl:stylesheet >
```

Appendix III

Format dependent properties document 306

HTML

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="Content">
    <FONT face="serif">
      <FONT size="+0">
        <FONT color="black">
          <xsl:apply-templates />
        </FONT>
      </FONT>
    </FONT>
  </xsl:template>
  <xsl:attribute-set name="text-style">
    <xsl:attribute name="align">justify</xsl:attribute>
  </xsl:attribute-set>
</xsl:stylesheet>
```

XSL-FO version

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:attribute-set name="text-style">
  <xsl:attribute name="font-family">serif</xsl:attribute>
  <xsl:attribute name="font-size">medium</xsl:attribute>
  <xsl:attribute name="font-style">normal</xsl:attribute>
  <xsl:attribute name="font-weight">normal</xsl:attribute>
  <xsl:attribute name="color">black</xsl:attribute>
  <xsl:attribute name="text-align">justify</xsl:attribute>
  <xsl:attribute name="space-before.optimum">2em</xsl:attribute>
  </xsl:attribute-set>
</xsl:stylesheet>
```

WML version

```
< xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform" >
   < xsl:template match = "Content" >
      < xsl:apply-templates / >
   < /xsl:template >
   < xsl:attribute-set name = "text-style" >
      < xsl:attribute name = "align" > left < /xsl:attribute >
   < /xsl:attribute-set >
< /xsl : stylesheet >
```

Appendix IV

XSL style sheet for process 2 combining format dependent properties document 306 with the content document 300

HTML version

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
  <!-- Imports all property sets descriptions -->
  <xsl:import                                                    550
      href="file:///C:/SysArch/Instances/CurrentTextElementHTMLPropertyIns
  tance.xSL" >
  <!-- end of imports -->
  <xsl:output method="xml" indent="yes" media-type="text/xsl"/>
  <xsl:template match="TextElement/Content" >
    <P xsl:use-attribute-sets="text-style" >
      <xsl:apply-imports/>
    </P>
  </xsl:template>
</xsl:stylesheet>
```

WML version

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
  <xsl:output method="xml" indent="yes" media-type="text/xsl"/>
  <! -- Imports all property sets descriptions -->
  <xsl: import
    href="file:///C:/SysArch/Instances/CurrentTextElementWMLPropertyInstance
  .xsl" >
    <!-- end of imports -->
    <xsl:template match= "TextElement" >
      <xsl:apply-templates />
    </xsl:template>
    <xsl:template match ="Content" >
      <p xsl:use-attribute-sets="text-style" >
        <xsl:apply-imports />
      </p>
    </xsl:template>
  </xsl:stylesheet>
```

XSL - FO version

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
```

```
xmIns:xsl = "http://www.w3.org/1999/XSL/Transform"
xmlns:fo = "http://www.w3.org/1999/XSL/Format" >
<!-- Imports all property sets descriptions -- >
<xsl: import
   href = "file:///C:/SysArch/Instances/CurrentTextElementFOPropertyInstance.
   xsl"
<!-- end of imports -- >
<xsl:output method = "xml" indent = "yes" media -type = "text/xsl"/ >
<xsl:template match = "Content" >
   <fo:block xsl:use-attribute-sets = "text-style" >
      <xsl:apply-templates / >
   </fo: block >
</xsl :template >
</xsl:stylesheet >
```

CLAIMS

1. A method of generating data suitable for transmission to at least one data-receiving device, said method comprising the following steps:

i. specifying said data in a first and a second portion, said first portion being substantially independent of any formatting, and said second portion containing said formatting for said first portion specified in a platform independent manner;

ii. transforming said second portion, using a first transform, to generate a platform dependent portion containing said formatting specified in a platform dependent manner;

iii. combining said first portion with said platform dependent portion using a second transform to generate said data suitable for transmission to said at least one data-receiving device.

2. A method according to claim 1 wherein the first portion comprises a portion of text.

3. A method according to claim 2 wherein the method is arranged such that the first portion contains at least one marker allowing said text to be identified.

4. A method according to claim 3 wherein said marker identifies a paragraph such that said paragraph specific formatting can be applied to said paragraph.

5. A method according to claim 1 wherein the method arranges said second portion such that it contains at least one of the following items of formatting information: font type, font size, font colour, font weight (i.e. whether the font is to be displayed in italic, bold, normal,), justification, line spacing, character spacing.

6. A method according to clam 1 wherein the first and second portions are held in separate files.

7. A method according to claim 1 wherein said first portion is written in a device independent language.

8. A method according to claim 7 wherein said device independent language is XML.

9. A method according to claim 1 wherein said second portion is a device independent language.

10. A method according to claim 9 wherein said device independent language is XML.

11. A method according to claim 1 wherein said first transform is written in XSL.

12. A method according to claim 1 wherein said second transform is written in XSL.

13. A method according to claim 1 comprising holding said data on a server and arranging said server to perform the method upon receipt of a request for said data.

14. A method according to claim 13 wherein said request is from a data-receiving device for data to be sent thereto.

15. A method according to claim 13 wherein said request is from a first data-receiving device requesting that data should be sent to a second.

16. A method according to said data suitable for transmission to said at least one data-receiving device is generated in advance and stored for transmission to a data-receiving device.

17. A method according to claim 1 comprising generating said data suitable for transmission to any of the following data-receiving devices: a WAP enabled telephone, a web enabled television, a printer, a browser (for example MICROSOFT EXPLORER (R.T.M.), or NETSCAPE NAVIGATOR (R.T.M.)), a PDA.

18. A computer readable medium holding a program arranged to run the method of claim 1.

19. A data structure accessible by a processing apparatus for processing and subsequent transmission to a data-receiving device, comprising in combination:

a first portion containing data that it is desired to send to said data-receiving devices, held in a platform independent form, and being substantially free of any formatting information;

a second portion specifying how the first portion should be displayed on said data-receiving device, said second portion containing said formatting information for said first portion specified in a plaform independent manner.

20. A structure according to claim 19 comprising writing said first portion in a mark-up language.

21. A structure according to claim 20 wherein said mark-up language is XML.

22. A structure according to claim 21 comprising writing said second portion in a mark-up language.

23. A structure according to claim 22 wherein said mark-up language is XML.

24. A structure according to claim 19 comprising arranging said first and second portions in separate files.

25. A structure according to claim 19 comprising writing the first transform in XSL.

26. A structure according to claim 19 comprising writing the second transform in XSL.

27. A processing apparatus arranged to hold data intended for transmission to at least one data-receiving device, said data being held in at least a first, data-receiving device independent portion substantially independent of any formatting, and a second portion containing said formatting for said first portion specifying how said first portion should be displayed on a data-receiving device, said apparatus comprising processing circuitry including a transmitter and receiver, the receiver arranged to receive a data request and pass said request to said processing circuitry, on receipt of said data request said processing circuitry being arranged to combine an appropriate second portion for said data-receiving

device to which data is to be sent with said first portion to generate a data-receiving device specific portion, and further being arranged to send said data-receiving device specific portion to said transmitter for transmission to said data-receiving device.

Application No: GB 0127722.7 28
Claims searched: 1-27

Examiner: Ben Micklewright
Date of search: 18 June 2002

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.T): G4A (AUDB, AMX)

Int Cl (Ed.7): G06F (17/21 17/30)

Other: Online: WPI, EPODOC, PAJ, INSPEC, XPESP, IBM TDB, COMPUTER, Selected Internet sites

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
X,P	WO 01/90873 A1	(2ROAM) See e.g. pages 12-15	1-27
A	WO 01/63481 A2	(SUN)	-
X	WO 00/23912 A1	(OCE-USA) See e.g. pages 2,9	1-4,6,7,9, 13-16, 18,24,27
X	WO 00/03332 A1	(NETPOST) See e.g. page 4	1-27
X	US 6101513	(SHAKIB) See e.g. column 2	1-4,6,7,9, 13-16, 18,24,27
X	http://www.oasis-open.org/cover/rml.html, "The XML Voer Pages. Relational Markup Language (RML)", R Cover, Last modified 2 June 2001		1-27

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.