WEB SERVER

This invention relates to web servers and in particular but not exclusively to a web server for responding to request messages received via the internet from remote user devices and to a web server capable of responding with web page code tailored to meet the capabilities of the remote user device.

The use of the internet has recently flourished to the extent where almost every business concern has its own website to provide information and online purchasing facilities. The majority of such websites have in the past been accessed by remote user devices which are PCs (Personal Computers) using one of a relatively small number of available browser applications so that, by and large, the websites were required to output web page code in a single format for viewing at the user device.

More recently, there has been a proliferation in new internet enabled devices such as PDAs (Personal Digital Assistants), mobile telephones with WAP (Wireless Application Protocol) facilities, IDTV (Interactive Digital Television), information kiosks, games, consoles and home appliances. The capabilities of these devices vary considerably from device to device, both in their display function in terms of screen size and colour processing ability, bandwidth and available memory as well as variables associated with communication including

image format, communication protocol and markup language.

In order to enable users of these different devices to access their websites, many businesses have resorted to building separate web servers to serve each device type. This however creates data management problems where a customer may wish to have the option of using different devices to communicate with the merchant using the same account details for example, the merchant then requiring a proliferation of different web server applications to be managed and updated when required with new or additional material.

An alternative approach has been to provide a separate port of a website for access by an alternative type of user device, the separate port obtaining the required web page code by translating the original PC based web page code, typically in HyperText Markup Language (HTML) format, to other device formats such as Wireless Markup Language (WML) used for WAP telephones. Such solutions however generally have failed to deliver optimum presentation to the user of the device. The translating process (often referred to as "transcoding") is difficult to engineer. For example, it is necessary to provide appropriate selection of information which is to be omitted from display in the device having lower capabilities than the PC. This difficulty is compounded when it is necessary to update the original PC web page

code because the transcoding process must also be re-engineered.

It has also been proposed that the protocols used for accessing web page information over the internet should be unified by the adoption of standards in order to overcome problems resulting from remote user devices having different capabilities. The use of Extensible Markup Language (XML) as a format for web page code has been proposed. XML is a mark-up language for defining content and can be used with XSL (Extensible Stylesheets Language) used to transform XML documents with style sheets at the point of presentation. As yet however, such standards have not been adopted and this solution is not seen as optimum for all device types.

The present invention seeks to provide an improved web server with multi-channel capabilities able to serve different types of remote user device.

According to the present invention there is disclosed an apparatus for responding to a request message from a remote user device for web page information by generating web page code representative of one or more web pages to be displayed by the user device and outputting a response message comprising the web page code.

Preferably the apparatus comprises extracting means for extracting from the request message a device type

identifier identifying the remote user device as being one of a set of possible device types having different capabilities.

The apparatus preferably has a processor for operating a code generating engine to generate the web page code, first memory means for storing the web page information as a set of instructions, and second memory means for storing device dependent information for each of the different device types for tailoring the web page code to the capabilities of the device type.

The code generating engine preferably comprises interpreting means for interpreting the instructions with reference to selected device dependent information corresponding to the device type identifier, the code generating means thereby being operable to generate the web page code in a form in which the web page code is tailored to the capabilities of the remote user device.

Preferred embodiments of the present invention will now be described by way of example only and with reference to the accompanying drawings of which:

Figure 1 is a schematic overview of connection between a web server and remote user devices;

Figure 2A is a schematic diagram of the web server of Figure 1;

Figure 2B is a functional diagram for illustrating signal flow in the web server of Figure 2A;

Figure 3 is a schematic diagram of policy tables for use with the web server of Figure 2;

Figure 4 is a schematic diagram illustrating operation of the code generating engine;

Figure 5A illustrates a layout for a PC;

Figure 5B illustrates a layout for a television;

Figure 5C illustrates a layout for a PDA;

Figure 5D illustrates a layout for a WAP telephone;

Figure 5E illustrates layout for a device having a large screen;

Figure 5F illustrates the display of fragments in a device having a small screen;

Figure 5G illustrates a multi-menu structure in a large screen device;

Figure 5H illustrates the relationship between control segments and content segments in the layout of Figure 5G;

Figure 6 illustrates the hierarchy of a device policy table;

Figure 7 illustrates schematically the software provided to the web server;

Figure 8 illustrates schematically the hardware required by the web server;

Figure 9 illustrates schematically the separation of content code and presentation of information for use by the code generating engine;

Figure 10 illustrates schematically a system comprising a production server and a development server;

Figure 11 illustrates the exchange of signals in the system of Figure 10 when a new type of user device is encountered for the first time;

Figure 12 illustrates schematically a server having a cache memory;

Figure 13 is a flowchart illustrating the method of operation of a gateway server of Figure 12;

Figure 14 is a flowchart illustrating operation of a dynamic web page server of Figure 12;

Figure 15 is a schematic diagram illustrating a component logic software module;

Figure 16 is a flowchart illustrating operation of code generation using the component logic software module;

Figure 17 is a diagram illustrating the hierarchy of data objects which are different versions of data referenced by a single component name;

Figure 18 illustrates the hierarchy of text objects with different text code sizes and languages;

Figure 19 illustrates the signal flow in a web server in which pane dissection occurs;

Figure 20 is a flowchart illustrating pane dissection;

Figure 21 illustrates the dissection of a pane into

shards;

Figure 22 illustrates use of a user session memory during form filling;

Figure 23 illustrates the dissection of a pane used in form filling;

Figure 24 illustrates the function of the graphical user interface; and

Figure 25 is a flowchart illustrating a method of file storage.

Figure 1 illustrates a web server 1 connected to remote user devices 2 via the internet 3. The remote user devices are illustrated to be a PC, television, PDA, and a mobile telephone with WAP facility but can include any number of different devices with internet capability. The web server 1 is provided with a web server GUI (Graphical User Interface) 4 enabling an operator of the web server to author documents and generally control and maintain operation of the web server. The web server 1 is also connected via the internet 3 to a service centre 5 which acquires information concerning any new types of user device and provides the web server 1 with consequential software revision when required.

Figure 2A is a simplified schematic diagram of the web server 1 of Figure 1. The web server 1 includes a processor 20 which operates a code generating engine 21 in the form of an application which is run by the

processor on demand in order to generate web page code. The code generating engine 21 in this embodiment is written in the Java programming language and during operation of the web server exists in the working memory of the processor 20 in compiled form as byte codes and is implementable within a Java run time environment.

A memory 22 stores content code 23 which defines web page information in the form of documents written in a mark up language which in the present example is JSP1.1 (Java Server Pages). These documents are typically authored by the operator of the web server 1 using the graphical user interface 4. The memory 22 also stores operational data 24 which includes for example data which may need to be incorporated into web page code generated at run time in response to specific queries originating from the user device 2 and data acquired by monitoring usage of the web server. Data objects referenced by the content code 23 are stored in a database 19. These data objects may include files containing images or text and other forms of data objects which an author may wish to include in the web page code to be generated.

Device dependent information 25 is also stored in memory 22 and consists primarily of policy tables described below. The memory 22 also stores program code 26 which may be accessed by the processor 20 during operation of the server 1 or used to retrieve software

such as the code generating engine 21 at the time of startup. The program code 26 also includes code defining a set of tags used in a mark up language in which the documents are authored as described below.

An interface 27 communicates between a databus 28 serving the processor 20 and memory 22 and external connection to the internet 3 for receiving request messages and transmitting response messages.

A device identification engine 29 is provided for extracting from received messages a device type identifier for identifying the remote user device 2 by type. The device identification engine 29 in the present example is implemented in software by the processor 20. In this example, a request message received via the internet contains a HyperText Transfer Protocol (HTTP) header which includes an ID string declaring the name of the user agent which the originating device has used when generating the request message. The name of the user agent software defines for example the browser version and markup language capability of the device and is used by the device identification engine 29 to identify the originating device using the device type identifier which may for example be stored as a lookup table.

The HTTP header may also yield information contained in a cookie regarding details of the user who has originated the request message. Such cookies may contain

other data used for personalisation of the response made by the web server 1.

Figure 2B is a further illustration of the web server 1 showing functional elements in a manner which illustrates the signal flow. A front end processor 300 provides a communications interface for request messages received from the internet 3 and response messages transmitted to the user device 2 via the internet. The front end processor 300 sends URL information 301 to a web application processor 302 which determines from the URL the appropriate information to be included in a web page requested by the user. The web application processor 302 has access to a database 303 which may be local to the processor or may be at a remote location which is accessible via an appropriate communications link.

The web application processor 302 outputs a document 304 comprising instructions for generating web page code. The document 304 is not itself capable of being interpreted by a browser of a user device and therefore does not constitute "web page code" for the purpose of the present description.

The document 304 is input to the code generating engine 21 which generates web page code with reference to content data structure 305 and the set of policy tables 40. The output of the code generating engine is

a web page code document 306 which is now in a form which is capable of being interpreted by the browser of the user device 2. The web page code document 306 is received by the front end processor 300 and packaged as a response message transmitted via the internet 3 through the user device 2.

To enable the code generating engine 21 to select the appropriate data from the content data structure 305 and to access the appropriate policy tables 40, it also requires the input of a device type identifier 45 which is provided by device identification engine 29 in response to receiving a message header signal 307 from the front end processor 300, this signal comprising information extracted from the message header of the request message received from the user device.

Figure 3 illustrates schematically the policy tables 40 which constitute the device dependent information 25 of Figure 2A.

In the present context, the term "policy" is used to signify information to be utilised by the code generating engine 21 and which can be defined differently for different device types, either out of necessity because of different technical capabilities of different device types or as a result of choice exercised by the author of the content code 23.

The policy tables 40 include a device policy table

30 which contains a number of fields relating to different technical aspects of the device type. By way of example, table 1 below lists a small number of device policies that can be associated with the device type and which each correspond to one field (i.e. row) of the table. Values are illustrated for a single device type (i.e. a single column). The device policy table 30 in practice will include a large number of different fields and values for multiple device types using a hierarchical table structure to represent the table in compact form..

TABLE 1 - Part of Device Policy Table

POLICY	VALUES
Supports GIF format images	true/false
Supports JPEG format images	true/false
Supports MP3 format audio	true/false
Supports WAV format audio	true/false
Can report global position (GPS)	true/false

By accessing the device policy table 30, the code generating engine 21 thereby is able to exercise decision making regarding choice of code generation to suit the technical capabilities of the user device 2. For example, if an image is to be included as an image file in the web page code, it is necessary for the code generating engine 21 to have knowledge of the form of

image data compression available to the user device 2 from which the request message originated. The device policy table 30 includes fields indicating whether GIF or JPEG format images can be decompressed by the device. Similarly, audio files may be compressed in a number of different forms including MP3 format and the device policy table 30 also includes a field indicating whether the device supports MP3.

The protocol policy table 31 is responsible for mapping equivalences between device output protocols, which are generally termed markup languages. Examples are WML, HTML, Handheld Device Markup Language (HDML) and compact HTML (cHTML). Table 2 is an example of six of the fields in the protocol policy table 31 for a single one of the device types.

TABLE 2 - Part of Protocol Policy Table

PROTOCOL	ELEMENT
HTML Version 4.0	<thead>
HTML Version 3.2	<thead>
HTML i-Mode	*no equivalent*
XHTML Basic	<thead>
WML Version 1.1	<thead>
HDML Version 3	*no equivalent*

A style policy table 32 contains a number of fields which define parameters such as font family, font size, font weight and colour of headings, image borders and paragraph background as illustrated in Table 3 which shows part of the style policy table for a single one of the device types. The code generating engine 21 is thereby able to generate code appropriate to the device type, in accordance with a predetermined decision which may for example reflect an agreed style adopted as a general trading style by a particular business entity.

The way in which stylistic information is handled varies between different devices, and indeed between different device output protocols. For some, style information must be included within the generated web page code. For others, style information must be provided in separately generated code, often called a "style sheet".

TABLE 3 - Part of Style Policy Table

POLICY	VALUES
Heading 1 font family	name of a font family, e.g. Times
Heading 1 font size	font size, e.g. 20
Heading 1 font weight	font weight, e.g. bold
Heading 1 colour	colour
Image border colour	colour for the border of images
Paragraph background colour	colour for the background of paragraphs

A themes policy table 33 similarly contains fields relating to decorative embellishments and logos adopted by business entities for their web pages and enables the author of the themes policy to tailor the decorative features and aspects of logo such as size for each of the device types.

A layouts policy table 35 contains fields relating to the layout of elements comprising text, logos, images etc at the point of rendering the image in the user device in order to take account for example of different screen shape and size for different device types.

A dynamic policy table 36 contains fields allowing code for different device types to be tailored according to time varying parameters. As an example, a mobile telephone having WAP facilities may have a bandwidth for

receiving response messages which varies with time according of the strength of signal available via the cellular mobile telephone network used by the telephone to connect to the internet. The dynamic policy may be configured to generate code requiring low bandwidth under low bandwidth conditions and higher bandwidth under better conditions. The amount of image data for example may be regulated in order to control the bandwidth required to carry the response message. Simply by omitting a logo or other image, the speed with which the response message can be downloaded to the device can be maintained under poor signal strength conditions.

A component policy table 37 is also provided the purpose of which is to determine selection of data objects as described below.

Figure 4 illustrates schematically the manner in which the code generating engine 21 makes use of the data in the policy tables 40. At startup, the processor 20 retrieves a startup program from the stored program code 26 and runs the startup program to assimilate the information in the policy tables 40 into a series of Java objects (Java beans) referred to herein as policy objects 41 which are then made available within a run time environment 46 of the code generating engine 21. The run time environment is provided by an appropriate Java Virtual Machine. The policy objects 41 each deal with

specific aspects of code generation and contain logic and data objects configured such that each policy object has knowledge of all of the relevant information available from the policy tables 40 for each device type. One policy object 41 for example may be concerned with image format capability whereas another policy object may be concerned with choice of markup language for the output code. The code generating engine 21 at run time processes a compiled version of a document of the content code 23, the code consisting of a series of instructions 42, each instruction including one of a set of markup tags 43.

In Figure 4, tag 43 is illustrated as being one of a number of custom or smart tags which, as represented schematically in Figure 4, refers when processed to a set of classes 44 which have been written to carry out the required function of the tags. These classes 44 in turn refer to one or more of the policy objects 41 whenever a program decision is to be taken which is dependent upon the device type on the basis of data contained in the policy tables 40.

Examples of custom tags 43 provided in accordance with the present embodiment are summarised in tables 4 and 5. All of the tags utilised in defining the content code 23 are device aware in the sense that they are implemented as JSP custom tags and are associated with

Java classes which access Java beans (policy objects 41) to make device dependent decisions. The more primitive tags are referred to here as simple in-line tags as shown in Table 4. More complex tags are referred to as major block tags in Table 5. The major tags typically translate into more than one tag in the markup language of the resulting generated code, for example HTML.

TABLE 4

Simple In-Line Tags	
cite	Causes text to be displayed as a citation
code	Formats text suitable for illustrating code samples
em	Causes text to be displayed emphasized
hr	Displays a horizontal rule

TABLE 5

Major Block Tags	
anchor	Link to another canvas, page or section
canvas	Define the layout and theme to be applied to this canvas
h1 to h6	Define headings at 6 different levels
form	Define an input form
textinput (part of form)	Define a text input field for a form
select (part of form)	Define a selection field for a form
option (part of form)	Define an option within a selection field for a form
image	Define an image
logo	Define a logo
menu	Define a menu
menuitem (part of menu)	Define one item within a menu
shopcart	A basic shopping cart
table	Define a table

The device identification engine 29 operated by processor 20 extracts from a header of the received request message from a remote user device 2 information which identifies the device type and outputs a device type identifier 45 which is input to the code generating engine 21 to enable the set of classes 44 to extract the required information from the policy objects 41.

An example of a document is given at Appendix 1 and comprises a set of instructions 42 including tags 43 to constitute the content code 23 in the markup language

operated upon by the code generating engine 21. At line 2 for example, the tag <vt:canvas> is used in defining the theme and layout for the web page to be created. Between the opening and closing canvas tags (occurring at the second line and last line) is a list of components which are also implemented using custom tags 43. The components reference panes in the layout, the term "pane" being used to signify one of a series of (generally) rectangular regions into which a display screen is to be divided when the web page is presented to the user on the user device 2. A collection of panes and their relative positions defines the layout for the page. The components can be used to define where in the rendered page a given component's output will appear, or alternatively to determine whether or not a given component is to be rendered when the web page is displayed for example on a user device having limited screen size. In this way, layouts can be used to limit the amount of information which appears on the display screen of certain device types.

Figure 5A for example illustrates a typical layout of panes 50 for display on a PC and including panes which respectively include a logo, a large colour logo, a site menu, a greeting, first, second and third promotions, and a news ticker providing continuous news information.

The layout corresponding to the same content document when generated for display on a television screen is shown in Figure 5B and consists of a reduced number of panes containing respectively a large colour logo, a greeting, a promotions menu, a site menu and a picture.

Figure 5C illustrates the corresponding rendered screen resulting from code generated based on the same content document when tailored for a PDA. The number of panes is considerably reduced during code generation to meet the limited capabilities of the PDA and includes panes which respectively contain a medium sized black and white logo, a greeting, a promotions menu and a site menu.

Figure 5D illustrates the layout resulting from code generation based on the same document when tailored to a WAP telephone. Instead of a single web page, the content is reduced to a set of fragments, often called decks, each comprising a single pane accessible via a menu deck 50, the set of decks further including single pane decks 51, 52 and 53 for promotions 1, 2 and 3 respectively. "Deck" is a term used in the WAP standards to indicate the units in which information is transmitted to the wireless device.

Appendix 2 illustrates the web page code generated

using the content code of Appendix 1 where the device type corresponds to a PC having a browser using the HTML protocol.

Appendix 3 illustrates the corresponding code generated when output to a device type corresponding to a telephone with WAP capabilities using WML protocol. Appendix 3 contains only one deck of a set of decks contained in the generated code, the user being required to select successive decks in order to view in turn a number of successive displays of the web page information. The fragmentation of the response message code into decks is illustrated in Figure 5D for example.

Appendix 4 illustrates a further deck resulting from the WML code for the WAP telephone, this time the deck corresponding to the menu deck 50 of Figure 5D in that it contains a list of stories which can be viewed in successive decks by the user.

Appendix 5 illustrates the deck created for one of the stories.

In the above example, the code generating engine 21 is capable of processing the same content code 23 to produce radically different web page code output corresponding to different device types. The first device type results in the code of Appendix 2 and the second device type which has lower capabilities results

in the code of Appendices 3, 4 and 5.

Figures 5E and 5F illustrate the manner in which the provision of different layout definitions for two different remote user devices 2 enables the entire content of a web page to be delivered to devices with dissimilar capabilities.

In Figure 5E, the layout for a device having a large screen has the effect of displaying the entire content in a single screen, portions of the content being contained in respective panes A, B, C, D, E and F. When formatted for the device with a small screen, only one of the panes A, B, C, D, E and F is displayed at any given time, the content thereby being fragmented into fragments A, B, C, D, E and F as illustrated in Figure 5F. Fragment A is designated as a root fragment via which the remaining fragments may be accessed using a menu.

For the device with small screen capability, the fragment may need to be re-shaped or the content presented differently so that it may for example be necessary for the user to scroll through text to see the entire contents of a given fragment containing text.

A multi-menu structure may alternatively be utilised as illustrated in Figures 5G and 5H. Instead of defining the layout in terms of panes, the layout is defined in

terms of a series of segments defining areas of the display and designated as either control segments or content segments. In Figure 5G, a control segment 55 contains a menu bar 56 composes of a series of icons which can be user selected.

A content segment 57 defines a further area of the display and itself contains a further content segment 58. Content segment 57 also contains a further control segment 59 which contains a second menu bar 510.

In this example, selection using the first menu bar 56 calls for display within control segment 59 of a selected one of a set of second menu bars 510 as illustrated in Figure 5H. Each of the second menu bars 510 allows selection from a respective set of content items 511.

The example of Figures 5G and 5H illustrates an alternative method of page description in which a montage tag defines page layout in terms of a series of segments which may be content segments or control segments. As shown in this example, it is possible for a content segment to contain further segments including both control and content segments.

Figure 5G corresponds to the web page viewed on a large screen device. When translated to a small screen device, the user might for example see only one segment

at any time, initially viewing the first control segment 55 to allow selection of one of the second menu bars 510, and subsequent selection of one of the content items 511.

The resulting layout of Figure 5G can be implemented using the inline frames feature of HTML.

The above examples of Figures 5A to 5H take two extremes of display capabilities of user devices. For devices of intermediate capabilities, alternative layouts showing perhaps two or three segments or panes may be appropriate. It may also be appropriate to modify the content within each pane or segment according to the capabilities of the device. For example, a PC is generally capable of a higher standard of image presentation including colour and moving images whereas a WAP phone may not have colour facility and may have limited screen resolution and refresh rate. This requires that for each pane or segment the content must be stored in a number of different forms appropriate to different devices.

The policy tables 40 are each arranged to have a hierarchical structure such as the structure illustrated for example in Figure 6 which shows the hierarchy of the device policy table 30. Characteristics of a notional root device type define an upper level 61 of the hierarchy. This root type may be regarded as

representing default values for the device policy table 30. A second layer 62 defines device types which differ from the root device type in a number of features. The device policy table will therefore include entries (columns) for those fields in which the parameters describing the root device type are not inherited. The third layer 63 of the hierarchy includes sub-divisions of the general device types of the first layer, for example a number of different PDA device types are illustrated. Each of these device types will inherit many of the parameters describing the capabilities of the PDA in the first layer but will differ in a number of respects, thereby requiring entries in respective columns of the device policy table for those fields which contain parameters not inherited.

Subsequent layers 64 and 65 of the hierarchy define further sub-divisions of device type.

The device identifier 45 will identify a node in this hierarchical tree and the complete set of attributes relevant to the device type may then be acquired by traversing the tree defined by the hierarchy structure from the node through successive layers to the root device type defined in the upper layer 61, acquiring further inherited attributes at each layer.

An advantage of organising each of the policy tables

40 in such a hierarchical structure is that, when it is required to expand the policy tables 40 to include data for an additional device type, this can be achieved by inserting a new node to the hierarchical tree structure and connecting the new node with a branch from a parent node in an adjacent upper layer. The parent node is selected to be the node from which the device type inherits the most complete set of attributes.

If for example a mobile telephone is already represented in the policy tables 40 and a new model of the mobile telephone becomes available, any change in the capabilities of the new model which require changes to policy can be represented by a minimal amount of data, corresponding to a new entry in each policy table containing values only in those fields for which attributes are not inherited from the existing model of mobile telephone represented by an existing node in the hierarchy.

Information for updating the policy tables 40 to include additional device types may be provided by the service centre 5 shown in Figure 1 and transmitted to the web server 1 for example over the internet 3. The device dependent information 25 including the policy tables 40 may then be updated by operation of data manager 190, a software module of the web server 1 represented

schematically in Figure 2. Generally this procedure may be undertaken during normal operation of the web server 1 and operation of the code generating engine 21. In response to the policy tables 40 being updated, the policy objects 41 automatically update themselves while managing their current use in the operating environment 46 of the code generating engine 21. This activity is regulated so as not to disrupt any web page code currently being generated by the code generating engine 21.

Figure 7 illustrates an example of software provided to the web server in a preferred embodiment. The software includes the set of policy tables 40 and a set of tags 70, including both tag definitions which provide the necessary information to enable authoring of content and code for implementing the tags i.e. the program code which runs when each tag is used in the operating environment of the code generating engine 21.

The code generating engine 21 and device identification engine 29 are among the items provided to the web server. Also provided is a reporting engine 71. The reporting engine is operated by the processor 20 to perform tasks such as measuring usage of the web site. This information may be used to demonstrate, for example, for the purpose of advertising, the extent to which the

site is used and also to enable personalised marketing to be targeted to users of the site. As an example, a database of users may be utilised to tailor a response to a request message from a particular user such that a response message includes an additional page containing a special offer on a product registered as being of interest to a particular user.

The software further includes an integration engine 72 for communicating with routine activities such as invoicing and purchase ordering that can arise in response to use of the website by users of the remote user devices 2.

The software further includes an operations manager 73 for managing the overall system to control operation, to allow errors to be detected and to allow resources to be controlled accordingly.

Also provided is a program 74 for the creation of the policy objects at startup as described above with reference to Figure 4.

The device identification engine 29 is also provided, as described above.

Also, a web application 75 processes the received URL request to identify the requested web page and select the appropriate document of content code 23 for import to the code generating engine 21.

Figure 8 illustrates schematically a typical hardware required for implementing the present invention in a small to moderate sized web server. A personal computer 80 having facilities for receiving programs such as those summarised in Figure 7 from a portable storage medium 81 is connected via a firewall 82 to a router 83. The router 83 is connected via a high bandwidth leased line 84 to an internet service provider 85 for connection with the internet 3.

Programs and data required for operating the web server 1 may be communicated from the service centre 5 in the form of the portable storage medium 81 or alternatively as signals 86 communicated via a network such as the internet 3.

As illustrated schematically in Figure 9, the above described web server and software configuration allows content code 23 to be stored separately from presentation information 90 so that, when a request message 91 is received by the system, the code generating engine 21 is able to generate dynamically a response message 92 containing newly generated web page code representing the combination of content code and presentation information. A particular advantage of this arrangement is that it is relatively easy to alter presentation information such as style, and themes including logos for example, simply

by updating the presentation information 90. Major re-engineering of the authored code is thereby avoided. Separate storage may involve location in separate files on the same hard disk or in physically separate memory devices.

To update the presentation information 90, it may simply be necessary to change data contained in the style policy table 32, the themes policy table 33 or the layouts policy table 35.

Figure 10 illustrates schematically a preferred embodiment in which the web server 1 is adapted to operate in a business environment for a medium to large scale business. The web server 1 comprises a production server 100 responsible for the production of web page information in response to request messages from remote user devices 2 and the outputting of response messages comprising the web page code. As such, the production server 100 performs the functions described above as being performed by the web server 1 of preceding figures. Here the web server 1 also comprises a development server 101 which is responsible for interaction between the web server and the service centre 5 and for periodically updating the production server 100 with new software.

Each of the production server 100 and development server 101 comprises a respective processor cluster with

substantially greater computing power than the personal computer 80 described above.

The development server 101 may be regarded as a series of processors performing development and staging processes enabling several levels of tests to be applied to new software before downloading the new software to the production server 100.

It is necessary for the software operated by the production server 100 to be updated periodically to take account of the evolution of new types of remote user device 2 and also to incorporate new content to be delivered as web pages.

As represented in Figure 10, the service centre 5 maintains a central reference database 102 from which data such as updated policy tables is periodically communicated to the development server 101. This communication of data occurs using the Internet 3, an XML syntax being used to structure data transmitted using HTTP protocol. As illustrated in Figure 10, the development server 101 periodically requests an update of the policy tables 40 by outputting a request message 103 and receives the requested data in a response message 104.

The development server 101 on receiving a response message 104 is used to conduct tests on the new data and

may be used to make local adjustments for example to add appropriate new layouts.

After testing and modification, new policy tables are communicated from the development server 101 to the production server 100 via appropriate firewalls.

When updating databases in the production and development servers it is not generally necessary to transmit the entire contents of the database, instead relying upon editing commands constituting the transmitted data in HTTP protocol accompanied by metadata in the XML syntax.

Also illustrated in Figure 10 is a probe site 105, the function of which will be described below, and which represents a server accessible via the Internet 3 and having a URL. The physical location of the probe site 105 is shown in the present example to be separate from the location of the web server 1.

Figure 11 illustrates schematically the manner in which the production server 100, probe site 105 and service centre 5 interact in response to an encounter for the first time with a remote user device 2 of a new type. The new type of device 2 may for example be the latest version of a mobile telephone using WAP and including features which are upgraded with respect to those of previous models by a given manufacturer, having for

example different display capabilities such as an increased size and the ability to display colour images.

The user of the new device 2 wishing to access the web site provided by the web server 1 actuates browser software within the device to generate a request message 110 which is output via the Internet 3 to the URL of the production server 100.

The production server 100 receives the request message 110 and inputs the message to the device identification engine 29 shown in Figure 2 in an attempt to extract a device type identifier from the ID string contained in the HTTP header of the request message. The device identification engine 29 however determines that the request message 110 contains an unknown identification stream. The production server 100 however needs a device type identifier to enable the appropriate policy tables to be accessed, the device ID 45 being an essential input to the code generating engine 21 as illustrated in Figure 4.

The production server 100 responds to this determination by the device identification engine 29 by generating a redirect message 111 which is transmitted to the new device 2 and causes the device to redirect the request message 110 to the URL of the probe site 105. In consequence, a new request message 112 is output by

the device 2 and transmitted to the probe site 105.

The probe site 105 analyses the identification string contained in the request message 112 and is able to extract some basic information as to the communication protocols and capabilities of the device 2. This limited information is sufficient to enable the probe site 105 to generate a probe 113 in the form of an agent which can be processed by the browser of the device 2 and constitutes software which can be run by the processor of the device.

The probe 113 is transmitted to the device 2 in a further message 114 and on receiving this message the browser of the device displays a standard response web page to the user while processing the probe in background.

The probe 113 causes the device 2 to generate a further request message 115 which is addressed to the probe site 105 and which contains more detailed information of the capabilities and protocols associated with the device 2, in accordance with the information requested by the probe.

The probe site 105 responds with a further redirect message 116 which is transmitted to the device 2 and instructs the device to redirect the request message 110 to the production server 100. The device 2 follows this

redirection and outputs a further request message 117.

At the same time as outputting the redirect message 116, the probe site 105 sends a notification message 118 to the service centre 5 to inform the service centre of the existence of the new device 2 and to pass on the information retrieved in message 115 by the use of the probe 113.

The service centre 5 stores the received information for later use and processes the information to generate a temporary update message 119 which is transmitted to the production server 100 and contains a device identifier to enable the code generating engine 21 to generate the web page code as requested by the user of the device 2. The web page code is then transmitted to the user in a final response message 120.

The temporary update message 119 generated by the service centre 5 provides the production server 100 with the nearest appropriate existing device type identifier. Consequently, the web page code generated may not be perfectly suited to the new device 2 but will generally be capable of being interpreted and displayed by the device, for example using default settings of the browser.

The service centre 5 is then required to update the central reference database 102 with appropriate

information including revised policy tables. This information may then be included in the next update communicated to the development server 101 for ultimately being used to update the database of the production server 100.

In this way, the above system is capable of continually reacting and adjusting to the evolution of new devices accessing web sites via the Internet, without necessarily being provided with advance notice by the manufacturers of the new devices or the direct provision by them of detailed technical data. Optionally, the service centre 5 may request such data by contacting the manufacturer, but this need not be obligatory to obtain satisfactory operation.

Figure 12 illustrates an embodiment in which the web server 1 is provided with a cache memory 120. A gateway server 121 acts as a gateway interfacing with the internet 3 to receive request messages from remote user devices 2 and to respond by sending the requested web page code 122 either by requesting the dynamic generation of the web page code from a dynamic word page server 123 or, if the requested web page code already exists as a static web page in the cache memory 120, retrieving the web page code from cache memory.

The dynamic web page server 123 functions to

dynamically generate the web page code in the manner described above using the device type identifier 45 extracted from the received URL request and the stored content code 23 corresponding to the URL.

Figure 13 illustrates schematically the steps carried out by the processor of the gateway server 121. At step 130, the gateway server receives the URL request and extracts the device type ID 45 and identifies the document of the content code 23 which contains the instructions necessary for generating the required web page code.

At step 131, the gateway server 121 queries whether the web page code already exists as a static web page in the cache memory 120 for this device type ID and URL and, if available, retrieves at step 132 the web page code.

If not available from cache memory 120, the gateway server 121 requests at step 133 from the dynamic web page server 123 the generation of the web page code and transmits the device type ID and URL details.

Accordingly, at step 134 the gateway server 121 receives the newly generated web page code.

At step 135, the gateway server 121 outputs the web page code to the remote user device 2 via the internet 3.

Figure 14 illustrates schematically the steps

performed under the control of the processor of the dynamic web page server. At step 140, the dynamic web page server receives the request for generation of the web page code and also receives the device type identifier information and information derived from the URL required to identify the document of content code 23. The required content code 23 is retrieved from memory and is processed in the run time environment 46 of the code generating engine 21 to produce the required web page code at step 142.

At step 143, the dynamic web page server 123 determines from the content code 23 whether the resulting web page code is flagged as being suitable for being stored in cache memory 120 and, if so, determines from the content code the period for which the cache memory version is to remain valid. This information at step 144 is used to create metadata which is added to the file containing the web page code and which at step 145 is output to the cache memory 120 to be stored for the validity period.

At step 146, the dynamic web page server 123 outputs the web page code to the gateway server 121 to be received at step 134 as described above.

When authoring the content code 23, the author therefore has the option of adding extra attributes to

canvas tags to define whether or not the resulting page can be cached and how long it can remain valid in cache memory. The cache memory 120 writes new cached pages into memory in the manner which overrides any stored pages for which the validity has expired.

In Figure 12, a set of pages of content code 23 is represented schematically together with respective cache memory control data 124 for each page, the control data being set by the author of the page to determine the above instructions for caching in memory 120.

The generation of code by the code generating engine 21 has been described above with reference for example to appendix 1 which shows instructions used for creating content code 23 for generating an example web page. Typically, as in this example, the execution of the content code 23 by the code generating engine 21 will require content to be retrieved from date storage. Data objects such as image files are therefore imported for inclusion in the web page code which is to be output by the code generating engine 21.

The author of the web page may author a component to refer by name to a data object to be imported. Different versions of the data object are stored in a data structure which is hierarchical and such that different hierarchical levels correspond to different

capabilities of remote user devices 2. The author of the content code 23 may define for each object the appropriate version to be utilised for each device type ID 45 by entering data in a in component policy table 37 as illustrated in Figure 3. The component policy table may then follow the hierarchy of Figure 6A to enable the data object version to be defined for each possible device type ID. This approach is referred to hereafter as defining data objects in terms of targeted content since it is the author of the code which targets the particular version of the data object to be used for each remote user device 2.

Alternatively, a number of versions of data object may be provided in a hierarchy with characteristics of each version of the data object being described in associated metadata for each version. Automatic selection of the appropriate version may then be made using a component logic software module 150 as illustrated schematically in Figure 15.

The component logic software module 150 selects the appropriate version on the basis of the data object metadata, layout considerations defined by the layout policy table 35 and information about the device provided by the device policy table 30. This approach to selection of the data object version will be referred

to below as untargeted content selection.

The code generating engine 21 may operate using either one of targeted content or untargeted content selection. Alternatively and preferably, the code generating engine 21 is capable of selectively making use of both targeted content and untargeted content selection, thereby providing the author with the freedom when authoring the content code 23 to either define data object versions or leave selection to be automatically processed using the component logic software module 150.

Figure 16 illustrates schematically the manner in which both targeted and untargeted content may be utilised. At step 160, content code 23 is input to the code generating engine 21 together with the device type ID 45 and at step 161 the processing of the code is started.

At each instance of a component of the content code 23 requiring a data object to be imported, the code generating engine 21 attempts at step 162 to import an author defined version of the data object. If it is determined at step 163 that such an author defined version exists and it is successfully imported, processing continues, determining at step 164 whether any further data objects remain to be imported.

If however at step 163 it is not possible to successfully import any author defined version, processing continues with step 165 which requires operation of the component logic software module to identify the appropriate object version. At step 165 the selected data object version is then imported and the code generating process continues. When it is determined at step 164 that no further data objects are to be imported, the remaining processing to generate the web page code is completed at step 167 and at step 168 the completed web page code is output.

Figure 17 illustrates schematically the manner in which data objects stored in the database 19 are stored in a hierarchical data structure to allow the component logic software module 150 to automatically select the appropriate data object version..

If for example the author designs a web page in which a video clip is to be displayed at a defined area of the screen of a user device in the form of a personal computer, the video clip is stored in the database 19 as a video data object 170 and an associated component name 171 is devised to enable the code generating engine 21 to reference the video data object 170 by name. The video data object 170 is characterised by metadata 172 stored in association with the data object and contains

data fields sufficient for the component logic software module 150 to make a determination as to whether a given user device 2 as defined by device type identifier 45 is capable of receiving and displaying the video clip from the data object.

The author may also store, as shown in Figure 17, different versions of the video data object suitable for use by different device types, each accompanied by respective metadata.

The author also stores in the database a photographic image data object 173 with associated metadata 174. Similarly, the author may store a set of related photographic image data objects 173 having attributes appropriate to different device types and accompanied by respective metadata. If the component logic software module 150 determines from the metadata 172 that none of the video data objects can be displayed by a user device for which the web page code is currently being prepared, the photographic image represents a fall back position enabling a static photographic image to be rendered in place of the video clip. The photographic image will therefore generally display related subject matter and may for example comprise a still taken from the video clip.

The author may also enter into the database a graphic image data object 175 with associated metadata 176. Similarly, the author may enter a set of related graphic image data objects with respective metadata, suited to different device types. The graphic image constitutes a further fall back position for use if an appropriate version of photographic image is not available.

The author may also enter additional fall back positions such as simple text objects 177 with associated metadata 178. The text data object 177 represents a final fall back for situations where the component logic software module 150 determines that even the graphic image 175 cannot be rendered by the user device 2.

The data objects 170, 173, 175 and 177 are stored in the database 19 in a hierarchical data structure as illustrated schematically in Figure 17 in which different levels of the hierarchy correspond to levels of user device capability. The text data object 177 is in this example a root level of the hierarchy representing the lowest level of capability of the user device 2 and therefore representing the ultimate fall back position.

If for example the user device 2 is identified by device type identifier 45 as being a personal computer, the component logic software module 150 will select data

object 170. If the user device 2 is identified as being a pocket PC without the ability to render video clips, but having the ability to render photographic images, the component logic software module 150 will select data object 173. If the user device 2 is identified as being a mobile telephone with WAP facilities, the module 150 will select the graphic image data object 175. If however the user device 2 is identified as being a conventional mobile telephone without WAP facilities then text data object 177 will be selected.

The hierarchical data structure applies to other types of data object to enable appropriate data objects to be selected, as for example in the case of the web page code defining a link of some sort. The link may comprise an HTML link, a WML link, an email link or simply a telephone number for autodialled connection in the case of the user device 2 being a mobile telephone.

A further example of data objects which can be accessed using the same component name is that of script components such as JavaScript and WML Script where the same content may be written in the appropriate script for use in different devices. The author may therefore write data objects in the variety of required script languages for storage in the hierarchical structure of Figure 17 and retrieval in response to component name. As in the

previous examples, selection may be made on the basis of the metadata accompanying each data object on the basis of the capabilities of the remote user device 2, as determined by using the device ID 45 with reference to the device policy table.

The use of the above hierarchical data structures provides an efficient method of asset management for the system where the assets comprise the collection of data objects which are capable of being delivered in web page code form by the system.

Figure 18 illustrates the manner in which data in the form of text may be stored in a hierarchical data structure to enable different versions of the text to be retrieved to suit the capabilities of the user device 2 and the preferences of the user. The full text in English of a document object is stored in box GB TEXT 1, suitable for example for display on the screen of a PC. A reduced version of the text which has been edited by the author is stored as data object GB TEXT 2, suitable for display on a palm held computer of reduced screen size. A smaller still version of edited text is stored in data object GB TEXT 3, suitable for display in a hand held device of limited memory and finally GB TEXT 4 contains a minimal text version for display in a mobile telephone.

Each of these text versions are in the English language. Corresponding text objects in the French language are contained in FR TEXT 1, FR TEXT 2, FR TEXT 3 and FR TEXT 4. Similarly, data objects containing text in the German language are contained in DE TEXT 1, DE TEXT 2, DE TEXT 3 and DE TEXT 4.

The data objects are addressed by component name 171 and a language identifier 180 which in this example indicates whether the English, French or German text version is required.

The language identifier is input to the code generating engine in addition to the device ID 45 and typically is extracted from the body of the request message generated by the browser of the remote user device 2 in response to input of a user preference by the user. Each of the text data objects such as GB TEXT 1 has associated metadata 181 indicating the relevant parameters of the object, including in particular the size of code representing the text. A decision may thereby be made on the basis of the size of the code indicated by the metadata 181 as to whether a given object is appropriate for retrieval for a given remote user terminal. The optimum data object may thereby be selected by traversing the hierarchical tree structure of Figure 18 until the appropriate data object is

located.

Figure 19 illustrates a further embodiment in which a pane dissector 190 is provided to modify the output produced by the code generating engine 21. Such an arrangement is advantageous in preventing the transmission of a response message containing an amount of code which is greater than the available memory capacity of the remote user device 2. For small memory devices such as mobile telephones, overloading the available capacity can cause the microprocessor of the device to crash or enter a lock situation. Although the author of the content to be provided in response to queries from such devices may take note of the limited memory capacity of the device, it is in practice difficult to predict with certainty the actual amount of code which will be output from the code generating engine 21 so that a solution is to additionally place a device such as the pane dissector 190 downstream of the code generating engine 21 to trap the outgoing code and measure the code amount.

If the code amount exceeds the indicated available data capacity of the user device 2, the dissector 190 is arranged to automatically divide or dissect the document into a number of fragments, referred to herein as shards.

As illustrated in Figure 19, the pane dissector 190

receives the device type identifier 45 from the device identification engine 29 and addresses the device policy table in order to determine from the device type identifier the available memory capacity of the user device 2.

If division of the pane is appropriate, a first one of the resulting shards is output by front end processor 190 and the remaining shards are stored in a buffer memory 191. The front end processor 190 may then respond to subsequent requests from the user device 2 by supplying in turn the remaining shards in respective response messages.

Figure 20 illustrates schematically the process of dissecting a pane, the term pane in this context being used to indicate a web page intended to be transmitted to a remote user device 2.

At step 200, the code representing the pane is generated by the code generating engine 21, based on the device type identifier 45 and the nature of the request message received via the internet 3.

At step 201, the code output from the code generating engine 21 is received and measured by the pane dissector 190 and at step 202 the pane dissector accesses the device policy table to look up the data capacity of the remote user device 2.

At step 203, the pane dissector 190 determines whether or not the amount of code is equal to or less than the data capacity of the remote user terminal 2 and, if so, at step 204 outputs the code without performing any dissecting step.

If however the amount of code is greater than the available data capacity, the dissector at step 205 dissects the code into a number of shards, each shard having an amount of code less than the available data capacity. At step 206 the shards of the divided pane are stored in buffer memory 191. At step 207 one of the shards is output to the server for inclusion in a response message to the remote user device 2.

Figure 21 illustrates a pane 210 from which regions A, B, C, D, E and F are dissected to form shards 211 to 216, each of which is separately stored in the buffer memory 191 and may be transmitted in separate response messages to the remote user device 2.

Figure 22 illustrates the additional use of a memory referred to below as a user session memory 220 for storing information defining a user session under circumstances where the messages between the server and user device 2 are fragmented to take account of the limited data capacity of the user device.

A typical example of when such an arrangement is

appropriate is when a form filling exercise is to be completed in a user session. For high capacity devices such as personal computers, it is commonplace for a form to be provided in a single screen to the user and for the form to contain multiple fields for completion by the user.

For example, in Figure 23 form 230 defines multiple data fields 231. For a device 2 of limited capacity, it may be necessary to fragment the form into a number of fragments 232 to 237 which contain respectively portions A, B, C, D, E and F of the form 230.

During a user session, the form 230 is dissected into the fragments 232 to 237 and the corresponding code for each fragment stored in buffer 191. Initially only fragment 232 is transmitted to the user device 2 and the user responds with a response message which is processed by the server 1. The response corresponding to data fields contained in region A of the form are stored in the user session memory 220. The next fragment 233 is transmitted to the user device and the corresponding response stored in the user session memory. This process is repeated until the final fragment 237 is transmitted. The response to this fragment F of the form includes actuation of a submit button 238. If actuation of the submit button 238 is contained in the final response

message, the information required in the user session memory 220 is complete and the complete set of data from the form filling exercise is passed to a data processing application 239.

The manner in which the form is divided into regions A, B, C, D, E and F is defined by the author as part of the layout policy.

The code transmitted to the user device 2 typically also includes script defining instructions for operating the browser of the remote user device 2 to perform validation and verification of data entered by the user during form filling. The script is generated by the code generating engine 21 based on the tags written by the author who includes a definition of validation rules to be applied to the data entered by the user during form filling.

If for example a data field is indicated as being a numerical field, numerical value limits may be applied for validation purposes.

The manner in which the script code is generated by the code generating engine 21 depends upon the version of script language appropriate to the remote user device 2, as indicated by the protocol policy.

In a further example, a user device 2 is of a type

with limiting processing power such as a mobile telephone in which validation rules are applied by software existing in the processor or SIM card of the device. In this instance, the code generating engine 21 is not required to generate script for transmission to the device to perform the validation steps since it is sufficient to provide validation parameters in a format dictated by the existing software within the user device.

When responding to a remote user device 2 which uses a protocol supporting style sheets, the author may specify that a style sheet is to be generated by the code generating engine 21. The style sheet may be shared across multiple pages. For those devices where the available protocol does not support style sheets, additional code is generated by the code generating engine 21 to create a visual effect in the resulting displayed pages which emulates the style sheet definition defined by the author.

For example, HTML 4 browsers can apply style sheets to received HTML whereas HTML 3 browsers cannot. It is therefore necessary for HTML 3 browsers to receive additional HTML code adding font, colour and other attributes in order to achieve the same effect as provided by the style sheet.

The author is required to define the style sheet information and the code generating engine 21 will, if necessary, automatically generate the additional code required if style sheets are not supported.

Some user devices do not provide visual display, as for example those devices designed for use by visually impaired users. Such devices are typically served by a service provider which hosts a voice browser on a computer of the service provider, thereby converting HTTP received code into communications signals delivering voice messages. Such a service provider could therefore act as an intermediary between the server 1 and the remote user device 2 of the above embodiments. The output web page code could then advantageously be output by the code generating engine in voiceXML.

Figure 24 illustrates one of the functions of the graphical user interface 4 utilised when storing files created during web page authoring. A common situation arises where the same background image and text are to be incorporated into web pages downloaded to a number of different device types requiring different formats. A data converter 241 is provided in accordance with a further embodiment of the present invention to assist the author in the task of genrating appropriate files. As shown in Figure 24, an authoring suite 240 is provided

with a data converter 241 for converting a data file output by the authoring suite into a set of files which are stored by action of a file manager 242 in the data structure.

Figure 25 illustrates the steps in the method of operating the elements of Figure 24. At step 250, the data converter 241 receives from the authoring suite 240 a file containing data which is to be stored as a data object. As an example, this data file comprises an image in GIF format.

At step 251, the data converter 241 determines the type of data (i.e. whether image, text or other data type) and refers to the device policy table 30 to determine a list of different data formats which may be required during operation of the code generating engine 21, according to the value of the device type identifier 45.

At step 253, the data converter 241 generates for each of these formats in the list a converted file having the appropriate format and associated metadata. At step 243, the file manager 242 stores the files in the data structure.

This arrangement enables the author to enter text and graphic parameters only once, the data converter and file manager then undertaking the task of generating sets

of files in different formats for storage and subsequent use as data objects.

The above described embodiments have referred to implementation using JAVA, by way of example. Alternatives to JAVA may be used in implementing the present invention and the references to JAVA, JAVA beans and JAVA virtual machine are not to be read as limiting the scope of the invention.

The present invention can be implemented by a computer program operated on a computer in the context of a web server. An aspect of the present invention thus provides a storage medium storing processor implementable instructions for controlling a processor to carry out the method as hereinbefore described.

Further the computer program can be obtained in electronic form for example by downloading the code over a network such as the internet. In accordance with another aspect of the present invention there is provided an electrical signal carrying processor implementable instructions for controlling a processor to carry out the method as hereinbefore described.

The present invention has applications to networks other than the internet including private networks and further public networks.

The policy tables, custom tags and the code

generating engine may be supplied as software products either separately or in combination as a software kit and constitute further aspect of the present invention, whether embodied in storage medium or electrical signal form.

The device names referred to in Figure 6 include in layers 63, 64 and 65 the names of device types which are hereby acknowledged as being Trade Marks.

Appendix I - Canvas Authored for the Volantis Product

```
<%@ include file="Volantis.jsp" %>

<vt:canvas layoutName="eportal" themeName = "theme1>

<vt:anchor href="TestPortal.jsp" pane="logo1" image="stars"/>

<vt:logo pane="logo2" src="volantis" alt="volantis"/>

<vt:welcome pane="welcome"/>

<vt:h2 pane="shop">Shop for Cool Stuff at <a href="shopcart.jsp">Shop
Volantis</a></vt:h2>

<vt:Headline pane="headlines" show="2" />

<vt:Content pane="cpynews"/>

</vt:canvas>
```

Appendix 2 - Resulting Page Sent to a Personal Computer Using HTML

```
<html>
 <head>
 <link REL=STYLESHEET HREF="css/JSP-Styles.css"
 TYPE="text/css">
 <script language="JavaScript">
 <!--
 //--></script>
 </head>
 <body>
 <table border=0 cellpadding=0 columns=2><tr>
 <td align=center valign=top>
 <table border=0 cellpadding=0 columns=1>
 <tr><td align=center valign=top>
 <table border=0 cellpadding=0 columns=1>
 <tr><td align=center valign=top>
 <table border=0 cellpadding=0 columns=2><tr>
 <td align=center valign=top>
 <table border=0 cellpadding=0><tr>
 <td>
 <a href="TestPortal.jsp" >
 <img src="images/stars0.jpg"/>
 </a>
 </td></tr></table>
 </td>
 <td align=center valign=top>
 <table border=0 cellpadding=0><tr>
 <td>
 <img src="images/volantis0.jpg" alt="volantis">
 </td></tr></table>
 </td>
 </tr></table>
 </td></tr>
 <tr><td align=center valign=top>
 <table border=0 cellpadding=0><tr>
 <td>
 <hr><b>Welcome to Volantis, Rhys
 Lewis</b>
 . You can set up your news preferences
```

```
<a href=e_setupnewsdevice.adp>here</a>
<br><hr>
</td></tr></table>
</td></tr>
<tr><td align=center valign=top>
<table border=0 cellpadding=0><tr>
<td>
<h2>Shop for Cool Stuff at <a href="shopcart.jsp">Shop Volantis</a></h2>
</td></tr></table>
</td></tr>
</table
</td></tr>
<tr><td align=center valign=top>
<table border=0 cellpadding=0 columns=2><tr>
<td align=center valign=top>
<table border=0 cellpadding=0><tr>
<td>
<table align=right border=0>
  <tr><td><table  borderwidth=1  bordercolor=cyan><tr><th  class=headline
align=left>XML and metadata news</th></tr>
<tr><td><a
href=clickthru.jsp?cat=278&url=http://c.moreover.com/click/here.pl?x8102028>Thin
kersGroup.com Completes Beta Test of Web-To-Wireless Application</a></td></tr>
<tr><td class=sub>07 Jul 2000 08:05AM</td></tr>
<tr><td><a
 href=clickthru.jsp?cat=278&url=http://c.moreover.com/click/here.pl?x8099873>XSL
gives your XML some style</a></td></tr>
<tr><td class=sub>07 Jul 2000 06:59AM</td></tr>
</table></td></tr>
    <tr><td><table  borderwidth=1  bordercolor=cyan><tr><th  class=headline
align=left>Wireless sector news</th></tr>
<tr><td><a
 href=clickthru.jsp?cat=277&url=http://c.moreover.com/click/here.pl?x8102312>Nokia,
C&W team up to offer mobile internet services</a></td></tr>
<tr><td class=sub>07 Jul 2000 08:16AM</td></tr>
<tr><td><a
 href=clickthru.jsp?cat=277&url=http://c.moreover.com/click/here.pl?x8102269>NASDAQ
Outlook; Qualcomm To Finish Physics Lesson Soon, Then Could Double</a></td></tr>
<tr><td class=sub>07 Jul 2000 08:15AM</td></tr>
</table></td></tr>
```

<tr><td><table borderwidth=1 bordercolor=cyan><tr><th class=headline align=left>Top technology stories</th></tr>

<tr><td><a

href=clickthru.jsp?cat=273&url=http://c.moreover.com/click/here.pl?x8102034>PRIMUS Telecommunications and Inktomi Forge Strategic Alliance To Build Global Infrastructure for Streaming Media</td></tr>

<tr><td class=sub>07 Jul 2000 08:05AM</td></tr>

<tr><td><a

href=clickthru.jsp?cat=273&url=http://c.moreover.com/click/here.pl?x8098274>Jesse Berst's Anchordesk: The Long Wait for Web Phones</td></tr>

<tr><td class=sub>07 Jul 2000 06:19AM</td></tr>

</table

</td></tr></table>

</td>

<td align=center valign=top>

<table border=0 cellpadding=0><tr>

<td>

<table width=100% align=top>

<tr><td colspan=2><h2>Informix Upgrades Java Database</h2></td></tr>

<tr><td width=30% valign=top align=left><img src=database.gif border=0></td>

<td width=70%> Informix (NASDAQ: IFMX) says it has upgraded its Cloudscape Java-based database to support Windows CE and Pocket PC. In addition to adding platforms, Cloudscape 3.5 has added greater security.

Cloudscape 3.5 consists of the Cloudscape database management system, Cloudsync for data and application synchronization and Cloudconnector, a server framework for Internet connections to the Cloudscape DBMS.

The company is demonstrating the update at the JavaOne conference this week in San Francisco. It will be commercially available in July 2000. Server pricing starts at $1,999. </td></tr>

<tr><td colspan=2><h2>AT&T Wireless, Nortel to Trial GPRS</h2></td></tr>

<tr><td width=30% valign=top align=left><img src=wireless.gif border=0></td>

<td width=70%> AT&T Wireless Services (NYSE: AWE) and Nortel Networks (NYSE/TSE: NT) say they will start testing broader-band General Packet Radio Service (GPRS) in the U.S. this summer and are working toward the eventual delivery of faster third-generation (3G) service.

The GPRS tests will begin in several unnamed major U.S. cities this summer, the companies said in an announcement. The companies are working together to eventually introduce full 3G TDMA- EDGE technology.

"These trials using GPRS core network solutions from Nortel Networks are a significant step in the development of TDMA-EDGE and underscore the rapid progress

```
industry is making", said Roderick Nelson, chief technology officer, AT&T Wireless
Services. The eventual goal is to offer broadband wireless coverage to the
company's customers anywhere in the world, he added.</p>
    <p>The companies didn't provide specifics about when the GPRS service will be
commercially available.</p>
    </td></tr>
    </table>
    </td></tr></table>
    </td>
    </tr></table>
    </td></tr>
    <tr><td align=center valign=top>
    <table border=0 cellpadding=0><tr>
    <td>
    </td></tr></table>
    </td></tr>
    </table
    </td>
    <td align=Left valign=Center>
    <table border=0 cellpadding=0><tr>
    <td>
    </td></tr></table>
    </td>
    </tr></table>
    </body></html>
```

Appendix 3 - Resulting Page Sent to an Internet-Enabled Mobile Phone

```
<?xml version="1.0"?><!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml><card>
<table align="center" columns="2"><tr>
<td>
<p>
<a href="TestPortal.jsp">
<img src="images/stars10.bmp"/>
</a>
</p>
</td>
<td>
<b>volantis</b>
</td>
</tr></table>
<p><em><a href="list1tmp1475618599.wml">Company News</a></em></p>
<p><em><a href="list1tmp1484083075.wml">Headline News</a></em></p>
<p><a href="LoginForm.jsp">Please log in</a></p>
</card></wml>
```

Appendix 4 - The Deck Generated Automatically to Hold the Company News

```
<?xml version="1.0"?><!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN""httpd://www.wapforum.org/DTD/wml_1.1.xml">
<wml><card>
<p><em><a href="list2tmp302605028.wml">Informix Upgrades Java Database</a></em></p>
<p><em><a href="list2tmp299142288.wml">AT&amp;amp;T Wireless, Nortel to Trial GPRS</a></em></p>
</card></wml>
```

Appendix 5 - The Deck Generated Automatically to Hold the Story About AT&T

<?xml version="1.0"?><!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN""httpd://www.wapforum.org/DTD/wml_1.1.xml">

<wml><card>

AT&T Wireless, Nortel to Trial GPRS AT&T Wireless Services (NYSE: AWE) and Nortel Networks (NYSE/TSE: NT) say they will start testing broader-band General Packet Radio Service (GPRS) in the U.S. this summer and are working toward the eventual delivery of faster third-generation (3G) service.

The GPRS tests will begin in several unnamed major U.S. cities this summer, the companies said in an announcement. The companies are working together to eventually introduce full 3G TDMA- EDGE technology.

"These trials using GPRS core network solutions from Nortel Networks are a significant step in the development of TDMA-EDGE and underscore the rapid progress industry is making", said Roderick Nelson, chief technology officer, AT&T Wireless Services. The eventual goal is to offer broadband wireless coverage to the company's customers anywhere in the world, he added.

The companies didn't provide specifics about when the GPRS service will be commercially available.

</card></wml>

CLAIMS

1. A method of operating a system for processing data for storage in readiness for being output to one of a number of possible user devices having different technical capabilities, the method comprising:

receiving a data element;

generating a set of data objects from the data element such that each data object comprises a respective version of data contained in the data element, and storing the set of data objects in a hierarchical data structure in which different levels correspond to respective technical capabilities of user devices.

2. A method as claimed in claim 1 further comprising:

determining the type of data contained in the data output;

referring to a device policy table using the type of data and a device type identifier representative of the technical capabilities of the user device to determine a list of different data formats;

wherein the generating step comprises generating for each of these formats in the list a converted file

having the appropriate format; and

wherein the storing step comprises storing the converted file in the data structure.

3. A method as claimed in claim 2 wherein the storing step further comprises storing associated metadata with each converted file, wherein the metadata contains data fields sufficient to allow a determination as to whether a given user device as defined by device type identifier is capable of receiving and display data from the data object.

4. A method as claimed in claim 3 wherein, for a data element comprising image data, the set of data objects generated comprises video data, still image data and text.

5. A method as claimed in any of claims 1 to 3 wherein, for a data element comprising a link, the set of data objects comprises an HTML link, a WML link, an email link and a telephone number.

6. A method as claimed in any of claims 1 to 3 wherein, for a data element comprising text, the set

of data objects comprise a data object including the complete text, and successive data objects comprising respective edited versions of the text.

7. A method as claimed in claim 6 wherein the generating step further comprises generating multiple versions of each data object in different languages.

8. A system for processing data for storage in readiness for being output to one of a number of possible user devices having different technical capabilities, the system comprising:

receiving means for receiving a data element;

a data converter for generating a set of data objects from the data element such that each data object comprises a respective version of data contained in the data element; and

a file manager for storing the set of data objects in a hierarchical data structure in which different levels correspond to respective technical capabilities of user devices.

9. A data structure in which data objects are accessible in a hierarchy in which different

hierarchical levels contain different versions of a data component corresponding to respective levels of technical capabilities of user devices to which the data objects are to be provided.

10. A computer program comprising processor implementable instructions for controlling a processor for carrying out the method of any one of claims 1 to 7.

11. A storage medium storing processor implementable instructions for controlling a processor to carry out the method of any one of claims 1 to 7.