(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 September 2006 (21.09.2006)

PCT

(10) International Publication Number
WO 2006/097148 A1

(51) International Patent Classification:
G06F 17/30 (2006.01) *G06F 9/445* (2006.01)

(21) International Application Number:
PCT/EP2005/051151

(22) International Filing Date: 14 March 2005 (14.03.2005)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **SENDO INTERNATIONAL LIMITED** [CN/CN]; 1601-3 KIN-WICK CENTER, 32 HOLLYWOOD ROAD CENTRAL, HONG KONG, 068645 (KY).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **CLARK, William Allan** [GB/GB]; Suite 1, 341 Lea Bridge Road, Leyton Greater London E10 7LA (GB).

(74) Agent: **WRAY, Antony John**; GROVE HOUSE, LUTYENS CLOSE CHINEHAM COURT, BASINGSTOKE Hampshire RG24 8AG (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

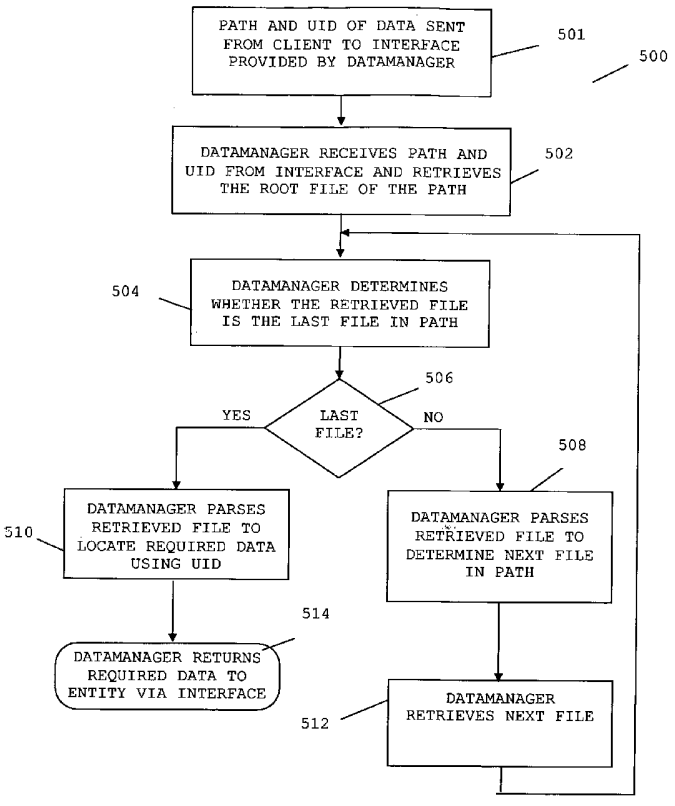
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

[Continued on next page]

(54) **Title:** ELECTRONIC DEVICE AND METHOD FOR RETRIEVING DATA ELEMENTS THEREIN



(57) **Abstract:** A DataManager function (406) is implemented on a resource-constrained electronic device, such as a wireless communication unit, that receives, say from a client resource (400), a location identifier such as a file-PATH and/or unique identifier of a data element embedded in a textual data file (306) by means of a markup language. The DataManager (406) then parses a root data file, locating the data element and returns the data element to the client resource (400). In this manner, an improved mechanism for accessing data is provided, which offers increased reliability and security. It is particularly advantageous where a number of resources of the resource-constrained electronic device are configurable and are required to be flexible and updateable during the lifetime of the wireless communication unit.

WO 2006/097148 A1

Published:
— *with international search report*
— *with amended claims and statement*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

ELECTRONIC DEVICE AND METHOD FOR RETRIEVING DATA ELEMENTS THEREIN

Field of the Invention

This invention relates to a resource-constrained electronic device and a method for retrieving data elements therein. The invention is applicable to, but not limited to, mechanisms for retrieving data elements in a wireless data and/or communication unit, such as a mobile phone.

Background of the Invention

In current and future wireless communication technology ease of configurability and re-configurability of mobile phone handsets is becoming an increasing and desirable requirement. This flexibility is required in order to facilitate the building of products that are customised to the requirements of Network Operators, and also to facilitate user personalisation. To offer such flexibility, many aspects of a handset are therefore required, by design, to be as configurable as possible.

Mobile phone handsets now incorporate their own operating systems (OS), which, for reasons of software re-use, reliability and customer familiarity, need to be utilised over several generations of handsets. Notably, OSs are generally developed to provide as little configurability as possible, in order to ensure consistent performance across multiple hardware platforms. A hardware platform can be considered as being constructed from typically many separate and distinct components, between which

there may be little consistency or commonality of design in the context of the OS functionality.

In the field of data management, it is known that persistence mechanisms are often required to 'persist' or store information, files, resources, etc., even when the device is switched off, and thereafter allow the persisted information to be easily retrieved. In relation to configuration, these persistence mechanisms would allow, for example, a user to store parameters, resources, etc. used when re-configuring the device.

However, presently these mechanisms have limited use, and when the configuration data is stored in binary format, it is very difficult for the data to be changed. For example, if data relating to a feature or function of a mobile phone is stored in binary format, the data needs to be defined at an early stage in the manufacturing of the phone, and would rarely be re-configurable (e.g. by a Network Operator or end user) at a later stage.

In the context of mobile phones, an important area of configuration is the configuring of images, text, etc. as required by a Network Operator. A mobile phone is known to be resource constrained in the area of battery power, memory (often random access memory (RAM)), processing power, etc.

In effect, a means of organising information, data, resources, etc. is required that can be used by the mobile phone handset in such a way as to allow the data to be used/updated/changed in order to easily configure the handset. A mechanism that would theoretically allow

such organisation of data exists in the form of Mark-up Languages.

A Mark-up Language is a set of labels that are embedded within text to distinguish individual elements or groups of elements for display or identification purposes. The labels are typically known as "tags". Mark-up Languages identify elements that occur within a continuous stream of text, rather than elements that occur in a more structured manner, such as data in a database.

One Mark-up Language that exists, which would be well suited for such wireless, resource-constrained communication units, is XML (E**X**tensible **M**ark-up **L**anguage). The World Wide Web Consortium (W3C) has developed XML. XML is an open standard for describing data. It is traditionally used for defining data elements on a web page and business-to-business documents. XML uses a tag structure similar to hypertext mark-up language (HTML); however, whereas HTML defines how elements are displayed, XML defines what those elements contain. XML allows tags to be defined by the developer of the page. Thus, virtually any data items can be identified, allowing, for example, web pages to function like database records. By providing a common method for identifying data, XML supports business-to-business transactions. In other words, XML is a Mark-up Language that turns text streams into the equivalent of database records.

One problem with using XML is that known methods of handling the data are inefficient in terms of memory and

processing cycles. An example of an XML file is illustrated in FIG. 1.

As can be seen from FIG. 1, data is organised in a single XML file or page, using "tags" such as: 'GROUP ELEMENT', 'INDIVIDUAL ELEMENT', 'DATA1' and 'DATA2'. In order to access the data, the entire page is parsed, and the data stored in memory (e.g. RAM). The problem with this traditional method is that, if the file is large, it can take a significant amount of time to parse the entire file, in order to obtain all the data. More significantly, if only a small part of the data is required, this approach is extremely inefficient in terms of the amount of time required to locate the required data.

Furthermore, if only a small amount of the data is actually required, and since all of the data contained in the XML file is stored in memory, it is also an inefficient use of both memory and processing cycles in accessing the memory. There is generally no limit to the size of an XML file, nor the amount of data that can be contained within it. Consequently, prior to parsing the XML file, the amount of memory required to store the data is unknown and can be considerable.

For devices such as mobile phone handsets, the amount of memory available is limited, and users expect operations to take place instantly. Thus, such time delays in accessing data, and the potentially large amount of memory required to store such data, is unacceptable. Consequently, this has meant that the use of XML files is generally restricted to circumstances where the XML files

are very small in size. Thus, the use of XML files on such devices for providing such features and configurability etc. has been somewhat impractical.

To help improve the structuring and organisation of data within XML files, an inclusion mechanism has been developed by W3C, to facilitate modularity and add to the XML language. This inclusion mechanism is called XInclude. XInclude provides a generic mechanism for merging XML documents. In this way, a large XML document can be divided into a plurality of smaller documents. The smaller XML files containing the data can then be referenced from a "parent" XML file.

Furthermore, XInclude processing is recursive. That is, an included document can itself include another document. Thus, the XML documents can be organised into a tree structure, starting with a root XML file, from which all other XML files can be navigated to, either directly or through other XML files.

To help in the navigation through the XML files to required data, a set of syntax rules for defining parts of an XML document has also been developed by the W3C, called 'XPath'. XPath uses path expressions, which are used to identify nodes in an XML document. These path expressions have a similar appearance to the expressions one would see when working with a computer file system.

XML documents can be represented as a 'tree' view of nodes, having a similar appearance to the tree view of folders one would see when working on a computer. XPath uses a pattern expression to identify nodes in a logical

XML document. An XPath pattern is a slash-separated list of child element names that describe a path through the XML document. The pattern 'selects' elements that match the path.

Thus, XInclude provides a method of producing a single logical XML document from a plurality of individual actual XML files, whilst XPath provides a means for locating elements and/or attributes within the logical XML document.

The specifications for XML, XInclude and XPath are located on the W3C website, at http://www.w3.org, and are known in the industry. Consequently they will not be further described herein.

The main advantage of using the Xinclude and XPath mechanisms is that data is no longer stored in a single and potentially unwieldy XML file, but rather a plurality of small, location-independent XML files whose organisation is tailored to the particular user of that file. This has made the use of XML popular in the personal and network computer domain.

However, traditional methods of parsing XML files still result in the entire XML tree structure (i.e. all files) to be parsed, and all data to be stored in memory. Therefore, although using XInclude in the aforementioned manner makes the organisation of the data slightly more manageable, it does not improve the efficiency in terms of memory usage and speed (processing cycles) in accessing the data. Thus, the use of XML in a resource-constrained environment such as a mobile phone has so far

been limited to, for example, the following basic modules of the phone:

(i) Basic input-output (BIO) message format;

(ii) Context sensitive help;

(iii) Wireless Access Protocol (WAP) Binary XML (WBXML), which is a variation on the XML standard involving the use of dictionary look-up of tokenised elements; and

(iv) XHTML (EXtensible HTML) in a web browser, namely the combining of HTML 4.0 and XML 1.0 into a single format for the Web. XHTML enables HTML to be eXtended with proprietary tags. XHTML is also coded more rigorously than HTML and must conform to the rules of structure more than HTML.

A need therefore exists for a resource-constrained device and method of retrieving one or more data elements in the resource-constrained device, such as a mobile phone, and a method for configuring large amounts of data in the device, wherein the above-mentioned disadvantages of current data management methods in resource-constrained devices may be alleviated.

Statement of Invention

In accordance with a first aspect of the present invention, there is provided a method for retrieving one or more data elements associated with hardware and/or a software resource of a resource-constrained electronic device such as a wireless communication or data unit, as claimed in Claim 1.

In accordance with a second aspect of the present invention, there is provided a data parser, as claimed in Claim 13.

In accordance with a third aspect of the present invention, there is provided a resource-constrained electronic device as claimed in Claim 14.

In accordance with a fourth aspect of the present invention, there is provided a resource-constrained electronic device, as claimed in Claim 15.

In accordance with a fifth aspect of the present invention, there is provided a storage medium, as claimed in Claim 27.

Further features of the present invention are as defined in the appended Claims.

In summary, a resource-constrained device such as a wireless communication unit is based around a hardware platform comprising a number of software configurable components or resources, whereby a textual configuration file is stored on the wireless communication unit. The textual configuration file is accessed by a DataManager function of the wireless communication unit at the request of a resource or component of the wireless communication unit. The DataManager locates a configuration data element and returns it to the resource, thus allowing the resource to access configuration data.

In this manner, all suitably enabled components or resources of the wireless communication unit can be configured or re-configured as and when required.

In the context of a mobile phone, the provision of a simple configuration method allows greater flexibility in the design phase of the phone as well as in the customisation of the phone, say, by the phone user.

The expression 'resource', in the context of the present invention, encompasses or extends to any functional unit of the wireless communication unit such as the display, signal processing unit, sound generation system, wireless interface, keypad, etc. or any part thereof which is or may be configurable.

Brief Description of the Drawings

FIG. 1 illustrates a very, simple XML document.

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

FIG. 2 illustrates a block diagram of a wireless communication unit adapted in accordance with the preferred embodiment of the present invention;

FIG. 3 illustrates a hierarchical XML document in accordance with the preferred embodiment of the present invention;

FIG. 4 illustrates a schematic of the data-manager/client interface in accordance with the preferred embodiment of the present invention; and

FIG. 5 illustrates a flow chart of a method for retrieving a data element in accordance with the preferred embodiment of the present invention.

Description of Preferred Embodiments

The preferred embodiment of the present invention will be described in terms of a mobile telephone. However, it will be appreciated that the invention may be embodied in any other type of memory and/or processor constrained device that incorporates embedded intelligence in the form of, say a micro-processor or digital signal processor (DSP), etc. Such embedded intelligence may be customised or (re-) configured and may incorporate individual devices or features that may themselves be parameterised or configured, such as, for example a portable computer or a personal digital assistant (PDA), etc. It is also envisaged that the present invention is not limited to wireless communication units, as fixed communication units such as business/home telephone devices that connect to the public services telephone network (PSTN) also often utilise embedded microprocessor intelligence and include configurable resources, which could benefit from the inventive concepts described herein.

Referring next to FIG. 2, there is shown a block diagram of part of a wireless communication unit 200, adapted to support the inventive concepts of the preferred

embodiments of the present invention. The communication unit 200, in the context of the preferred embodiment of the invention is a mobile phone with an antenna 202. As such, the communication unit 200 contains a variety of radio frequency devices or circuits 206, operably coupled to the antenna 202 that will not be described further here. The communication unit 200 further comprises a signal processing function 208. An output from the signal processing function 208 is provided to a suitable user interface (UI) 210 comprising, for example, a display, keypad, loudspeaker and/or microphone.

The signal processor is coupled to a memory device 216 that stores operating regimes, such as decoding/encoding functions and the like and may be realised in a variety of technologies such as random access memory (RAM) (volatile), (non-volatile) read only memory (ROM), Flash memory or any combination of these or other memory elements. A timer 218 is typically coupled to the signal processor 208 to control the timing of operations within the communication unit 200.

In accordance with the preferred embodiment of the present invention, the signal processing function 208 is modified to incorporate, or be operably coupled to, a DataManager function 222. The DataManager function 222 is capable of communicating with device memory 216 and other client resources of the phone, such as the display or display-driver software. The DataManager function 222 is also able to receive requests for data from these client resources. The DataManager function 222 is also designed to locate the required data and return it to the

client application or resource as described in greater detail with respect to FIG. 3, FIG. 4 and FIG. 5.

Referring now to FIG. 3, a schematic of a file hierarchy with a Root file 300 and a number of sub-files 306 is illustrated. Each of the sub-files 306 is XML conformant. The preferred method organises files into a file structure conforming to the Xinclude standard. Furthermore, the method preferably generates a PATH 302 through the hierarchy conforming to the XPath standard. However, when the inventive concepts are applied to other mark-up languages, it is envisaged that other means of locating a data element could be employed, which may or may not be similar to the XInclude and XPath approaches. The advantage of using the XInclude and XPath standards is that data is no longer stored in a single and potentially unwieldy XML file, but rather a plurality of small, more organised XML files, which can be more easily handled, i.e. read from/written to memory and parsed.

A number of files 306 (identified as Operator.XML) are shown containing a data element that may be required by a resource of the communication unit, the element being uniquely identifiable by its PATH 302 and its Unique Identifier (UID).

In accordance with the preferred embodiment of the present invention, a DataManager (such as DataManager function 222 of FIG. 2) exports an interface to one or more client entities, where a client entity is an application or device or component of the communication unit. The interface preferably takes two parameters from the client entity:

(i) PATH - A tag path that identifies an XML document: the path is intended to be used as a limited implementation of the XPath standard. XPath provides the means to locate elements within the logical XML document, which is created from the individual actual XML documents using XInclude; and

(ii) UID - A Unique Identifier of a tag that is the sole means of addressing a required XML tag. XML tags that do not have a UID attribute are not addressable.

The PATH parameter is preferably not a mandatory parameter. The UID is preferably sufficient to identify, without ambiguity, any addressable data tag. It is envisaged that the DataManager may use any such location identifier to identify a particular location of a data element or data file within a single hierarchical textual file. However, in order to discover the data tag that has as its UID attribute the supplied UID, the complete logical XML document would have to be parsed in order to locate the UID. The provision of a PATH enables the identification of the actual XML document that contains the UID attribute, by only passing those documents required to follow the path, thereby reducing by orders of magnitude the amount of XML parsing required.

Thus, in accordance with the preferred embodiment of the present invention, a client resource of the communication unit requests a data element by sending a PATH 302 and UID of the required data element to the DataManager. The DataManager retrieves the relevant XML files from memory, parsing the files, locating the data 304 and returning the data to the client resource.

Advantageously, the DataManager retrieves and parses only the specific files 306 that lie on the file PATH 302 provided by the client resource. Thus, in this manner, the volume of data to be parsed, particularly the amount of physical volatile memory (RAM) required for the operation, is significantly reduced. Furthermore, the time required to fulfil the request is minimised. This, in turn, reduces the hardware and processing requirements on the communication unit.

Thus, a communication unit incorporating a DataManager function is provided, whereby the DataManager function, upon receiving an instruction to supply a data element from a client resource residing on the communication unit 200, supplies the File-PATH and UID 412 of the data element 304 with respect to a root data-file 300. The DataManager function is capable of parsing the Root Data file 300, locating the data element 304 and returning the data element 304 to the requesting client.

Referring now to FIG. 4, a DataManager 406 is illustrated in schematic form, in accordance with a further advantageous feature of the present invention.

The DataManager function 406 is, in this example, implemented on, say, the signal processor 208 of the wireless communication unit 200 of FIG. 2. Further, the DataManager function 406 has an interface 404 over which it exchanges information 402, 412 with a client resource 400. The interface 404 is preferably a logical interface and not a physical interface, and can be thought of as a

type of `socket' that the DataManager 406 creates for the client resource 400.

The DataManager interface 404 will return an instantiated object, effectively a copy of the retrieved data. The returned object is preferably owned by the client resource/application, so that the DataManager is not responsible for persisting (i.e. storing) the data. If the DataManager is able to locate the data addressed by the PATH and UID parameters, the returned object preferably contains a Document Object Model (DOM) representation of the addressed data. Such a DOM preferably contains, for example, a name, a list of attributes, the data and an indication of the 'children' (sub-documents) relating to the data. If the DataManager is not able to locate the addressed data an indication that the required data is not available is preferably provided.

Thus, in summary, the DataManager function 406 is able to open interfaces simultaneously with a number of client resources 400, thus optimising its own operation. Preferably, the interfaces 404 are destroyed once the transaction is complete. The DataManager function 406 is preferably implemented as a system level library, such as a dynamically linked library (dll). Multiple client resources 400 can then use this library simultaneously. As noted earlier, addressed tags will be returned to the client application by the DataManager function in the form of a DOM representation.

The DataManager function 406 also communicates with a memory element/storage device 410 (say memory element 216

of FIG.2) in which the hierarchical textual data file (such as textual file 300 of FIG. 3) is stored. The DataManager 406 is able to read from and write to this memory element/storage device 410. It is envisaged that the memory element/storage device 410 may comprise volatile and/or non-volatile memory as required. The DataManager function 406 is advantageously capable of creating a data cache, which at a minimum includes the location of actual XML documents resolved by the `PATH' parameter. This optimised caching of data improves the communication unit's data retrieval and data management performance.

It is noteworthy that, for communication units such as mobile phones, the amount of available RAM (volatile memory) is limited. Hence, ideally the DataManager function 406 should use as little memory as possible, whilst at the same time retrieving data as quickly as possible. Therefore, it is important that an appropriate balance is achieved between caching enough information in order to be able to retrieve data quickly, whilst keeping the amount of RAM used to a minimum. The use of XInclude allows the data to be partitioned in a manner that enables a minimum of (RAM and processor cycle) resources to be used. It is also envisaged that the DataManager function 406 may access a root file via a wireless communication link of the wireless communication unit, where the file is stored on a physical hardware element that is external to the wireless communication unit and accessible, say, via a wireless communication network.

Preferably, the cache, at a minimum, includes the location of actual XML documents resolved by the DataManager PATH parameter.

The data obtained by the DataManager function 406 is preferably dynamically updateable. In this regard, it is envisaged that the update mechanism and process may be performed, say, via over-the-air (OTA) programming, by a multi-media card (MMC) or other transport mechanism. The granularity of the update preferably ranges from that of a single individual data file through to a complete replacement of all data files. The DataManager function 406 is therefore preferably able to recognise that one or more files have changed and update the client resource(s) 400 accordingly. This functionality provides for the rapid updating of all client resource data on the wireless communication unit.

Advantageously, the full filename of the root XML document, such as root XML document 300 of FIG. 3, is the only element that needs to be programmed (coded) directly into the DataManager function 406, so as to ensure that it cannot be superseded by any uploading of a replacement document. In this way, the DataManager function 406 always looks for the root file at the same location. For example, if the root file, or device file system in general, becomes corrupted, a default root file can be loaded to that location. The full filename of all subsequent XML documents is not required to be known as it is referred to in the referring XML document. Thus, because the process is location-independent, OTA updates can be reversed by deleting the newly installed files,

assuming that the original files had not been deleted the process would revert back to these files.

Advantageously, the location of resource files that are received and stored following an OTA update, to be retrieved by a client resource, is no longer critical. Using the inventive concepts described herein, it is now possible to parse sections of that resource file and also update sections of the resource file transparent to the client resource and DataManager.

In a further embodiment of the present invention, authentication and encryption of the XML documents, as parsed by the DataManager function 406, is supported. In the context of the preferred embodiment of the present invention, the parsing of data comprises, inter-alia, the interpretation of data conforming to a known syntax. The DataManager function 406 may, for example, make use of a validating parser, such as a SAX2 XML parser, which would fulfil the authentication requirement. This, along with the use of a handset dependent encryption key, i.e. a key specific to the physical hardware of the wireless communication unit 200, provides the requisite encryption and authentication.

SAX is a simple, event-based application programming interface (API) for XML parsers. Although not an official standard, SAX is very much a de facto standard, since it is commonly supported by most XML parsers and is well known in the industry. SAX2 is an extended version of the original SAX 1.0 API for XML parsers, which is now obsolete. SAX 2.0 adds support for XML namespaces,

configurability as well as lexical and Document Type Definition (DTD) information.

To improve performance the parser should be capable of being halted, prior to completion of parsing the complete XML document, i.e. once it has parsed the required information and located the next document on the PATH or the UID of the requested data element, it does not continue to parse the rest of the document unnecessarily.

In addition, the full filename of the root XML document should be coded directly into the DataManager function, so as to ensure that uploading of a replacement document cannot supersede it. In this way, the DataManager function always looks for the root file at the same location, so that, for example, if the root file or device file system in general, becomes corrupted, a default root file can be loaded to that location.

Thus, the schema of the XML data documents uses the XInclude standard. The XInclude functionality is preferably used because it enables arbitrarily large XML documents to be decomposed into an arbitrary number of well-formed smaller XML documents, which has the following benefits:

(i) OTA download is easier (since small files can be downloaded);

(ii) The actual functionally-complete XML documents will be smaller (and thus easier to manage);

(iii) Download of an individual XML document will override the original XML document, but not necessarily need to overwrite the original document because of persistent (as opposed to RAM) memory requirements, thereby retaining the ability to revert to the original document;

(iv) Better performance due to more efficient parsing: a SAX parser will only parse a complete XML document and individual XML documents can be parsed more quickly than a single large XML document for any arbitrarily addressed tag; and

(v) Bounded RAM consumption: memory consumption is mainly determined by the size of individual XML documents.

To further aid in the understanding of the present invention, it is useful to define the following terms:

(i) Logical XML document - An XML document where all XInclude directives have been recursively resolved;

(ii) Actual XML document - A textual file containing a well-formed XML document;

(iii) Root XML document - An actual XML document that is not referenced by XInclude directives from other actual XML documents. There should be only one root XML document;

(iv) Leaf XML document - An actual XML document that does not contain XInclude directives; and

(v) Addressable Tag - A single or compound tag that has an UID attribute.

A logical XML document consists of a hierarchical tree of actual XML documents with a single root XML document and potentially many leaf XML documents. Addressable tags may be placed in any XML document, whether it is an actual, root or leaf XML document.

Preferably the DataManager function does not use the XInclude directive to recursively resolve all actual XML documents into a single, stored in-memory (i.e. RAM), logical XML document. This would imply that the embedded XML parser would have unbounded memory consumption while parsing an arbitrarily large XML document. For devices such as mobile phone handsets, this is unacceptable due to the memory limitations.

Instead the DataManager function preferably uses the XInclude directive to resolve the PATH parameter, starting at the root XML document and separately parsing each referenced XML document in turn to resolve the next element of the given PATH. When the final element of the PATH has been resolved, the leaf XML document will have been identified. The DataManager will then locate the actual **leaf** XML document which will then be parsed for the addressed data, as given by the UID parameter.

An illustrative format of the XML document types is shown in tables 1 through to 4.

Table 1	Root XML Document (e.g. Root.xml)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<root      xmlns:xi="http://www.w3.org/2001/XInclude"
      ref="xyz"
      date="01Jun03"
      ver="abc" >
      <path name="Config">
            <xinclude href="/system/data/config.xml" />
      </path>
      <path name="Content">
            <xinclude href="/system/data content.xml" />
      </path>
</root>
```

Table 2	Config.xml

```
<root      name="Config"
      xmlns:xi="http://www.w3.org/2001/XInclude"
      ref="xyz"
      date="01Jun03"
      ver="abc" >
      path name="Messaging">
            <xinclude href="/system/data/Messaging.xml" />
      </path>
      <path name="Themes">
            <xinclude href="/system/data/Themes.xml" />
      </path>
</root>
```

Table 3	Themes.xml

```
<root      name="Themes"
      xmlns:xi="http://www.w3.org/2001/XInclude"
      ref="xyz"
      date="01Jun03"
      ver="abc" >
      <path name="Default">
            <xinclude href="/system/data/Default.xml" />
      </path>
      <path name="Operator">
            <xinclude href="/system/data/Operator.xml" />
      </path>
      <path name="User">
            <xinclude href="/system/data/User.xml" />
      </path>
</root>
```

Table 4	Default.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<root name="Default"
      ref="xyz"
      date="01Jun03" ver="abc" >
      <ATag uid="11111111">
            <data>whatever data1</data>
      </ATag>
      <ATag uid="22222222">
            <data>whatever data2</data>
      </ATag>
</root>
```

In this example, the data is contained within the "leaf" Default.htm, as illustrated in Table 4.

Referring now to FIG. 5, a flow-chart 500, illustrates a method of retrieving a data element requested from a DataManager function, such as DataManager function 406 of FIG. 4, by a client resource of the wireless communication unit, according to a preferred embodiment of the present invention.

To further clarify the operation of the DataManager function, we refer again to FIG. 4 where the file default.xml is assumed to contain, say, the text:

```
<root name="Default"
      ref="xyz"
      date="01Jun03" ver="abc" >
      <ATag uid="11111111">
            <data>whatever data1</data>
      </ATag>
      <ATag uid="22222222">
            <data>whatever data2</data>
      </ATag>
</root>
```

A client resource, for example a process running on a mobile phone handset, requests data from the DataManager function by passing the following PATH and UID parameters 501 of the required data to an interface (say interface 404 of FIG. 4) offered by the DataManager function.

```
PATH = "/Root/Config/Themes/Default"
UID = "11111111"
```

The DataManager function receives the PATH and UID information in step 502 from the interface, and retrieves the root file of the PATH, say for the previous example, the file Root.xml. Preferably there is a single root file, which the DataManager function always starts with, and knows the location of.

Having retrieved the root file, the DataManager function determines whether the retrieved file (i.e. root.xml) is the last file in the PATH, as shown in step 504. In the above example of FIG. 3, the retrieved file is not the last file in step 504 and step 506 since "Config", "Themes" and "default", follows it. The PATH contains file aliases, rather than complete file names, which are preferably limited in their length. Advantageously, this avoids the use of potentially long file names and extensions within the PATH. As the root file is not the last file in the PATH, the DataManager parses the file in step 508 to locate the next file, namely "Config". In the above example of FIG. 3, the location of the "Config" file is identified by the lines:

```
<path name="Config">
"<xinclude href="/system/data/config.xml"/>"
</path>
```

Where the tags <path name="Config"> and </path> denote that the information therebetween represents the location of the "Config" file. As can be seen in this example, the "Config" file is located at:

"/system/data/config.xml".

Having determined the location of the next file, the DataManager function retrieves the next file, and once again determines whether it is the last file in the PATH, as shown in step 504 and step 506. If it is not the last file in step 506, the DataManager function parses the retrieved file to determine the location of the next file in step 508, and then retrieves that next file in step 512.

This process continues until the DataManager function has retrieved the last file in the PATH, as shown in step 512. For the above example, this is the "Default" file. When the DataManager function determines that it has retrieved the last file in the PATH, as shown in step 506, the DataManager function parses the file to determine the required data, as shown in step 510, using the UID.

In the above example, the required data is determined by the lines:

```
<ATag uid="11111111">
<data>whatever data1</data>
</ATag>
```

The tags <ATag uid="11111111"> and </ATag> denote that the information therebetween is the data relating to the UID "11111111".

Having retrieved the required data, the DataManager function returns the data to the entity that initially requested it, as shown in step 514, via the interface, for example as an instantiated object.

Preferably, the UIDs and file aliases follow a convention limiting their length, and provide a logical naming standard.

Thus, in this manner, the preferred embodiment of the present invention provides a number of advantages over current mechanisms for retrieving data, for example in wireless communication units such as mobile phones, PDA's, etc. In particular, it is proposed that, by using XInclude and Xpath, a plurality of XML files can be organised in a cogent and distinct manner, as opposed to one long file. The maximum size of each file is preferably strictly limited.

This means that:

(i) Time is not wasted parsing large files when trying to locate date; and

(ii) The DataManager function requires less RAM for performing its operations because the size of the files it is parsing are smaller (a particularly advantageous feature for a mobile phone application).

Furthermore, the inventive concepts described herein conform to the XML standards and find particular applicability when it is necessary for reasons of reliability and security to develop a mobile phone using a stable OS with limited flexibility. The inventive concepts are particularly advantageous where a number of resources of the wireless communication unit are configurable and are required to be flexible and updateable during the lifetime of the wireless communication unit.

An example of the use of XML files according to the present invention is to specify the contents of a 'theme'. For communication units such as mobile phones, a range of themes may be provided, for example, a default theme, Network Operator themes, and user themes. The default theme can be stored in, for example, a ROM image on the mobile phone handset. In this way, if a deep 'restore factory settings' (RFS) is performed, this theme shall not be destroyed and can then be used to recreate a themes database.

It is also envisaged that a Network Operator theme can be provided, to support the specific requirements of a Network Operator. The theme may be stored in, for example, a flash file system such that a user-level RFS operation will not destroy it.

It is also envisaged that user themes can be supplied to the mobile phone handset in any appropriate manner. For example, the user may create a new theme from scratch, or alternatively download a new theme from another device. In addition to supplying the theme XML document, it is also required to provide any new graphics (Bitmap, JPEG, Animations) and tones, such as ring-tones, beeps, message alerts.

It is within the contemplation of the present invention that the type of data that could be included within a theme-based XML file may comprise, for example, one or more of the following:

(i) Colour schemes (e.g. background, text, etc.);

(ii) Ring tones and other audio alerts;

(iii) Graphics (e.g. icons, background images, etc.); and

(iv) WEB/WAP links, etc.

Preferably, a theme-based application (not shown) manages a themes database, depending on the particular theme that is to be used. In this regard, the theme application maintains a record of the particular theme that the user has selected to use, and accordingly for that particular theme requests the required data from the DataManager function.

Due to the nature of XML, the theme application is also preferably able to obtain a list of available themes from the DataManager function, and in this way provide a list of themes to the user. In this way, new themes can easily be added, and unwanted ones removed.

A further advantage provided by the inventive concepts of the present invention is that the use of XML files allows themes to be exchanged with other, say, Smartphone users, simply by exchanging the relevant XML file. In addition, new themes may be downloaded in a standard format using, say, over-the-air (OTA) or other transport mechanisms.

Since XML is self-describing data, it is universal, i.e. there is no requirement for explicit or implied context to the data. Therefore, all textual data elements required to describe a complete mobile phone handset can be effected in XML, including the data used by the mobile phone handset. Thus, it is envisaged that automated transfer technology can be used to take XML data from a

compliant XML document stored in an electronic device such as a mobile phone, to program the electronic device.

An application (i.e. DataManager) working with an embedded standard XML parser can therefore be designed to utilise desired data, so long as the XML document is configured in a manner that allows the client resource to access the appropriate sections of the XML document. The application is preferably matched to an appropriately re-configured XML document (i.e. the XML document is configured in accordance with the preferred embodiment of the present invention in that it comprises a root plus any other XML sub-documents). Advantageously, the appropriately re-configured XML document is still compliant with existing XML documents.

A yet further advantage of using XML files is that the files are extensible, as will be increasingly required in the future. In other words, because of the use of tags in XML, adding further data, tags, etc. to any XML file will not affect the locating of existing data. In this way, it is simple to add/remove data etc. without significantly affecting existing data. Importantly, clients requesting data are not required to know about changes to the XML files, as long as the UIDs and PATHs are not changed.

Thus, existing data management mechanisms that are currently available, particularly on mobile phone platforms, fail to adequately address, inter-alia, configurability requirements. The inventive concepts of the present invention, for example when applied to data

management applications of future mobile phones, tend to provide one or more of the following advantages:

(i) The fundamental process of handset re-configuration is easy, quick and amenable to automation;

(ii) There is a unified method of defining all configurable data, whether configuration (e.g. language, colour schemes, etc.) or content (e.g. images, data, etc.);

(iii) There is a high degree of granularity of configuration data, i.e. relatively small amounts of data should be able to be replaced without consequent effects on the greater amount of unchanged data;

(iv) Management of configuration versions is straightforward, without adverse effects on the compatibility or stability of the handset design;

(v) The data management operation is able to support a single standardised parser to avoid a need for multiple data parsers to interpret variously formatted, and potentially incompatible, data elements;

(vi) An automated transfer can be used to take XML data from a compliant XML document stored in an electronic device, to program the device; and

(vii) There should be efficiency of memory use, data size and processor cycles.

Whilst the specific and preferred implementations of the embodiments of the present invention are described above, it is clear that one skilled in the art could readily apply variations and modifications of such inventive concepts.

Thus, a highly configurable wireless communication unit, a method of arranging configuration data for the wireless communication unit, and a method for retrieving one or more data elements associated with a hardware and/or a software resource of a wireless communication unit have been described, where the aforementioned disadvantages with prior art arrangements have been substantially alleviated.

CLAIMS

1. A method (500) for retrieving one or more data elements associated with a hardware and/or a software resource (400) in a resource-constrained electronic device (200), wherein the one or more data element(s) (304) is embedded in a textual data file (306) by means of a mark-up language, the method characterised by the steps of:

forming multiple data files (306) into a substantially single hierarchical textual file (300), such that a location of a number of respective data files within the hierarchical textual file (300) is described by a respective location identifier;

parsing the at least one location identifier to identify a location of at least one data file; and

retrieving (502) only the at least one data file from the multiple data files.

2. A method (500) for retrieving one or more data elements according to Claim 1 further characterised in that the mark-up language conforms to the XML standard.

3. A method (500) for retrieving one or more data elements according to Claim 1 or Claim 2 further characterised in that each data file (306) is an XML text file.

4. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised in that multiple data files (306) are incorporated into a Root file (300), for example by a method conforming to an Xinclude standard.

5. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised in that the location identifier (302) for each data file (306) is generated by a method conforming to the XPATH standard.

6. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised in that the location identifier comprises a unique identifier and/or a file PATH (302) to identify a location of a data file from the multiple data files (306).

7. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised in that the resource-constrained electronic device is a wireless communication unit (200).

8. A method (500) for retrieving one or more data elements according to Claim 7, wherein a DataManager function (406) and a client resource reside on the wireless communication unit (200), the method further characterised by the steps of:

receiving a location identifier (302) of a data element (304), by the DataManager function (406) from a client resource (400), such that the DataManager (406) performs the steps of:

parsing a Root Data file (300);

locating one or more data elements (304); and

returning the one or more data elements (304) to the client resource (400).

9. A method (500) for retrieving one or more data elements according to Claim 8 when dependent upon Claim 6, wherein a number of client resources (400) maintain a list of File PATH (302) and unique identifier data of data elements (304) relevant to a functionality of the respective client resource(s) (400).

10. A method (500) for retrieving one or more data elements according to Claim 8 or Claim 9 further characterised by the steps of:

recognising changes to a data structure within the electronic device; and

updating client resources (400) of the device (200).

11. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised by the steps of:

creating a copy of a single data-file (306) being parsed, for example beginning with a Root file (300) and comprising an address of a next file on a PATH (302); and

discarding a previously stored data file.

12. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised in that the data files (306) being parsed are encrypted and authenticated prior to parsing.

13. A data parser adapted to perform the steps of any of the preceding method Claims.

14. A resource-constrained electronic device (200), for example a wireless communication unit, adapted to perform the steps of any of the preceding method Claims.

15. A resource-constrained electronic device (200) comprising:

a memory element (216) comprising one or more data elements associated with a hardware and/or a software resource (400) in the resource-constrained electronic device (200), wherein the one or more data element(s) (304) is embedded by means of a mark-up language in a file within a substantially single hierarchical textual file (300) of multiple files, such that a location of a respective data file within the hierarchical textual file (300) is described by a corresponding location identifier;

a signal processing function (208), operably coupled to the memory element, for accessing or storing data in the memory element;

wherein the resource-constrained electronic device (200) is characterised by:

a DataManager function (406), operably coupled to or contained within the signal processing function (208), which parses at least one location identifier in order to retrieve only one or more data files from the multiple data files.

16. A resource-constrained electronic device (200) according to Claim 15 wherein the location identifier comprises a unique identifier or a file PATH (302) to identify a location of a number of data files within the hierarchy.

17. A resource-constrained electronic device (200) according to Claim 15 or Claim 16 further characterised in that the mark-up language conforms to the XML standard.

18. A resource-constrained electronic device (200) according to any of preceding Claims 15 to 17 further characterised in that each data file (306) is an XML text file.

19. A resource-constrained electronic device (200) according to any of preceding Claims 15 to 18 further characterised in that multiple data files (306) are incorporated into a Root file (300) conforming to an Xinclude standard.

20. A resource-constrained electronic device (200) according to any of preceding Claims 16 to 19 further characterised in that the File PATH (302) for each data file (306) conforms to the XPATH standard.

21. A resource-constrained electronic device (200) according to any of preceding Claims 15 to 20 further characterised in that the resource-constrained electronic device is a wireless communication unit (200).

22. A resource-constrained electronic device (200) according to any of preceding Claims 15 to 21, further characterised by at least one client resource (400) operably coupled to the DataManager function (406) such that the at least one client resource (400) provides the file PATH (302) and unique identifier of a data element (304) to the DataManager function (406), which parses a

Root Data file (300); locates the data element (304); and returns the data element (304) to the client resource (400).

23. A resource-constrained electronic device (200) according to Claim 22, further characterised in that a number of client resources (400) maintain a list of File PATH (302) and unique identifier data of data elements (304) relevant to a functionality of the respective client resource(s) (400).

24. A resource-constrained electronic device (200) according to Claim 22 or Claim 23, further characterised in that the DataManager function (406) recognises a change to a data structure within the memory element and updates at least one client resource(s) (400).

25. A resource-constrained electronic device (200) according to any of preceding Claims 15 to 24, further characterised in that the DataManager function (406) creates a copy of a single data-file (306) being parsed, and discards a previously stored data file.

26. A resource-constrained electronic device (200) according to any of preceding Claims 15 to 25, further characterised in that the data files (306) are decrypted and authenticated by the DataManager function (406) prior to parsing.

27. A storage medium storing processor-implementable instructions for controlling one or more processors to perform the steps of any of Claims 1 to 12 or one or more

processors in the resource-constrained electronic device according to any of preceding Claims 14 to 26.

AMENDED CLAIMS
[received by the International Bureau on 07 November 2005 (07.11.05);
original claims 1-27 replaced by new claims 1-24 (6 pages)]

+ STATEMENT UNDER ARTICLE 19(1) PCT

1. A method (500) for retrieving one or more data elements associated with a hardware and/or a software resource (400) in a resource-constrained electronic device (200), wherein the one or more data element(s) (304) is embedded in a textual data file (306) by means of a mark-up language, the method characterised by the steps of:

forming multiple data files (306) into a substantially single hierarchical textual file (300) conforming to an Xinclude standard, such that a location of a number of respective data files within the hierarchical textual file (300) is described by a respective location identifier;

parsing the at least one location identifier to identify a location of at least one data file; and

retrieving (502) only the at least one data file from the multiple data files.

2. A method (500) for retrieving one or more data elements according to Claim 1 further characterised in that each data file (306) is an XML text file.

3. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised in that multiple data files (306) are incorporated into a Root file (300).

4. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised in that the location identifier (302) for

each data file (306) is generated by a method conforming to the XPATH standard.

5. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised in that the location identifier comprises a unique identifier and/or a file PATH (302) to identify a location of a data file from the multiple data files (306).

6. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised in that the resource-constrained electronic device is a wireless communication unit (200).

7. A method (500) for retrieving one or more data elements according to Claim 6, wherein a DataManager function (406) and a client resource reside on the wireless communication unit (200), the method further characterised by the steps of:

receiving a location identifier (302) of a data element (304), by the DataManager function (406) from a client resource (400), such that the DataManager (406) performs the steps of:

parsing a Root Data file (300);

locating one or more data elements (304); and

returning the one or more data elements (304) to the client resource (400).

8. A method (500) for retrieving one or more data elements according to Claim 8 when dependent upon Claim 5, wherein a number of client resources (400) maintain a list of File PATH (302) and unique identifier data of

data elements (304) relevant to a functionality of the respective client resource(s) (400).

9. A method (500) for retrieving one or more data elements according to Claim 7 or Claim 8 further characterised by the steps of:

recognising changes to a data structure within the electronic device; and

updating client resources (400) of the device (200).

10. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised by the steps of:

creating a copy of a single data-file (306) being parsed, for example beginning with a Root file (300) and comprising an address of a next file on a PATH (302); and

discarding a previously stored data file.

11. A method (500) for retrieving one or more data elements according to any preceding Claim further characterised in that the data files (306) being parsed are encrypted and authenticated prior to parsing.

12. A data parser adapted to perform the steps of any of the preceding method Claims.

13. A resource-constrained electronic device (200), for example a wireless communication unit, adapted to perform the steps of any of the preceding method Claims.

14. A resource-constrained electronic device (200) comprising:

a memory element (216) comprising one or more data elements associated with a hardware and/or a software resource (400) in the resource-constrained electronic device (200), wherein the one or more data element(s) (304) is embedded by means of a mark-up language in a file within a substantially single hierarchical textual file (300) of multiple files, such that a location of a a number of respective data files within the hierarchical textual file (300) conforming to an Xinclude standard is described by a corresponding location identifier;

a signal processing function (208), operably coupled to the memory element, for accessing or storing data in the memory element;

wherein the resource-constrained electronic device (200) is characterised by:

a DataManager function (406), operably coupled to or contained within the signal processing function (208), which parses at least one location identifier in order to to identify a location of at least one data file and retrieve only the at least one data file from the multiple data files.

15. A resource-constrained electronic device (200) according to Claim 14 wherein the location identifier comprises a unique identifier or a file PATH (302) to identify a location of a number of data files within the hierarchy.

16. A resource-constrained electronic device (200) according to any of preceding Claims 14 to 15 further characterised in that each data file (306) is an XML text file.

17. A resource-constrained electronic device (200) according to any of preceding Claims 14 to 16 further characterised in that the File PATH (302) for each data file (306) conforms to the XPATH standard.

18. A resource-constrained electronic device (200) according to any of preceding Claims 14 to 17 further characterised in that the resource-constrained electronic device is a wireless communication unit (200).

19. A resource-constrained electronic device (200) according to any of preceding Claims 14 to 18, further characterised by at least one client resource (400) operably coupled to the DataManager function (406) such that the at least one client resource (400) provides the file PATH (302) and unique identifier of a data element (304) to the DataManager function (406), which parses a Root Data file (300); locates the data element (304); and returns the data element (304) to the client resource (400).

20. A resource-constrained electronic device (200) according to Claim 19, further characterised in that a number of client resources (400) maintain a list of File PATH (302) and unique identifier data of data elements (304) relevant to a functionality of the respective client resource(s) (400).

21. A resource-constrained electronic device (200) according to Claim 19 or Claim 20, further characterised in that the DataManager function (406) recognises a

change to a data structure within the memory element and updates at least one client resource(s) (400).

22. A resource-constrained electronic device (200) according to any of preceding Claims 14 to 21, further characterised in that the DataManager function (406) creates a copy of a single data-file (306) being parsed, and discards a previously stored data file.

23. A resource-constrained electronic device (200) according to any of preceding Claims 14 to 22, further characterised in that the data files (306) are decrypted and authenticated by the DataManager function (406) prior to parsing.

24. A storage medium storing processor-implementable instructions for controlling one or more processors to perform the steps of any of Claims 1 to 11 or one or more processors in the resource-constrained electronic device according to any of preceding Claims 13 to 23.

Statement under Article 19(1) PCT

Re: Patent Application PCT/EP2005/051151 in the name of Sendo International Limited.

Amendments and support therefor:

(i) Claim 1 and Claim 14 (re-numbered from Claim 15) have been amended to incorporate the Xinclude feature. Support for this feature can be found throughout the Specification as originally filed, not least Claim 4.
(ii) Claim 2, Claim 17 and Claim 20 have been deleted without prejudice.
(iii) Subsequent Claims and their dependencies have been re-numbered accordingly.
(iv) Features of Claim 1 have been incorporated into Claim 14 (re-numbered from Claim 15) to ensure consistency.

FIG. 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited with XML Spy v4.2 -->

<GROUP ELEMENT>

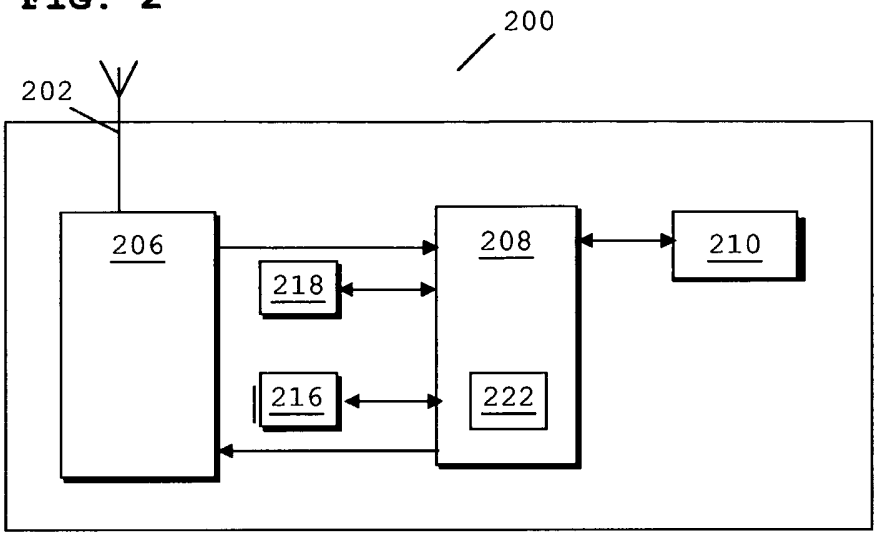
  <INDIVIDUAL ELEMENT> first element
       <DATA1>some data</DATA1>
       <DATA2>some more data</DATA2>
  </INDIVIDUAL ELEMENT>

  <INDIVIDUAL ELEMENT> second element
       <DATA1>some data</DATA1>
       <DATA2>some more data</DATA2>
  </INDIVIDUAL ELEMENT>

  <INDIVIDUAL ELEMENT> third element
       <DATA1>some data</DATA1>
       <DATA2>some more data</DATA2>
  </INDIVIDUAL ELEMENT>
:
:
  <INDIVIDUAL ELEMENT> last element
       <DATA1>some data</DATA1>
       <DATA2>some more data</DATA2>
  </INDIVIDUAL ELEMENT>

</GROUP ELEMENT>
```

FIG. 2


200
202
206
208
210
218
216
222

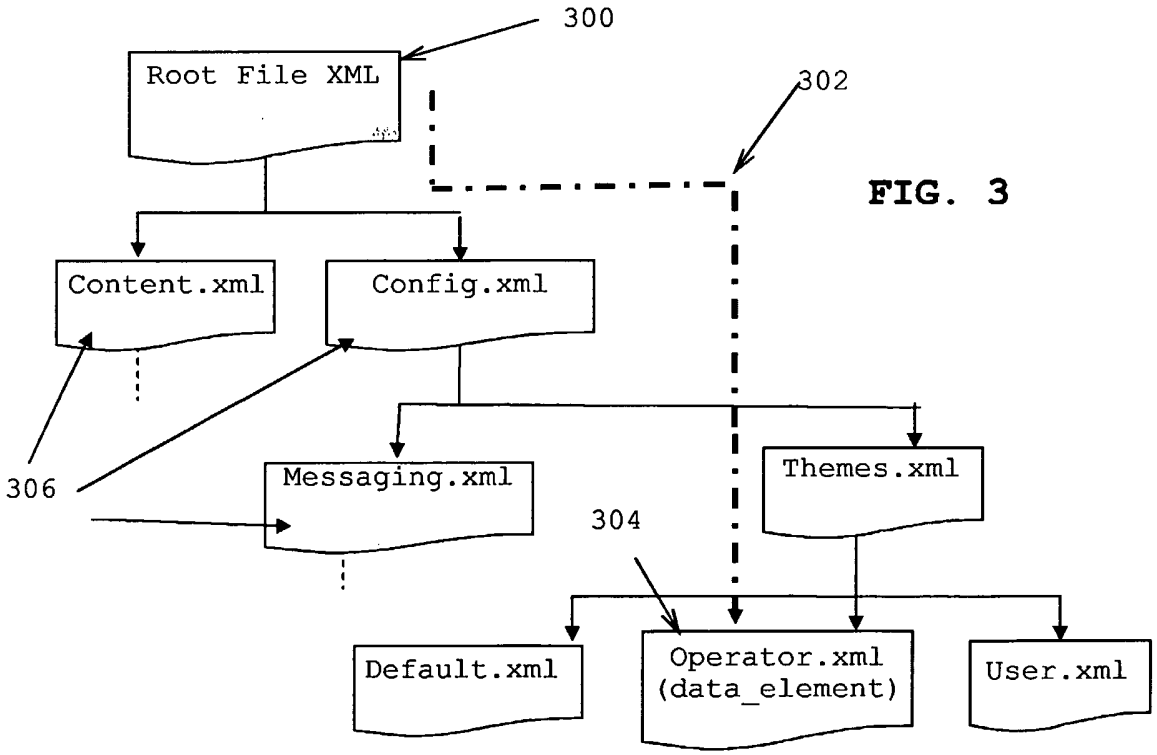

300
Root File XML
302
Content.xml
Config.xml
306
Messaging.xml
Themes.xml
304
Default.xml
Operator.xml
(data_element)
User.xml

FIG. 3

FIG. 4

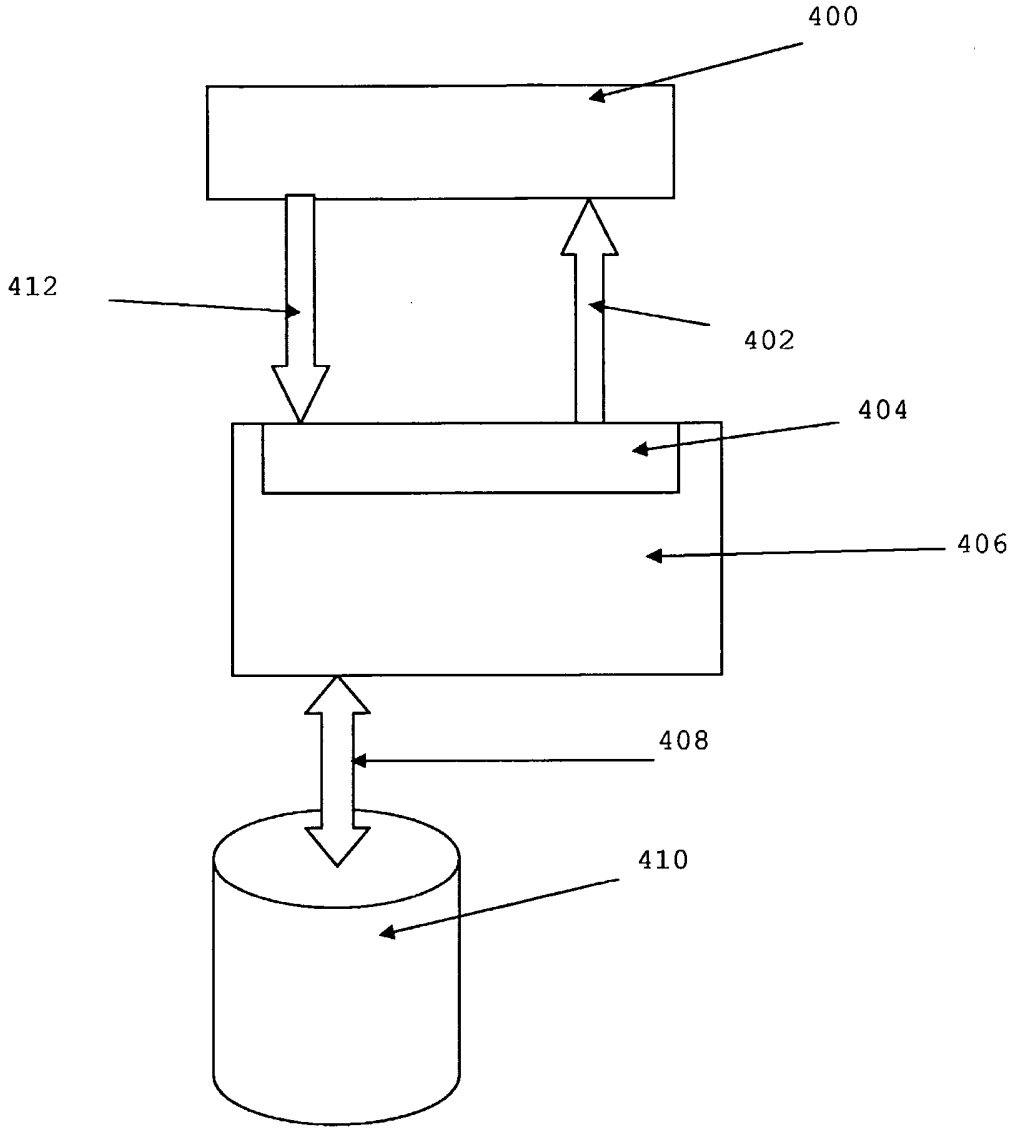
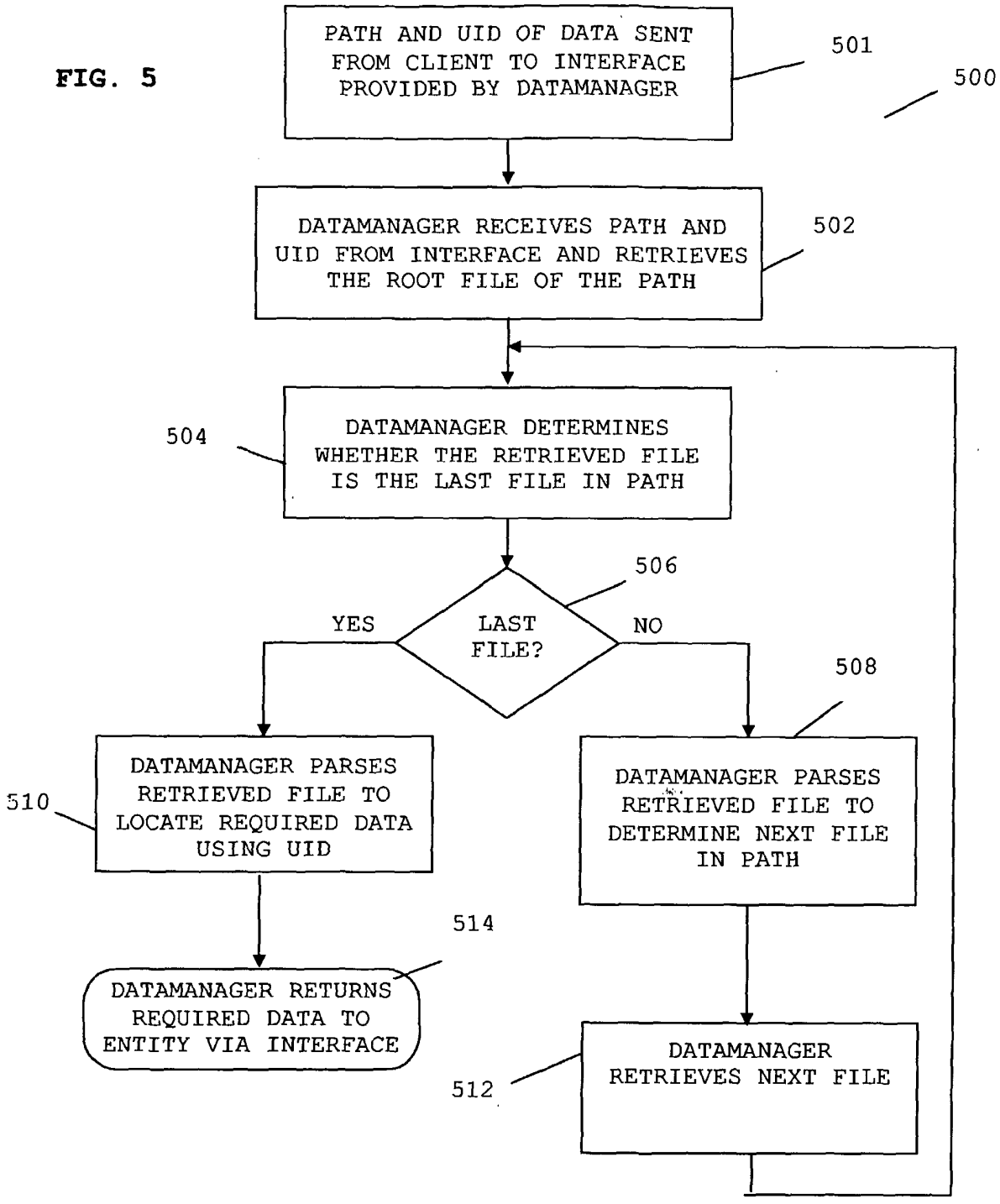


FIG. 5


PATH AND UID OF DATA SENT FROM CLIENT TO INTERFACE PROVIDED BY DATAMANAGER
501
500
DATAMANAGER RECEIVES PATH AND UID FROM INTERFACE AND RETRIEVES THE ROOT FILE OF THE PATH
502
DATAMANAGER DETERMINES WHETHER THE RETRIEVED FILE IS THE LAST FILE IN PATH
504
506
LAST FILE?
YES
NO
508
DATAMANAGER PARSES RETRIEVED FILE TO LOCATE REQUIRED DATA USING UID
510
DATAMANAGER PARSES RETRIEVED FILE TO DETERMINE NEXT FILE IN PATH
514
DATAMANAGER RETURNS REQUIRED DATA TO ENTITY VIA INTERFACE
DATAMANAGER RETRIEVES NEXT FILE
512

INTERNATIONAL SEARCH REPORT

International Application No
PCT/EP2005/051151

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/30 G06F9/445

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	SETH M: "XInclude - Introduction and Usage" DEVELOPER.COM, 'Online! 29 January 2004 (2004-01-29), pages 1-8, XP002341991 Retrieved from the Internet: URL:http://www.developer.com/tech/print.php/3305511> 'retrieved on 2005-08-22! page 2, line 2 - line 21; figure 1 page 3, line 11 - line 15; figure 2 page 5, line 12 - line 30; figure 3 page 6, line 7 - line 31 page 7, line 1 - line 10 ----- -/--	1-27

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
24 August 2005	06/09/2005

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Pose Rodríguez, J

INTERNATIONAL SEARCH REPORT

International Application No
PCT/EP2005/051151

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2003/023707 A1 (RYAN FINTAN) 30 January 2003 (2003-01-30) abstract paragraph '0014! - paragraph '0015! paragraph '0018! paragraph '0056! - paragraph '0058!; figures 2,4,5 paragraph '0063! - paragraph '0066!; figures 6,7	1-27
X	US 2003/182405 A1 (GAVA FABIO M ET AL) 25 September 2003 (2003-09-25) paragraph '0006! - paragraph '0007! paragraph '0026! - paragraph '0028!; figures 2B,2C paragraph '0030! - paragraph '0034!; figure 3	1-27
A	EP 1 477 915 A (COGNOS INCORPORATED) 17 November 2004 (2004-11-17) paragraph '0003!; figure 2	6,16

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/EP2005/051151

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2003023707 A1	30-01-2003	NONE	
US 2003182405 A1	25-09-2003	NONE	
EP 1477915 A	17-11-2004	CA 2429095 A1	16-11-2004
		CA 2467315 A1	16-11-2004
		EP 1477915 A2	17-11-2004
		US 2005033725 A1	10-02-2005