



GRAPPLE

D4.3 Version: 1.0

Device and network adaptation model

Document Type	Deliverable
Editor(s):	Lucia Oneto (Gilabs)
Author(s):	Stefania Cantini (Gilabs), Andrea Lorenzon (Gilabs), Lucia Oneto (Gilabs)
Reviewer(s):	Avi Naim (UCAM), Frederic Minne (UCL)
Work Package:	WP4
Due Date:	31-08-2009
Version:	1.0
Version Date:	31-08-2009
Total number of pages:	26

Abstract: This report examines the state-of-the-art in standards and technologies that need to be implemented or integrated for content device adaptation. The focus of this document is to use the available technology to provide a device and network adaptation module, able to perform content adaptation based on device.

Keyword list: device adaptation, content adaptation, User Agent Profile, mobile, profile

Summary

This document presents first considerations on the Device and network adaptation model to be designed and implemented in the scope of WP4.

Authors

Person	Email	Partner code
Andrea Lorenzon	a.lorenzoni@giunttilabs.com	GILABS
Lucia Oneto	l.oneto@giunttilabs.com	GILABS
Stefania Cantini	s.cantini@giunttilabs.com	GILABS

Table of Contents

SUMMARY	2
AUTHORS	2
TABLE OF CONTENTS	2
TABLES AND FIGURES.....	3
LIST OF ACRONYMS AND ABBREVIATIONS	3
1 INTRODUCTION	5
1.1 Task and Deliverable Description	5
2 STATE OF THE ART	5
2.1 Introduction of network and device profile management.....	5
2.1.1 W3 CC/PP	5
2.1.2 UAProf	8
2.1.3 DELI.....	11
2.1.4 WURFL.....	13
2.2 Approach	13
2.2.1 Extensible Style sheet Language (XSL)	14
2.2.2 Device and network features.....	18
2.3 Device and network adaptation infrastructure	20
3 REFERENCES	22
APPENDIX	23

Tables and Figures

List of Figures

Figure 1: HTTP request processing (<http://www.w3.org/TR/CCPP-struct-vocab/#Outline>). 7

Figure 2: RDF graph referred to the example above. 10

Figure 3: Adaptation process. 13

Figure 4: XSL Two Processes: Transformation & Formatting. 15

Figure 5: Transform to Another Vocabulary..... 15

Figure 6: Build the XSL Formatting Object Tree. 16

Figure 7: Refine the Formatting Object Tree. 17

Figure 8: Generate the Area Tree. 18

Figure 9: Summary of the Process..... 18

Figure 10: GRAPPLE adaptation process. 20

Figure 11: GRAPPLE adaptation process details..... 21

List of Acronyms and Abbreviations

CC/PP	Composite Capabilities/Preferences Profiles
CSS	Cascading Style Sheet
GALE	GRAPPLE Adaptive Learning Environment
GDA	GRAPPLE Device Adaptation
GRAPPLE	Generic Responsive Adaptive Personalized Learning Environment
GUMF	GRAPPLE User Model Framework
HTTP	Hyper Text Transfer Protocol
I/O	Input/Output
LMS	Learning Management System
JSP	Java Server Page
PC	Personal Computer
PDA	Personal Digital Assistant
QoS	Quality of Service
R&D	Research & Development
RDF	Resource Description Framework
RTD	Research and Technology Development
UA	User Agent
UM	User Model
URL	Universal Resource Locator
WP	Work Package
WURFL	Wireless Universal Resource File
XML	eXtensible Markup Language

XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

1 Introduction

The purpose of this deliverable is to report on the RTD work carried out in order to characterize the challenges in automating the distribution of adaptive courses also to mobile clients.

An account of the latest evolution of the relevant profiling standards such as CC/PP and Deli is given at the outset. Similarly the design of the distribution of adaptive courses to mobile devices is described and the problems and the work performed to resolve them are outlined. The report ends with a description of the work to be done in the second year of the project (M12-M24).

1.1 Task and Deliverable Description

T4.3 Definition of device and network adaptation models (GILABS)

This task deals with the definition of suitable adaptation models, in order to offer to the user a proper presentation of the needed learning information and services while preserving their semantic value and features. This will imply identifying abstract descriptions of output devices and network features, as well as defining languages for their descriptions. The adoption of an extension to existing standards (e.g. W3C CC/PP) will be investigated. The adaptation model will include also a limited set of adaptation rules that can be applied without full access to the user model (that may be off-line).

D4.3 Device and network adaptation model (GILABS, M9)

This report will contain the description of the adaptation model, resulting from T4.3.

2 State of the art

As the number and variety of devices connected to the Internet grows, there is a corresponding increase in the need to deliver content that is tailored to the capabilities of different devices. As part of a framework for content adaptation and contextualisation, a general purpose profile format is required that can describe the capabilities of a user agent and preferences of its user.

2.1 Introduction of network and device profile management

2.1.1 W3 CC/PP

CC/PP stands for **Composite Capabilities/Preferences Profiles** [4], and is a way to specify what exactly a user agent (web browser) is capable of doing. This allows for sophisticated content negotiation techniques between web servers and clients, to produce optimized XML-based markup for display and use on a wide variety of web user agents.

A CC/PP profile is a description of device capabilities and user preferences that can be used to guide the adaptation of content presented to the particular device. Here profile does not refer to a subset of a particular specification, for example the CSS Mobile profile, but refers to the document(s) exchanged between devices that describe the capabilities of a device. A CC/PP profile is broadly constructed as a two level hierarchy: a profile has a number of components and each component has a number of attributes.

As an example, a CC/PP profile of device “Sony Ericsson T39” in given is the Appendix.

The CC/PP proposal describes an interoperable encoding for capabilities and preferences of user agents, specifically web browsers. The proposal is also intended to support applications other than browsers, including email, calendars, etc. Support for peripherals like printers and fax machines will require other types of attributes such as type of printer, location, Postscript support, colour, etc. We believe an XML/RDF based approach would be suitable [2][3]. However, metadata descriptions of devices like printers or fax machines may use a different scheme. The CC/PP proponents state that effort has been made to provide interoperability with other important proposals. However the recent trend for most mobile sets has been increasingly towards adopting UAProf (2.1.2) as an alternative to CC/PP.

The basic data model for a CC/PP is a collection of tables. Though RDF makes modelling a wide range of data structures possible, it is unlikely that this flexibility will be used in the creation of complex data models for profiles. In the simplest form each table in the CC/PP is a collection of RDF statements with simple, atomic properties. These tables may be constructed from default settings, persistent local changes or temporary changes made by a user. One extension to the simple table of properties data model is the notion of a separate, subordinate collection of default properties. Default settings might be properties defined by the vendor. In the case of hardware the vendor often has a very good idea of the physical properties of any given model of the product. However, the current owner of the product may be able to add options, such as memory or persistent store or additional I/O devices that add new properties or change the values of some original properties. These would be persistent local changes. An example of a temporary change would be turning the sound on or off.

2.1.1.1 W3C's CC/PP Working Group

The CC/PP Working Group was chartered in summer of 1999 and is an outgrowth of the W3C's Mobile Access Interest Group. The first meeting was held in Stockholm, Sweden, on the 23rd and 24th of September, and was hosted by Ericsson. Johan Hjelm of the W3C (and Ericsson) is the chair of the CC/PP Working Group.

Companies and organizations currently participating in the CC/PP working group include:

- Ericsson
- Fujitsu Laboratories
- HTML Writers Guild
- IBM
- IETF
- Interleaf
- Matsushita Electric Industrial Co.
- Nokia
- Nortel Networks
- SAP AG
- SBC Technology Resources
- Sun
- T-Mobile.

For more information about CC/PP please refer to its homepage on:
<http://www.w3.org/Mobile/CCPP/>.

2.1.1.2 Outline of request processing in HTTP

CC/PP is envisaged to be used with HTTP [11] in the following fashion.

(This is not a protocol specification, just an indication of the kind of information flow envisaged. Defining a protocol to convey CC/PP information is a separate effort [5]).

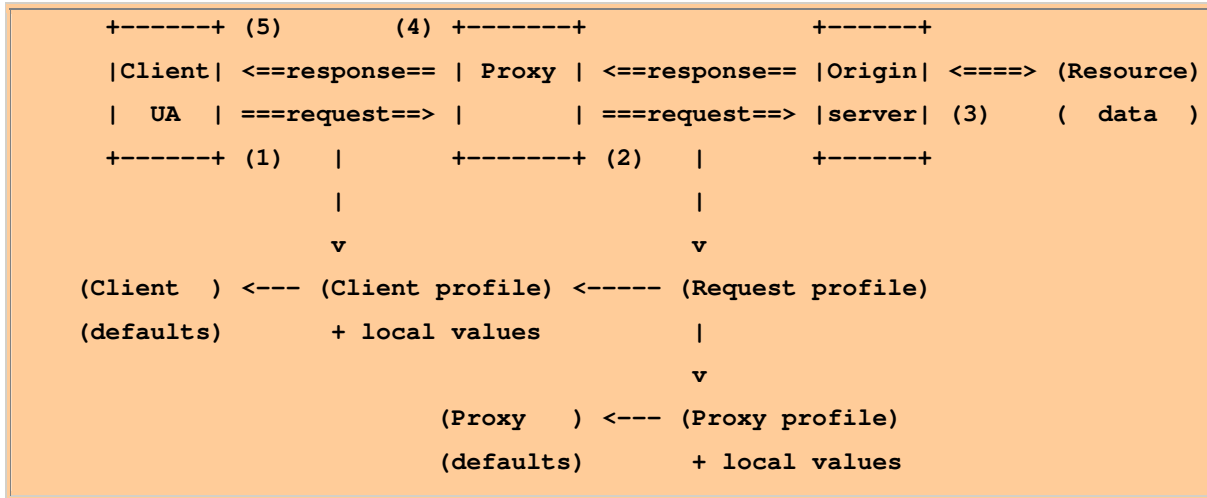


Figure 1: HTTP request processing (<http://www.w3.org/TR/CCPP-struct-vocab/#Outline>).

1. The client sends an HTTP request, with an accompanying CC/PP client profile. The client profile may contain references to default profiles describing a range of common capabilities for the client concerned (e.g. a particular computer/operating system/browser combination, or a particular model of mobile device), and values that are variations from the default profile.
2. The HTTP request may pass through a firewall/proxy that (a) imposes constraints on the kinds of content that can be accessed, or (b) can adapt other forms of content to the capabilities of the requesting client. This proxy extends the CC/PP profile with a description of these constraints and adaptations, and sends this with the HTTP request on to the origin server. The request may pass through several such proxies.
3. The origin server receives the request and interprets the CC/PP profile. It selects and/or generates content that matches the combined proxy and client capabilities described in the profile. This is sent to the last proxy in the request chain in an HTTP response.
4. If required, the proxy applies any content adaptations, and any other functions it is designed to perform. The resulting response and content is passed back toward the requesting client.
5. The client receives the HTTP response and presents the content it contains.

NOTE: There is some overlap between CC/PP and the various HTTP accept-* headers. A protocol specification for using CC/PP with HTTP must indicate how HTTP 'accept-*' headers may be used, and how they interact with CC/PP profiles.

For CC/PP to work, several parties must participate. The good news is that everything's invisible to users unless they want to transmit preferences not found in the default profile, in which case some user configuration is required. To enable interoperability between web servers and access mechanisms, and to facilitate development of device independent web applications, a set of Java APIs has been developed for processing CC/PP information [12].

The parties involved and their respective roles are as follows:

1. **Device vendors** are responsible for authoring the reference profile describing the base capabilities of each type of device. In addition, vendors must provide browser support so that applications can attach a CC/PP profile to the HTTP request. Finally, vendors must ensure that all profiles are in valid CC/PP or UAProf syntax.

2. **Content authors** create text, images, sounds, and other media that can be read, viewed, or listened to. The authors are responsible for creating multiple versions of the same content, suited to multiple devices. A widely used approach is to write content in XML and provide XSLT stylesheets that transform content for various delivery contexts.
3. **Application developers** write the Java code that retrieves different content based on the delivery context. Such applications can be written using JavaServer Pages (JSPs) or servlets.

2.1.2 UAPProf

The UAPProf specification [6] is based on the CC/PP specification. Like CC/PP, a UAPProf profile is a two level hierarchy composed of components and attributes. Unlike CC/PP, the UAPProf specification also proposes a vocabulary - a specific set of components and attributes - to describe the next generation of WAP phones. The specification also describes two protocols for transmitting the profile from the client to the server. There is an open-source library called DELI developed at HP Labs that allows Java servlets to resolve HTTP requests containing delivery context information from CC/PP or UAPProf capable devices and query the resolved profile. Currently DELI only supports the W-HTTP protocol.

Profiles using the UAPProf vocabulary consist of six components: HardwarePlatform, SoftwarePlatform, NetworkCharacteristics, BrowserUA, WapCharacteristics and PushCharacteristics. These components contain attributes. In DELI each attribute has a distinct name and has an associated collection type, attribute type and resolution rule.

In UAPProf there are three collection types:

- i. *Simple* contains a single value e.g. ColorCapable in HardwarePlatform.
- ii. *Bag* contains multiple un-ordered values e.g. BluetoothProfile in the HardwarePlatform component.
- iii. *Seq* contains multiple ordered values e.g. Ccpp-AcceptLanguage in the SoftwarePlatform component.

In addition attributes can have one of four attribute types:

- *String* e.g. BrowserName in BrowserUA.
- *Boolean* e.g. ColorCapable in HardwarePlatform.
- *Number* is a positive integer e.g. BitsPerPixel in HardwarePlatform.
- *Dimension* is a pair of positive integers e.g. ScreenSize in HardwarePlatform.

Finally attributes are associated with a resolution rule:

- *Locked* indicates the final value of an attribute is the first occurrence of the attribute outside the default description block.
- *Override* indicates the final value of an attribute is the last occurrence of the attribute outside the default description block.
- *Append* indicates the final value of the attribute is the list of all occurrences of the attribute outside the default description block.

The profile is associated with the current network session or transaction. Each major component may have a collection of attributes or preferences. Examples of major components are the hardware platform upon which all the software is executing, the software platform upon which all the applications are hosted and each of the applications.

The UAPProf vocabulary is described using the file uaprofspec.xml. This describes the attribute name, component, collectionType, attributeType and resolution rule of each component. The vocabulary description file has the following format:

```
<?xml version="1.0"?>
<vocabspec>
  <attribute>
    <name>CcppAccept</name>
    <component>SoftwarePlatform</component>
    <collectionType>Bag</collectionType>
    <attributeType>Literal</attributeType>
    <resolution>Append</resolution>
  </attribute>
</vocabspec>
```

The following is a simplified example of the sort of data expected to be encoded in these profiles.

Hardware Platform

Memory = 64mb

CPU = PPC

Screen = 640*400*8

BlueTooth = Yes

Software Platform

OS version = 1.0

HTML version = 4.0

Sound = ON

Images = Yes

Email

Language = English

...

Some collections of properties and property values may be common to a particular component. For example: a specific model of a smart phone may come with a specific CPU, screen size and amount of memory by default. Gathering these "default" properties together as a distinct RDF resource makes it possible to independently retrieve and cache those properties. A collection of "default" properties is not mandatory, but it may improve network performance, especially the performance of relatively slow wireless networks.

Any RDF graph consists of nodes, arcs and leafs [1]. Nodes are resources, arcs are properties and leafs are property values. An RDF graph based on the previous example that includes "Default" properties for each major component is relatively straightforward as depicted below:

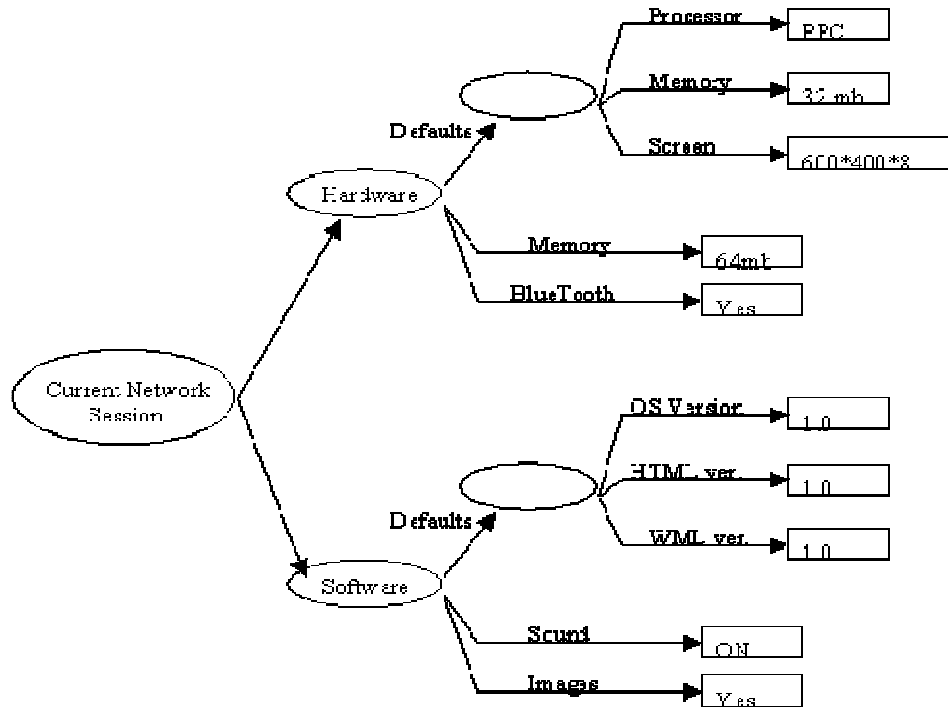


Figure 2: RDF graph referred to the example above.

The introduction of "Defaults" makes the graph of each major component more of a simple tree than a table. In this example the major components are associated with the current network session. In this case, the network session is serving as the root of a tree that includes the trees of each major component. RDF was originally intended to describe metadata associated with documents or other objects that can be named via a URL. The closest thing to a "document" associated with a CC/PP is the current network session.

From the point of view of any particular network transaction the only property or capability information that is important is whatever is "current". The network transaction does not care about the differences between defaults or persistent local changes; it only cares about the capabilities and preferences that apply to the current network transaction. Because this information may originate from multiple sources and because different parts of the capability profile may be differentially cached, the various components must be explicitly described in the network transaction.

The CC/PP is the encoding of profile information that needs to be shared between a client and a server, gateway or proxy. The persistent encoding of profile information and the encoding for the purposes of interoperability (communication) need not be the same. In this document we have considered the use of XML/RDF as the interoperability encoding. Persistent storage of profile information is left to the individual applications.

For the purpose of illustration consider below a more realistic example of inline encoding of a CC/PP for a hypothetical smart phone. This is an example of the type of information a phone might provide to a gateway/proxy/server. Note that we do not explicitly name the "current network session". Instead, the profiles of each major component are collected in a "Bag". This is probably not necessary since the document in question, the network session, is unlikely to contain additional RDF.

```

<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.w3.org/TR/WD-profile-vocabulary#">
  <rdf:Description about="HardwarePlatform">
    <prf:Defaults
      Vendor="Nokia"
      Model="2160"
      Type="PDA"
      ScreenSize="800x600x24"
      CPU="PPC"
      Keyboard="Yes"
      Memory="16mB"
      Bluetooth="YES"
      Speaker="Yes" />
    <prf:Modifications
      Memory="32mB" />
  </rdf:Description>
  <rdf:Description about="SoftwarePlatform">
    <prf:Defaults
      OS="EPOC1.0"
      HTMLVersion="4.0"
      JavaScriptVersion="4.0"
      WAPVersion="1.0"
      WMLScript="1.0" />
    <prf:Modifications
      Sound="Off"
      Images="Off" />
  </rdf:Description>
  <rdf:Description about="EpocEmail1.0">
    <prf:Defaults
      HTMLVersion="4.0" />
  </rdf:Description>
  <rdf:Description about="EpocCalendar1.0">
    <prf:Defaults
      HTMLVersion="4.0" />
  </rdf:Description>
  <rdf:Description about="UserPreferences">
    <prf:Defaults
      Language="English"/>
  </rdf:Description>
</rdf:RDF>

```

This sample profile is a collection of the capabilities and preferences associated with either a user or the hardware platform or a software component. Each collection of capabilities and preferences are organized within a description block. These description blocks may contain subordinate description blocks to describe default attributes or other collections of attributes.

2.1.3 DELI

As noted above, DELI [7] is an open-source library developed at HP Labs that allows Java servlets to resolve HTTP requests containing delivery context information from CC/PP or UAProf capable devices and supporting queries on the resolved profile.

Different web-enabled devices have different input, output, hardware, software, network and browser capabilities. In order for a web server or web-based application to provide optimised content to different clients it requires a description of the capabilities of the client known as the delivery context. Of the two recent standards that support device profiling for describing the delivery context, we have already described

the Composite Capabilities / Preferences Profile (CC/PP) created by the W3C, and the User Agent Profile (UAProf), an increasingly popular standard created by the WAP Forum.

DELI provides support for legacy devices so that the proprietary delivery context descriptions currently used by applications can be replaced by standardised CC/PP descriptions.

DELI can read vocabularies described using RDF schemas. The WAP Forum has published two such schemas to describe the two versions of UAProf currently in use. However the RDF Schema in its previous form did not provide an easy way of describing attributeType or resolution rules. Therefore the UAProf schemas store this information in the comments field for each attribute. Therefore DELI also parses the comments fields to create the vocabulary. This is not an ideal solution so hopefully a more robust way of representing this information will be used in later versions of UAProf.

2.1.3.1 DELI Structures and Profile Resolution Futures

Most mobile devices now use UAProf because this is capable of preserving the ordering of profile elements and resolution of how to treat the profiles as new profiles are added (device, personal and channel profiles).

As the personal use circumstances change (e.g. location etc) these profile elements have to be merged within the profile structure in a consistent way (for example those which have to update a field should update it and those which are simply to be added (appended) to append it and those that are time/location invariant to remain as they were (for example the display size) etc ...

Deli was produced to structure the three profile models as follows:

- *Device reference profiles* e.g. Nokia 6255 phone profiles,
- *In-line personal profiles* e.g. personal preferences/customisation of device
- *In-line channel profiles* e.g. Network BW and other constraints affecting traffic management, filtering, buffering etc.

This was so as to preserve the ordering and in this way deliver what is needed to deliver Profile Resolution. So all Deli's main code (a small 60K of JavaServlet) does is to take the profile elements and build its own structure. Then it could use this structure consistently to merge the profiles in an order preserving manner and with consistent resolution i.e. update those elements to be updated in a correct manner; i.e. either update or append or leave alone the fields respectively as it should for correct interpretation and usage of the profiles by the media transcoder and other server services provided to the client.

This Deli's task and its development was necessary before because all the profiling representation schemas have traditionally been RDF based and though RDF has a structure and a vocabulary, it is not structured in the sense of being order preserving and this structuring was what Deli had to do additionally at that time by building its own structures according to the above three categories.

Recently RDF Multiple Models concept has been introduced and it is now possible, whilst working from RDF, to distinguish the above three profiles as essentially belonging to three distinct models and thus having an easier Profile Resolution and order Preserving task to do by querying across the three models separately and for each using UAProf resolution as follows:

- If a profile field is "locked" then leave alone or use the first element
- If a profile field is "over-ride" then use the last element (updated one)
- If a profile field is "append" then merge together in that sequence e.g. English Italian will mean that the sever will first serve the English and then the Italian version of whatever it is (e.g. as in aircraft or airport announcements or songs etc.

2.1.4 WURFL

Another relevant source is WURFL (Wireless Universal Resource File). This is a free, open source project that provides an alternative source of information to UAProf. It provides a comprehensive resource of device information, and contains device information for 6.000 variants of devices. It is part of a FOSS (Free and Open Source Software) community effort focused on the problem of presenting content on the wide variety of wireless devices. Because WURFL is open source, anyone can contribute device information and corrections, not just device manufacturers. WURFL provides its own XML format for device characteristics description.

The main scope of the WURFL is to collect as much information as possible about all the existing mobile devices that access WAP pages so that developers are able to build better applications and better services for the users.

The Documentation and Developers guide is available on:

<http://wurfl.sourceforge.net/>.

2.2 Approach

Central to the core vision of GRAPPLE is the adaptation of accessible content to the user based on his profile and preferences. This document concerns a lateral work of adaptation based on the features of the device the user uses to access to the courses.

Content provided to the learner must take into account the user's needs and preferences, the capabilities of the devices used by the user and the presentation modalities of the content.

The basic workflow of the content personalisation process is illustrated in Figure 3.

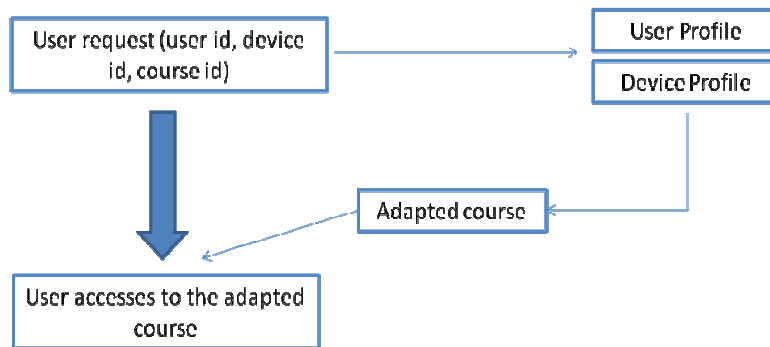


Figure 3: Adaptation process.

The user requests to access an adaptive course from his device that can be a computer, a PDA or a mobile. The course has to be adapted in base of the user and device profile.

The adaptation is performed in a central core engine, GRAPPLE Adaptation Learning Environment (GALE) where the courses are adapted based on the user and device profiles. The device profile adaptation is performed in a module, GRAPPLE Device Adaptation (GDA), located internally into GALE. This is described in detail later in section 2.3.

The adaptation based on the user profile is the main objective of the GRAPPLE project and it is not dealt with in this deliverable.

The GRAPPLE Device Adaptation component uses mainly XSL in order to perform the requested device adaptation.

2.2.1 Extensible Style sheet Language (XSL)

XSL [8] is a language for expressing style sheets. An XSL style sheet is, like with CSS, a file that describes how to display an XML document of a given type. XSL shares the functionality and is compatible with CSS2 (although it uses a different syntax). It also adds:

- A transformation language for XML documents: XSLT [9]. Originally intended to perform complex styling operations, like the generation of tables of contents and indexes, it is now used as a general purpose XML processing language. With XSLT, it is possible to take an XML document and choose the elements and values, then generate a new file with the selected choices. Because of XSLT's ability to change the content of an XML document, XSLT is referred to as the stylesheet for XML. XSLT is thus widely used for purposes other than XSL, like generating HTML web pages from XML data.
- An expression language (XPath) [10] used by XSLT to access or refer to parts of an XML document
- Advanced styling features, expressed by an XML document type, which defines a set of elements called Formatting Objects (XSL-FO), and attributes (an XML vocabulary for specifying formatting semantics).

Given a class of arbitrarily structured XML documents or data files, designers use an XSL stylesheet to express their intentions about how that structured content should be presented; that is, how the source content should be styled, laid out, and paginated onto some presentation medium, such as a window in a Web browser or a hand-held device, or a set of physical pages in a catalog, report, pamphlet, or book.

An XSL **stylesheet processor** accepts a document or data in XML and an XSL stylesheet and produces the presentation of that XML source content that was intended by the designer of that stylesheet. There are two aspects of this presentation process: first, constructing a result tree from the XML source tree and second, interpreting the result tree to produce formatted results suitable for presentation on a display, on paper, in speech, or onto other media. The first aspect is called **tree transformation** and the second is called **formatting**. The process of formatting is performed by the **formatter**. This formatter may simply be a rendering engine inside a browser.

2.2.1.1 Tree transformation

Tree transformation allows the structure of the result tree to be significantly different from the structure of the source tree. For example, one could add a table-of-contents as a filtered selection of an original source document, or one could rearrange source data into a sorted tabular presentation. In constructing the result tree, the tree transformation process also adds the information necessary to format that result tree.

Formatting is enabled by including formatting semantics in the result tree. Formatting semantics are expressed in terms of a catalog of classes of **formatting objects**. The nodes of the result tree are formatting objects. The classes of formatting objects denote typographic abstractions such as page, paragraph, table, and so forth. Finer control over the presentation of these abstractions is provided by a set of formatting properties, such as those controlling indents, word- and letter spacing, and widow, orphan, and hyphenation control. In XSL, the classes of formatting objects and **formatting properties** provide the vocabulary for expressing presentation intent.

The XSL processing model is intended to be conceptual only. An implementation is not mandated to provide these as separate processes. Furthermore, implementations are free to process the source document in any way that produces the same result as if it were processed using the conceptual XSL processing model. A diagram depicting the detailed conceptual model is shown below:

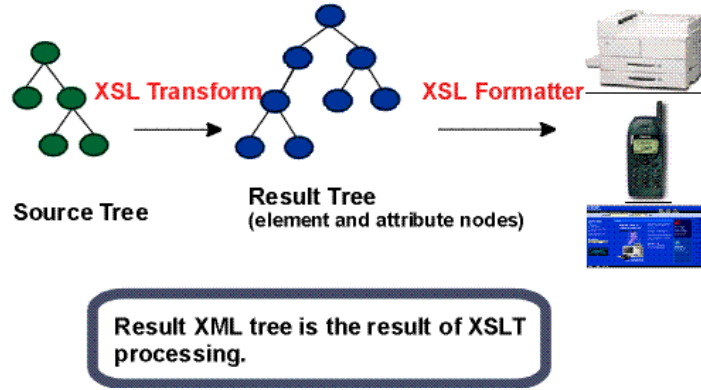


Figure 4: XSL Two Processes: Transformation & Formatting.

Tree transformation constructs the result tree. In XSL, this tree is called the **element and attribute tree**, with objects primarily in the "formatting object" namespace. In this tree, a formatting object is represented as an XML element, with the properties represented by a set of XML attribute-value pairs. The content of the formatting object is the content of the XML element. Tree transformation is defined in the XSLT Recommendation [9]. A diagram depicting this conceptual process is shown below.

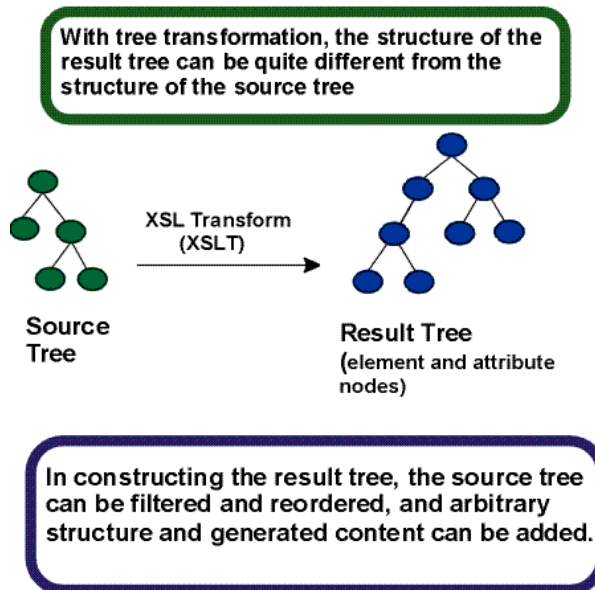


Figure 5: Transform to Another Vocabulary.

The XSL stylesheet is used in tree transformation. A stylesheet contains a set of tree construction rules. The tree construction rules have two parts: a pattern that is matched against elements in the source tree and a template that constructs a portion of the result tree. This allows a stylesheet to be applicable to a wide class of documents that have similar source tree structures.

2.2.1.2 Formatting

Formatting interprets the result tree in its formatting object tree form to produce the presentation intended by the designer of the stylesheet from which the XML element and attribute tree in the "fo" namespace was constructed.

The vocabulary of formatting objects supported by XSL - the set of fo: element types - represents the set of typographic abstractions available to the designer. Semantically, each formatting object represents a specification for a part of the pagination, layout, and styling information that will be applied to the content of that formatting object as a result of formatting the whole result tree. Each formatting object class represents a particular kind of formatting behaviour. For example, the block formatting object class represents the breaking of the content of a paragraph into lines. Other parts of the specification may come from other formatting objects; for example, the formatting of a paragraph (block formatting object) depends on both the specification of properties on the block formatting object and the specification of the layout structure into which the block is placed by the formatter.

The properties associated with an instance of a formatting object control the formatting of that object. Some of the properties, for example "colour", directly specify the formatted result. Other properties, for example 'space-before', only constrain the set of possible formatted results without specifying any particular formatted result. The formatter may make choices among other possible considerations such as aesthetics.

Formatting consists of the generation of a tree of geometric areas, called the **area tree**. The geometric areas are positioned on a sequence of one or more pages (a browser typically uses a single page). Each geometric area has a position on the page, a specification of what to display in that area and may have a background, padding, and borders. For example, formatting a single character generates an area sufficiently large enough to hold the glyph that is used to present the character visually and the glyph is what is displayed in this area. These areas may be nested. For example, the glyph may be positioned within a line, within a block, within a page.

Rendering takes the area tree, the abstract model of the presentation (in terms of pages and their collections of areas), and causes a presentation to appear on the relevant medium, such as a browser window on a computer display screen or sheets of paper. The semantics of rendering are not described in detail in this specification.

The first step in formatting is to "objectify" the element and attribute tree obtained via an XSLT transformation. Objectifying the tree basically consists of turning the elements in the tree into formatting object nodes and the attributes into property specifications. The result of this step is the **formatting object tree**.

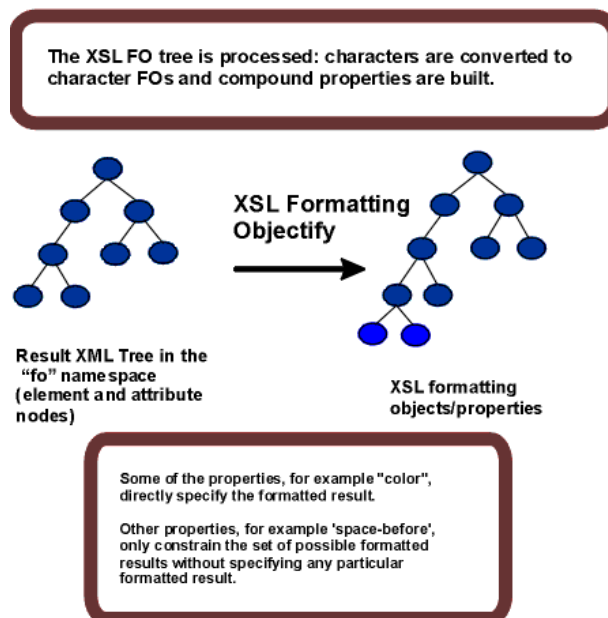


Figure 6: Build the XSL Formatting Object Tree.

As part of the step of objectifying, the characters that occur in the result tree are replaced by *fo:character* nodes. Characters in text nodes which consist solely of white space characters and which are children of elements whose corresponding formatting objects do not permit *fo:character* nodes as children are ignored. Other characters within elements whose corresponding formatting objects do not permit *fo:character* nodes as children are errors.

The content of the *fo:instream-foreign-object* is not objectified; instead the object representing the *fo:instream-foreign-object* element points to the appropriate node in the element and attribute tree. Similarly any non-XSL namespace child element of *fo:declarations* is not objectified; instead the object representing the *fo:declarations* element points to the appropriate node in the element and attribute tree.

The second phase in formatting is to refine the formatting object tree to produce the **refined formatting object tree**. The refinement process handles the mapping from properties to traits. This consists of: (1) shorthand expansion into individual properties, (2) mapping of corresponding properties, (3) determining computed values (may include expression evaluation), (4) handling white-space-treatment and linefeed-treatment property effects, and (5) inheritance.

The refinement step is depicted in the diagram below.

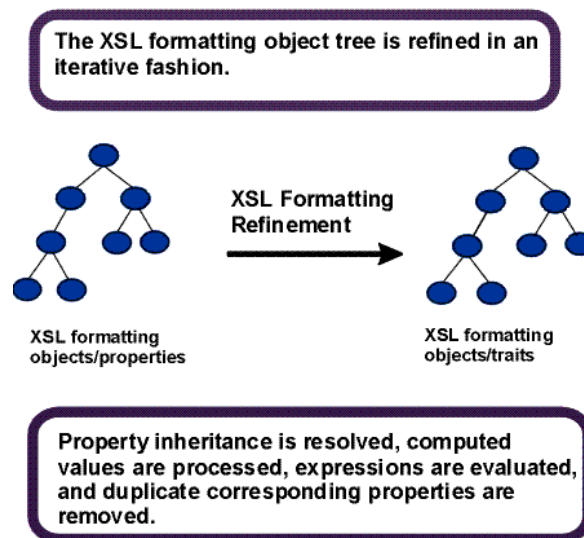
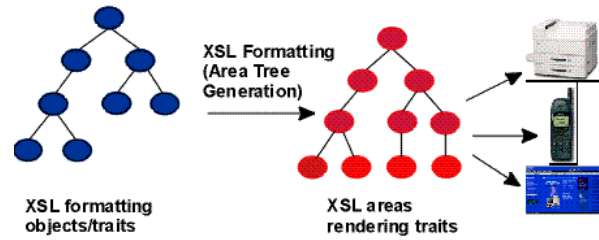


Figure 7: Refine the Formatting Object Tree.

The third step in formatting is the construction of the area tree. The area tree is generated as described in the semantics of each formatting object. The traits applicable to each formatting object class control how the areas are generated. Although every formatting property may be specified on every formatting object, for each formatting object class, only a subset of the formatting properties are used to determine the traits for objects of that class.

Area generation is depicted in the diagram below.

The last part of formatting describes the generation of a tree of geometric areas. These areas are positioned on a sequence of one or more pages.



Each geometric area has a position on the page, a specification of what to display in that area and may have a background, padding, and borders.

Figure 8: Generate the Area Tree.

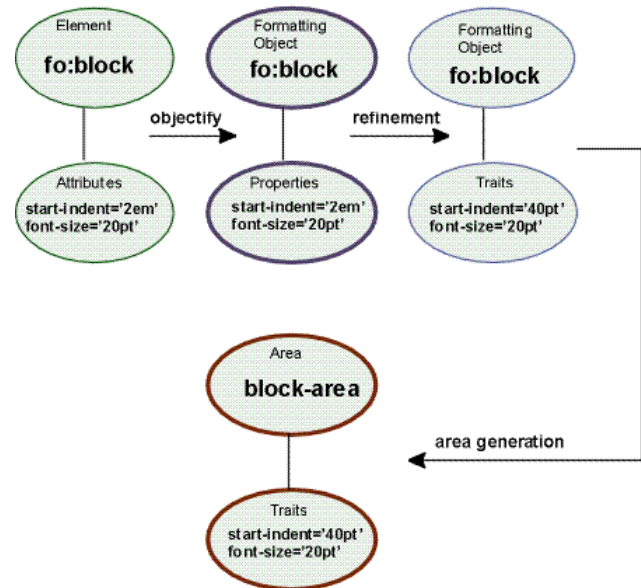


Figure 9: Summary of the Process.

2.2.2 Device and network features

In the GRAPPLE project, the user can access to an adaptive course provided by an LMS, through three different types of device:

- PC
- Personal Digital Assistant (PDA)
- Mobile.

A content that has been designed for fruition on a PC when presented onto a PDA or a Mobile may undergo a set of changes. These changes are decided and performed in the GDA. The first, and probably the more apparent is that content is highly probable to be scaled down to a smaller dimension, as even the simplest of all apparent changes occurring (that is converting from landscape to portrait or vice versa) implies a change in:

- areas sizes (to retain the same content) and
- location of some specific content.

Unless the PDA/Mobile is able to automatically flip the screen (that is presenting data in landscape rather than portrait) such a transformation has to be done prior to delivery. Now this is not the only point as passing from PCs to PDA/Mobile, the change in display size implies:

- Lower display area
- Lower computational power
- Lower image quality
- Smaller fonts
- Worse anti-aliasing
- Scroll-bars / page tabs
- Different content distribution.

This latter point may sound strange, but given the fact that the most diffused browser for Windows CE /Windows Mobile equipped PDA or Mobiles does not support CSS files and formatted HTML page will lose most of what was foreseen at content design and production time.

This, in some contexts, where content quality or publisher / distributor branding is highly related to content quality may be a major limiting factor that may lead to the voluntary production of device customized content production. Moreover at rendering time (either at server or client side) of a content has to be taken into account that passing from a single page to a set of pages, links between portions of the same page (anchors) are turned into links between pages (URLs), including the return to “top of page” as the way content is browsed changes from:

- Scrolling to
- Paging.

This device profile captures information related to user’s devices, that the service provider must take into account in delivering the contents. Generally the minimal set required comprises those that affect the proper utilization of the contents on the device e.g. screen resolution for images (the resolution most used for PDA/mobile is 240x320 pixels). This profile type comprises the most vital information for the content providers to transcode the content to match the content to device characteristics to enhance the usage of the content.

The network profile captures bandwidth utilization and QoS related information for the carrier to use to optimize delivery of the content to the user’s device. It is necessary for the distributors to know various characteristics of the network so as to attempt to provide the promised QoS. The information contained in this profile can also be used for transcoding of the contents to better utilize both network and device capabilities.

2.3 Device and network adaptation infrastructure

As mentioned above, the adaptation is performed in a central core engine, GRAPPLE Adaptation Learning Environment (GALE) where the courses are adapted based on the user and device profiles. The device profile adaptation is performed in a module, GRAPPLE Device Adaptation (GDA), located internally into GALE. Figure 10 shows how the adaptation process is located in the GRAPPLE system: only a few components are present here, the Learning Management System (LMS), which is the entry point for the learner to the GRAPPLE system, GALE, which is the adaptation core engine that performs the proper adaptation by using the user profile provided by the GRAPPLE User Model Framework (GUMF) and the device profile through GDA.

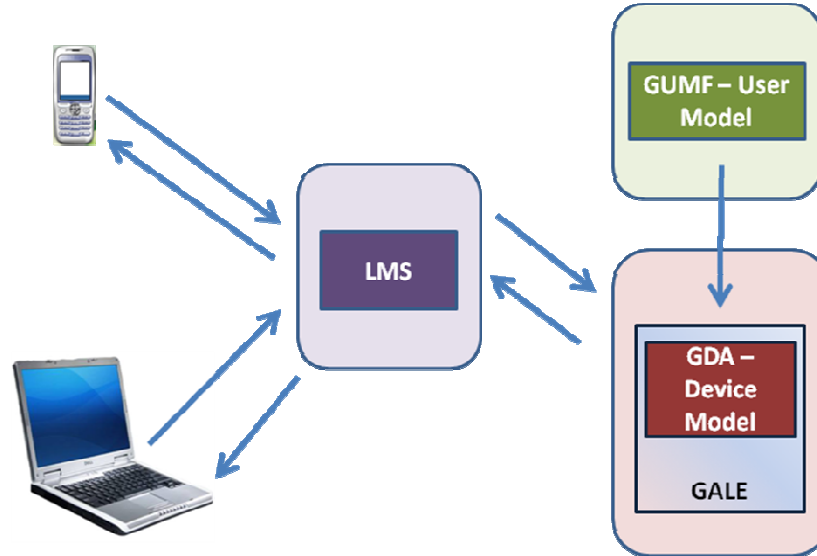


Figure 10: GRAPPLE adaptation process.

The LMS is only an intermediate between the mobile device and GALE that is in charge of delivering the adaptive courses. The LMS is not responsible for providing the adapted contents to the mobile devices.

As described above, GDA is an internal component of GALE and it is in charge to analyse the HTTP request [11] messages that carry device profile information.

As shown in the figure below, when the learner requests to access an adaptive course, GALE passes the HTTP request message to GDA and, once GALE has adapted the course in function of the user profile, GALE asks GDA to adapt the course in function of the learner's device profile.

The device model adaptation is composed by an XSL Processing Engine able to apply the XSL file appropriate to the learner's device, and a series of modules in charge of resizing image and text recap, as shown below.

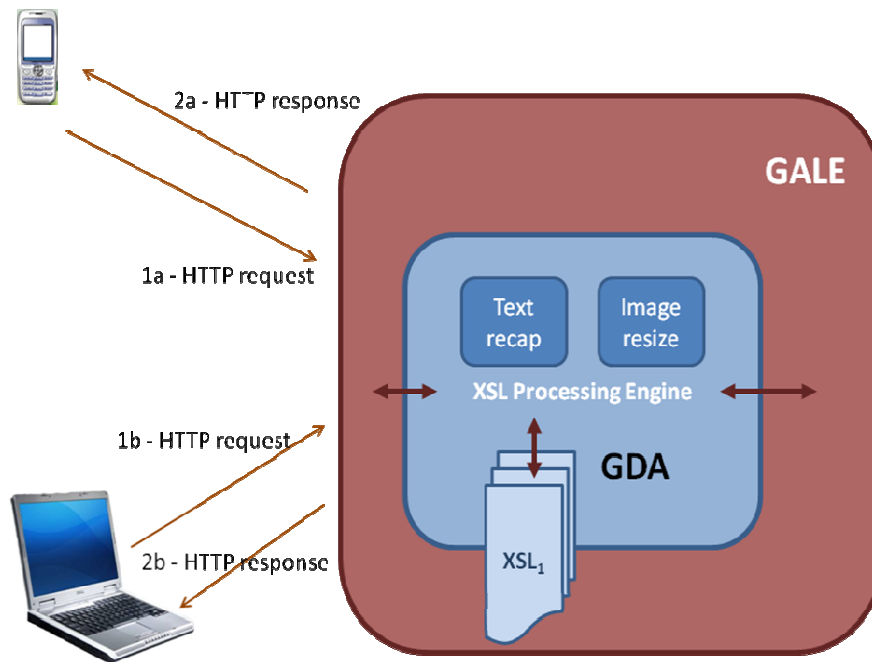


Figure 11: GRAPPLE adaptation process details.

The related development activity is organized in three phases:

1. Definition of the XSL files appropriate to the single device type (PC, PDA, mobile), as described in 2.2.1.
2. Implementation of the XSL Processing Engine able to perform the course adaptation based on the device model. It is the core engine of the GDA module: it is able to read the User Agent Profile information and adapt the course to the device features.
3. Implementations of the single modules in charge of particular tasks, such as image resizing, text recap.

The work requires a strong collaboration with the WP1 group in charge of the GALE development. The GDA module is planned to be available for M20.

3 References

1. Hjelm, J., Nilsson, M.: Towards RDF-based profiles for context-based position-dependent services, <http://www.w3.org/Mobile/posdep/ericsson-position-paper.htm>
2. Manola F., Miller E.: Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation February 10, 2004, <http://www.w3.org/TR/REC-rdf-syntax/>
3. Brickley D., Guha R.V.: Resource Description Framework (RDF) Schema Specification, February 10, 2004, <http://www.w3.org/TR/PR-rdf-schema>
4. Reynolds F., Hjelm J., Dawkins S., Singhal S.: Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation, July 27, 1999, <http://www.w3.org/TR/NOTE-CCPP/>
5. Klyne G., Reynolds F., Woodrow C., Ohto H., others: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0, January 15, 2004, <http://www.w3.org/TR/CCPP-struct-vocab/#35#35>
6. WAP Forum: WAG UAPProf, October 20, 2001, <http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf>
7. Butler M.H.: DELI: A DELivery context LIBrary for CC/PP and UAPProf, September 25, 2001, <http://www.hpl.hp.com/techreports/2001/HPL-2001-260.pdf>
8. Berglund A.: Extensible Stylesheet Language (XSL) Version 1.1, December 05, 2006, <http://www.w3.org/TR/xsl11/>
9. Clark J.: XSL Transformations (XSLT), November 16, 1999, <http://www.w3.org/TR/xslt#data-model>
10. Clark J., DeRose S.: XML Path Language (XPath), November 16, 1999, <http://www.w3.org/TR/xpath>
11. Fielding R, Irvine U.C., Gettys J., others: Hypertext Transfer Protocol -- HTTP/1.1, September 1, 2004, <http://www.w3.org/Protocols/rfc2616/rfc2616.html>
12. Tran L., Butler M.: JSR 188: CC/PP Processing, October 30, 2003, <http://jcp.org/en/jsr/detail?id=188>.

Appendix

CC/PP Profile for Device "Sony Ericsson T39"

```

<?xml version="1.0" ?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdf="http://www.w3.org/1999/02/22-
rdf-syntax-ns#" xmlns:prf="http://www.wapforum.org/UAPROF/ccppschemata-20000405#">
<rdf:Description ID="Profile">
<prf:component>
    <rdf:Description ID="HardwarePlatform">
    <prf:ScreenSize>101x54</prf:ScreenSize>
    <prf:Model>T39m</prf:Model>
<prf:InputCharSet>
    <rdf:Bag>
        <rdf:li>ISO-8859-1</rdf:li>
        <rdf:li>US-ASCII</rdf:li>
        <rdf:li>UTF-8</rdf:li>
        <rdf:li>ISO-10646-UCS-2</rdf:li>
    </rdf:Bag>
</prf:InputCharSet>
<prf:ScreenSizeChar>15x5</prf:ScreenSizeChar>
<prf:BitsPerPixel>2</prf:BitsPerPixel>
<prf:ColorCapable>No</prf:ColorCapable>
<prf:TextInputCapable>Yes</prf:TextInputCapable>
<prf:ImageCapable>Yes</prf:ImageCapable>
<prf:Keyboard>PhoneKeypad</prf:Keyboard>
<prf:NumberOfSoftKeys>0</prf:NumberOfSoftKeys>
<prf:Vendor>Ericsson Mobile Communications AB</prf:Vendor>
<prf:OutputCharSet>
    <rdf:Bag>
        <rdf:li>ISO-8859-1</rdf:li>
        <rdf:li>US-ASCII</rdf:li>
        <rdf:li>UTF-8</rdf:li>
        <rdf:li>ISO-10646-UCS-2</rdf:li>
    </rdf:Bag>
</prf:OutputCharSet>
<prf:SoundOutputCapable>Yes</prf:SoundOutputCapable>
<prf:StandardFontProportional>Yes</prf:StandardFontProportional>
<prf:PixelsAspectRatio>1x0,9</prf:PixelsAspectRatio>
</rdf:Description>
</prf:component>

```

```

=<prf:component>
  <rdf:Description ID="SoftwarePlatform">
    <prf:AcceptDownloadableSoftware>No</prf:AcceptDownloadableSoftware>
  </rdf:Description>
</prf:component>
<prf:component>
  <rdf:Description ID="NetworkCharacteristics">
    <prf:SecuritySupport>WTLS class 1/2/3/signText</prf:SecuritySupport>
    <prf:SupportedBearers>
      <rdf:Bag>
        <rdf:li>TwoWaySMS</rdf:li>
        <rdf:li>CSD</rdf:li>
        <rdf:li>GPRS</rdf:li>
      </rdf:Bag>
    </prf:SupportedBearers>
  </rdf:Description>
</prf:component>
<prf:component>
  <rdf:Description ID="BrowserUA">
    <prf:BrowserName>Ericsson</prf:BrowserName>
    <prf:CcppAccept>
      <rdf:Bag>
        <rdf:li>application/vnd.wap.wmlc</rdf:li>
        <rdf:li>application/vnd.wap.wbxml</rdf:li>
        <rdf:li>application/vnd.wap.wmlscriptc</rdf:li>
        <rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
        <rdf:li>text/x-vCard</rdf:li>
        <rdf:li>text/x-vCalendar</rdf:li>
        <rdf:li>text/x-vMel</rdf:li>
        <rdf:li>text/x-eMelody</rdf:li>
        <rdf:li>image/vnd.wap.wbmp</rdf:li>
        <rdf:li>image/gif</rdf:li>
        <rdf:li>application/vnd.wap.wtls-ca-certificate</rdf:li>
        <rdf:li>application/vnd.wap.sic</rdf:li>
        <rdf:li>application/vnd.wap.slc</rdf:li>
        <rdf:li>application/vnd.wap.coc</rdf:li>
        <rdf:li>application/vnd.wap.sia</rdf:li>
      </rdf:Bag>
    </prf:CcppAccept>
    <prf:CcppAccept-Charset>
      <rdf:Bag>

```

```

        <rdf:li>US-ASCII</rdf:li>
        <rdf:li>ISO-8859-1</rdf:li>
        <rdf:li>UTF-8</rdf:li>
        <rdf:li>ISO-10646-UCS-2</rdf:li>
    </rdf:Bag>
</prf:CcppAccept-Charset>
<prf:CcppAccept-Encoding>
    <rdf:Bag>
        <rdf:li>base64</rdf:li>
    </rdf:Bag>
</prf:CcppAccept-Encoding>
<prf:FramesCapable>No</prf:FramesCapable>
<prf:TablesCapable>Yes</prf:TablesCapable>
</rdf:Description>
</prf:component>
<prf:component>
    <rdf:Description ID="WapCharacteristics">
        <prf:WapDeviceClass>C</prf:WapDeviceClass>
        <prf:WapPushMsgSize>3000</prf:WapPushMsgSize>
        <prf:WapVersion>1.2.1/June 2000</prf:WapVersion>
        <prf:WmlVersion>
            <rdf:Bag>
                <rdf:li>1.2.1/June 2000</rdf:li>
                <rdf:li>1.1</rdf:li>
            </rdf:Bag>
        </prf:WmlVersion>
        <prf:WmlDeckSize>3000</prf:WmlDeckSize>
        <prf:WmlScriptVersion>
            <rdf:Bag>
                <rdf:li>1.2.1/June 2000</rdf:li>
                <rdf:li>1.1</rdf:li>
            </rdf:Bag>
        </prf:WmlScriptVersion>
        <prf:WmlScriptLibraries>
            <rdf:Bag>
                <rdf:li>Lang</rdf:li>
                <rdf:li>Float</rdf:li>
                <rdf:li>String</rdf:li>
                <rdf:li>URL</rdf:li>
                <rdf:li>WMLBrowser</rdf:li>
                <rdf:li>Dialogs</rdf:li>
            </rdf:Bag>
        </prf:WmlScriptLibraries>
    </rdf:Description>
</prf:component>

```

```
</rdf:Bag>
</prf:WmlScriptLibraries>
<prf:WtaiLibraries>
  <rdf:Bag>
    <rdf:li>WTA.Public.makeCall</rdf:li>
    <rdf:li>WTA.Public.sendDTMF</rdf:li>
    <rdf:li>WTA.Public.addPBEntry</rdf:li>
  </rdf:Bag>
</prf:WtaiLibraries>
</rdf:Description>
</prf:component>
</rdf:Description>
</RDF>
```