

# WebProfiles: A Negotiation Model for User Awareness in Personal Area Networks

Juan Ignacio Vazquez  
Faculty of Engineering, Deusto University,  
Bilbao, Spain  
ivazquez@eside.deusto.es

Diego López de Ipiña  
Faculty of Engineering, Deusto University,  
Bilbao, Spain  
dipina@eside.deusto.es

## Abstract

*Automatic adaptation of the environment to user preferences results in personalized ambients that fulfill user interaction requirements in a more suitable way. However, the problem of environment adaptation has been traditionally solved via ad-hoc context-specific solutions. In this paper we present the WebProfiles model, a negotiation mechanism for Ubiquitous Computing, based on Web technology extensions, that enables user preferences awareness capability in surrounding devices and services, so that the environment meets the user and knows him as if specifically designed for that purpose.*

## 1. Introduction

Context-awareness mechanisms and, in particular, user-related information awareness, are one of those required extensions for the Ubiquitous Computing to fulfill present and future services demands. Context-awareness would allow a service to perceive user-related and task-related information that can be used to provide a more suitable and effective outcome for that user. Context information can be provided by the user-agent explicitly (user-related data), or can be extracted by the service from other available sources in a scenario dependent paradigm.

On the other hand, the Web model, including HTTP technology, has proven to be suitable to support communication needs in Ubiquitous Computing scenarios [1,2], so that the same communication model can be applied to provide both global scale and local scale services.

HTTP context awareness is a broad concept than can embrace the traditional HTTP state management mechanism [3], which has been very criticized over the years, despite the Web would not be as powerful as it is without those small chunks of information called cookies [4]. These pieces of data allow a web service

to recognize immediately a visiting user and parameterize the nature of the information being presented based on past visits and interaction, and it can be considered a very simple form of user-awareness mechanism.

In order to materialize new capabilities we have created the WebProfiles model: an HTTP extension that supports context information management as well as a negotiation process that allows clients and service providers to establish the appropriate informational environment for the service execution in Ambient Intelligence or Ubiquitous Computing scenarios. The goal is to extend Personal Area Networks to create what we call *Personal Area Webs*, applying the successful web communication model around the user physical circumstances in order to weave context-aware user-device relationships.

In our vision, Personal Area Webs exist around the user and move with him, linking that user with the surrounding devices, and creating a services-rich digital ecosystem extended with user awareness capabilities.

Some other initiatives in the global web model could be applied also for this personal web model, such as the Open Profiling System (OPS) [6], but our proposal stresses the use of user-related context in the form of preferences about service characteristics.

The Platform for Privacy Preferences (P3P) initiative [7] also included in the initial specifications the idea of some form of information exchange to support web service adaptation, but it was finally considered out of the scope of the standardization, as well as some criticisms arose about privacy concerns [8].

Special attention must be paid to WS-Context [9], an ongoing work to define a mechanism for context information sharing among multiple coordinated services for executing a task. This specification is tightly linked to the Web Services technologies such as SOAP [10], WSDL [11] and more concretely to WS-

CAF (Composite Application Framework), WS-Coordination and WS-Transactions.

The WebProfiles model introduced in this work shares many similarities with these other technologies and inherits some of their characteristics, but we stress the use of user-related context in the form of preferences. While CC/PP [18] seems to be a good initial alternative, it is too oriented to express device information and concrete data instead of conditional preferences as explained below.

In section 2, we introduce the concept of context awareness and its implications for personal area web-based services. In section 3, we present WPML (WebProfiles Markup Language) to represent user preferences, which can be considered one form of context data. In section 4 we introduce the basics of the WebProfiles negotiation model. In section 5, the model is fully explored through HTTP extensions analysis, basically new headers and intensive usage of HTTP multipart messages. This section covers all the practical aspects of WebProfiles implementation showing how it is a full working model. Finally in section 6, we present some open issues about the evolution of the WebProfiles model and personal area user-awareness in general.

## 2. User-context awareness in the Personal Area Web

It is not easy to find a widely accepted definition for “context”, since it depends on the framework in which it is applied. One of the most precise and open statements we can mention is found in the WS-Context specification [9] and declares that a “context contains information about the execution environment of an activity”.

That is, a context is an information entity that can be used to provide additional data for some process execution. Probably, that execution could be performed without that supplementary information, but surely its influence can be used to establish a user-adapted execution framework more precisely.

Probably, and important part of the context information for a service is related to the user, expressing data about him, his preferences maybe depending on other context information, and so on. We can define user context information as the subset of the context information influencing a service that model user-related aspects.

When coping with personal area web services and web processes, it is often necessary to exchange a large amount of data to execute a service. The service provider needs to be supplied with all the data the user keeps that are relevant to the situation. For example, if

a user wants to configure his temperature preferences every time he enters a new room in a building, he must repeat similar interactions over the temperature control devices once and again at every location.

Web-based surrounding devices populated with embedded web services are not aware of users’ context, provoking unnecessary interactions refinements over the time that end up in entering the same data manually along different devices repeatedly.

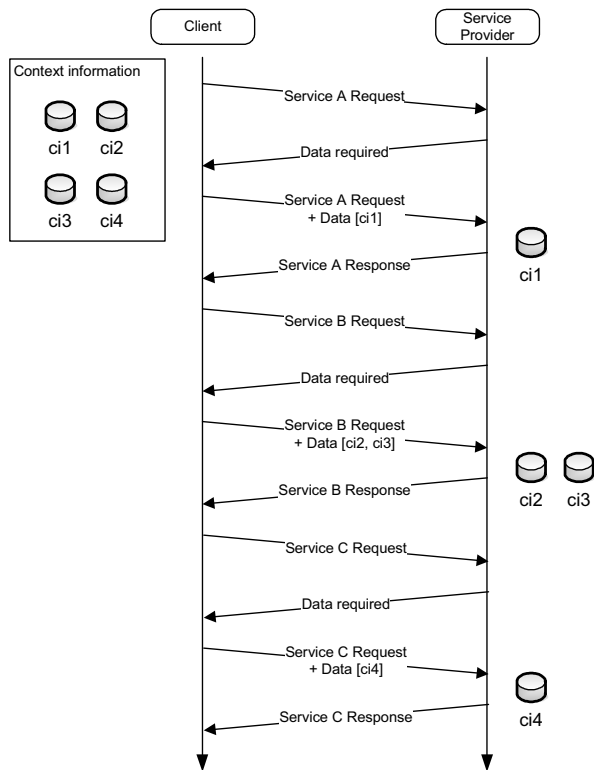
HTTP state management mechanism has provided a simple method for a service provider to recognize the user in subsequent visits via cookies. Nevertheless, cookies are used primary for client identification, not for context information representation due to format limitations and security considerations.

Our goal was to find a mechanism as simple as cookies but able to cope with user context information sharing between embedded clients and servers, where user preferences could be formally defined and structured so that they could be passed forward to validated devices in order to obtain a more personalized service execution.

That is, prior to actual service interaction between the user and the service provider (a web-enabled device), the user-agent and the embedded server negotiate and set up an information-rich context in such a way that it seems that the service provider knows the user beforehand, despite the latter has never interacted with the device before. Further interactions can be accomplished inside that mutual knowledge framework.

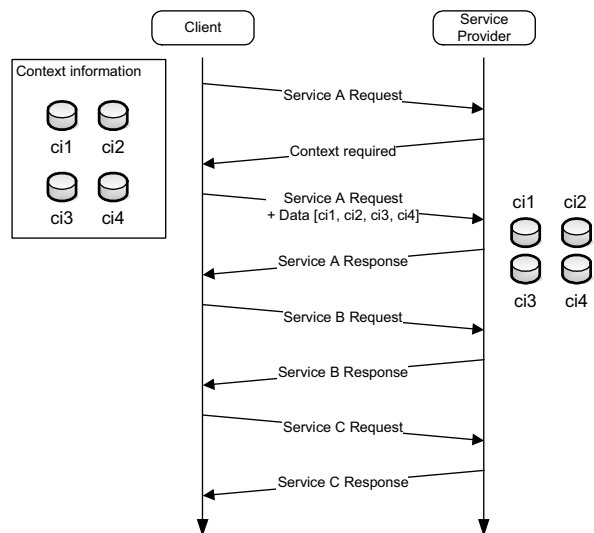
Figure 1 illustrates the interaction process between a client, also called user-agent (some kind of process running within the user’s mobile phone or PDA), and a service provider (a surrounding device) in the usual way, without previous context negotiation.

Data are supplied by the client as needed, increasing the number of interactions. This diagram is familiar in the traditional Web paradigm, since several extensions implement similar mechanisms, such as HTTP Authentication, where the client supplies authentication data under demand in a client-driven negotiation.



**Figure 1. Service interaction without previous context negotiation**

Figure 2 illustrates the same services requests with a previous context negotiation process.



**Figure 2. Service interaction with previous context negotiation**

As we can see, context is established in the initial phases of the communication process. The service provider obtains immediately a perfect knowledge

about required user information, which can be applied to carry out a personalized service execution. Moreover, the number of interactions decreases dramatically, resulting in saved time and communication efforts.

Of course, these advantages depend significantly in how accurately the user context and preferences information can be identified beforehand. Imprecise negotiation can result in a large amount of unusable exchanged data along with a lack of relevant information that forces extra interactions. How the WebProfiles model identifies, represents and negotiates the user context set up is analyzed in the next section.

### 3. The WebProfiles Markup Language (WPML)

A service or a system can be probably represented at any time via state information, which evolves along the state space that represents all the possible situations under which the service can be found.

After all, expressing and transmitting user preferences is a way of influencing the state of the service or system when interacting with the user [5] to meet his desires or requirements.

But the reality is a bit more complex. Probably the user wants his preferences to be applied in a context-sensitive way, that is, depending on the service actual state or information, the preferences can vary.

Here, we redefine the concept and define *context* as the set of conditions that must be tested and probably fulfilled by the service to activate the user preferences. Thus, the context represents the surrounding information that must be checked to determine the need for setting up some concrete preferences.

On the other hand, we define *configuration* as the set of related preferences that express user requirements or predilections for some features of the service operation.

Finally, we define *profile* as the association of a context to a configuration, that is, the set of conditions under which some preferences must be activated. In fact, an accepted configuration provokes a change in the service state related to the user, creating a *new context* closer to the user's desires, so the whole process can be called *context negotiation* and it is described at a higher level in section 4.

Via context negotiation the user (or user-agent) expresses and transmits profiles that must be processed by the target service, influencing its behaviour and state, thus achieving user-aware web services.

For example, a user preference can represent "I want the temperature of my present location to be

between 20°C and 30°C when outside because out of this range I use to get ill, so this is mandatory, and between 25°C and 30°C when at home at night, mandatory too.” In this case “temperature of my present location to be between 20°C and 30°C” is the preference to be activated in a context “when outside” and “between 25°C and 30°C” is a preference to be activated in the context “when at home at night”.

Both contexts and configurations are expressed with two complementary mechanisms. First, data structures of XML Data Schemas are used to identify the concepts about which conditions and preferences are going to be expressed. Second, we have developed an XML-based language called WPML (WebProfiles Markup Language) to relate configurations to contexts

in which those preferences must be activated, that is, to represent profiles.

In order to express both the context information and the preferences we need to use XML Data Schemas that structure the involved domain of knowledge, maybe the “location” domain, the “time” domain, and the “ambient conditions” domain, which includes the temperature, in the above example. Depending on some characteristics in the location and time domain we want some preferences in the ambient/temperature domain. Since every domain is identified via a unique namespace, no ambiguities must arise when generating our profile.

The above example can be represented in WPML in the following way:

```
<?xml version="1.0" encoding="UTF-8"?>
<wpm1 xmlns="http://www.webprofiles.org/schemas/wpm110"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.webprofiles.org/schemas/wpm110
  http://www.webprofiles.org/schemas/wpm110.xsd" querylang="xpath">
  <profile uri="urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6">
    <context xmlns:loc="http://www.webprofiles.org/dataschemas/location">
      <pattern ID="pat1" use="required" match="/loc:location[@loc:type!='Home']"/>
    </context>
    <configuration xmlns:amb="http://www.webprofiles.org/dataschemas/ambientconditions">
      <preference ID="pre1" use="required" about="/amb:ambient/amb:temperature/text()"
        operator="gt" value="20"/>
      <preference ID="pre2" use="required" about="/amb:ambient/amb:temperature/text()"
        operator="lt" value="30"/>
    </configuration>
  </profile>
</wpm1>
```

The <profile> element contains an attribute uri, with a unique universal identifier for referencing this profile and two elements: <context> and <configuration>. The <context> expresses a set of patterns (the technical word we use for conditions) in domains to activate preferences. Those patterns are expressed using XPath and are considered to be fulfilled if the XPath expression yields an object when evaluated. The <configuration> element contains the user preferences, addressing them also via XPath but expressing ranges via the operator and value attributes.

This is a remarkable difference with other systems like CC/PP [18], which merely conveys user-agent information using the classical attribute-value method. In the WebProfiles model, we can express *ranges* of values that are preferred by the user for a concrete attribute, thus allowing more expressive power about real preferences. We can even represent our desire for a concrete attribute *not to be* of a certain value or

range, using the MathML-based operators eq, neq, gt, lt, geq and leq.

When the XPath expression in a preference yields more than one object, a node-set, only the first one is selected.

Of course, we could express our temperature preferences without any condition related to the location. In those cases where preferences are not attached to existing context conditions, the context section can be omitted, so that only the configuration information is conveyed. We name this type of profiles, *context-less profiles*.

The “required” value at the use attribute in the pattern element indicates that the condition must be present and fulfilled, considering it as failed if not present or nor checkable. An “optional” value there indicates that the condition must only be fulfilled if present, which is not mandatory

Several patterns must be provided in the same or different domains. For example,

```

<context
  xmlns:loc="http://www.webprofiles.org/dataschemas/location"
  xmlns:time="http://www.webprofiles.org/dataschemas/time">
  <pattern ID="pat1" use="required"
    match="/loc:location[@loc:type='Home']"/>
  <pattern ID="pat2" use="required"
    match="/time:time[@time:hours<6 or
      @time:hours>20]"/>
</context>

```

With these context patterns, the associated configuration must only be applied if the location type is “Home” and it is sooner than 6:00 or later than 20:00 which can be considered “night time”. Both patterns are mandatory to exist and fulfill.

Again, we want to stress that there is a subtle but important difference among the context-related information structures and the “configuration of preferences”-related structures. Context information represents *state information* that the service provider is able to check, either directly from databases or files, or indirectly by requesting the state from some originating sources. In both cases, that state information must be structured in XML format meeting the requirement of an associated grammar, possibly in the form of a XML Schema. That XML formatted state information is the target of the XPath expressions in the context section of the profile. So, we call *context domains* to the set of domains of knowledge the service is aware of.

On the other hand, preferences configuration information represents *domains over which the service keeps control* to make changes to fulfill user preferences and drive the system towards the desired state. The service can implement those changes invoking some low level functions on actuators, or invoking operations on other devices, for instance. The selected mechanism is up to the service and out of the scope of the WebProfiles model. So, we call *configuration domains* to the set of domains of knowledge over which the service keeps control

WPML has some additional but powerful features such as variables and complex data structures that can be declared and used as comparison values both in patterns and preferences.

XPath is the preferred element addressing language as well, but WPML is open for other mechanisms such as XQuery, just establishing the `querylang` attribute at the `<wpm1>` element (in our current implementation only XPath is supported).

At this point, WPML is powerful enough to express user’s profiles relating context and configuration information about different domains the service is

supposed to be aware and control (some of them). Next, we will illustrate how the user-agent can determine the supported context and configuration domains for the service, so that it can generate and send the right profiles in WPML.

## 4. The WebProfiles model

The goal of the WebProfiles model is to provide an HTTP-based mechanism to negotiate and convey user preferences information to obtain more adapted results when interacting in personal area webs. The user-agent, acting as the client, is the unique entity that manages the user preferences repository, providing the authorized services with the appropriate subset to generate adaptation.

The client repository stores user-related profiles on different knowledge domains, allowing several profiles about the same domains to be supported and to be applicable to different service scenarios.

The point with the WebProfiles model is that information is not statically determined, but it is dynamically generated depending on the situation by selecting and grouping the convenient profiles and forwarding them to the service provider. That is why we can state that the WebProfiles model provides user-related context-awareness in the Personal Area Web vision.

The elements that define the situation and, thus, influence the selection of profiles are: the involved domains of knowledge, the service provider related information, the user’s established permissions about profile information access, and the existence of suitable profiles to convey.

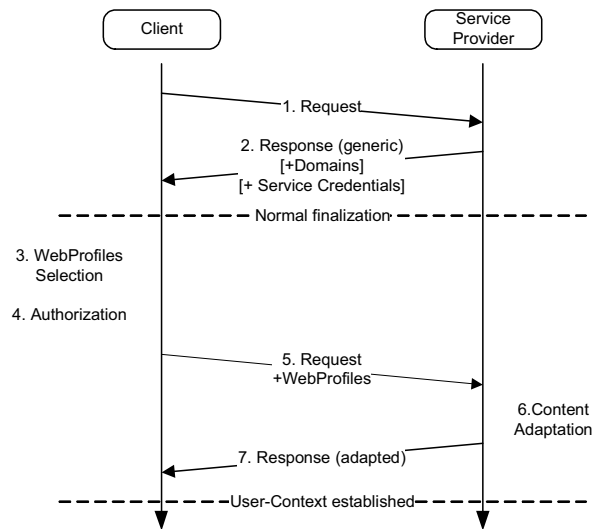
All these entities’ data serve as criteria to negotiate and exchange the user-related context information with the service provider, and so, set up the environment for further services execution.

### 4.1. Negotiation

The WebProfiles model defines an HTTP-based negotiation mechanism that allows both client and service providers to set up the user-related context in which further interactions can be performed.

The most remarkable phases within this negotiation process involve notification of domains of knowledge about profiles, profile transmission and service adaptation.

The following diagram illustrates the negotiation process at a higher level, stressing the sequence of tasks each party must accomplish.



**Figure 3. The WebProfiles negotiation process**

The detailed description of each step is:

1. The user-agent, a mobile phone or PDA process that represents the user, issues a normal request to get some resource from the service provider.
2. The service provider, a web-based device, processes the request and sends back the resource (in a generic form) along with the list of context and configuration domains through which the user agent can express profile information. Optionally, the service provider can attach some kind of Service Credentials certifying the privacy for user profile if sent. These Service Credentials can take the form of a P3P policy [7]. If the client does not support WebProfiles or the domains to express preferences, or it does not validate credentials, or it does not require adaptation for this service, the negotiation process ends at this point as if it was a normal finalization without WebProfiles.
3. If the client demands service adaptation, it checks the context and configuration domains to select all the stored WebProfiles that express user preferences.
4. The client filters the list of candidate WebProfiles using the Service Credentials supplied by the service provider, and thus obtaining the final list of validated WebProfiles suitable for that concrete service adaptation.
5. The client issues the original request adding the validated WebProfiles.

6. The service provider uses the information conveyed in the received WebProfiles to better know the client and adapt the further responses and the overall service.
7. The service provider generates the corresponding response to the request, conveniently adapted by means of the WebProfiles. Now, the user-contextual information is established between the user-agent and the service provider for further exchanges.

After negotiation, the service provider *knows* the user and *gets aware* of his preferences as if it was a returning visitor, despite maybe it is the first time the user accesses the device.

This interaction model illustrates the process of contextualization via WebProfiles. In the case user-context information is not needed or WebProfiles are not supported neither by the client or the service provider, the interaction finishes at step 2 and the overload is minimal in relation to the normal process.

Only if WebProfiles are applicable and agreed by both parties, a further interaction is required where WebProfiles are exchanged in an overall process that resembles HTTP Basic Authentication [12], in the sense that the client is the responsible for resending the original request extended with additional information to obtain a preferred response (client-driven negotiation).

In fact, this resemblance is not casual. The WebProfiles model has been designed in such a way that shares many similarities with existing HTTP mechanisms in order to be easily integrated within the hypertext protocol.

Nevertheless, the WebProfiles negotiation model does not follow an strict client-driven or server-driven negotiation model as specified in [13], but it shares hybrid characteristics with both of them as it is explained in the following sections.

## 5. WebProfiles HTTP extensions

Since WebProfiles are intended to be applied in Ubiquitous Computing web-enabled scenarios, the use of HTTP as the supporting protocol for negotiation is more than evident. Although the primary goal is reusing the most functionality present today, some tasks in the WebProfiles negotiation process require extra protocol information to be exchanged between clients and service providers.

The WebProfiles model has been designed with a clear orientation to the web paradigm, which is reflected not only in the name itself, but also in the synergies with other HTTP technologies. WebProfiles

can and should be used in conjunction with HTTP mechanisms such as HTTP Multipart Messages [14], cookies [3] and HTTPS [15] secure communication to enhance the context establishment process under certain scenarios.

### 5.1. Identification mechanism

The WebProfiles model requires the definition of an identification mechanism that allows clients and servers to identify profile instances unambiguously.

Even if the profile document is syntactically the same, the identification tag must be different if it was generated by distinct parties or in different periods of time. There must be a unique “WebProfile ID” for every profile expressing user requirements for adaptation; so that client and servers can check the WebProfiles they share, avoiding the need to exchange profiles once and again, by checking only IDs.

Analysing the identification mechanisms traditionally used in HTTP, none of them was found appropriate. The ETag format [13] is not suitable by definition and cannot be used for universal identification purposes due to its nature (collisions can easily appear). An MD5 digest [16] represents a digest only dependant on the content, which means that two user-agents that create the same WebProfile information would associate it to the same MD5 identifier. That is not a problem now, since the service provider can associate a cookie to the WebProfile and distinguish among clients with identical WebProfile IDs.

However, we have in mind some future extensions of the WebProfiles model, out of the scope of this paper, that allow the service providers to update profiles at the user-agent side (of course, in those domains in which the client allows the servers to do so). Two servers could generate the same MD5 digest over the same content, causing conflicts in the client for distinguishing one profile from the other without extra metadata information.

Finally, the URN UUID format [17] was found successful for this task. It assigns a universal unique identifier while being an URI after all, so it fits perfectly in the web model.

An example of such identifier is:

```
urn:uuid:fede9406-5151-4a10-8d26-7d6908ae7559
```

### 5.2. WebProfiles HTTP headers

In this section we are going to start introducing the extensions required in HTTP to support the WebProfiles negotiation process, which take the form of new HTTP headers for different purposes.

To illustrate the use of WebProfiles HTTP headers we will step through an example client-server interaction with successful profile exchange, omitting obvious traditional HTTP headers (Content-Length, Connection, Host, ...) for stressing the importance of new ones. Also, we do not use in the example the HTTP Extension mechanism [24] for the sake of clarity, but any implementation should apply it.

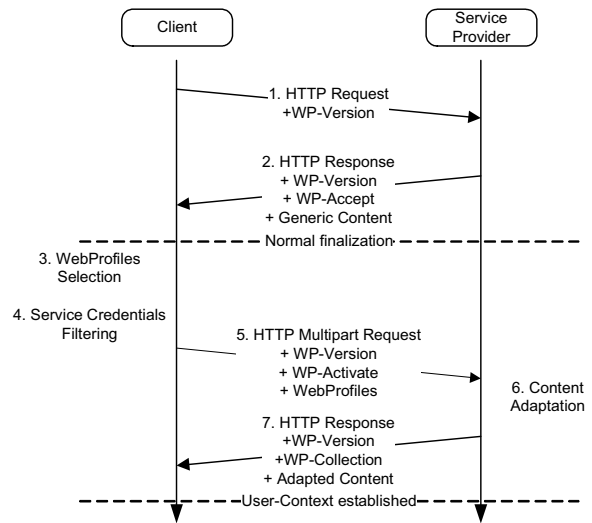


Figure 4. The WebProfiles HTTP-based negotiation process

In this example we suppose that the user-agent has some preferences configured about desired temperature depending on location and time, it is the first time contacting this particular server, and has just downloaded the P3P privacy policy from it, verifying there are no conflicts with user policy about sending WebProfiles. Graphically, the interactions can be represented following the scheme depicted in Figure 4.

```

GET /service HTTP/1.0
WP-Version: 1.0

HTTP/1.0 200 OK
WP-Version: 1.0
WP-Accept: text/vnd.webprofiles.wpm1+xml;
           ctx-1="http://www.webprofiles.org/dataschemas/location";
  
```

```

    ctx-2="http://www.webprofiles.org/dataschemas/time";
    cnf-1="http://www.webprofiles.org/dataschemas/ambient"

<!-- Generic content: Temperature control device ready -->

POST /service HTTP/1.0
WP-Version: 1.0
WP-Activate: urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6

--multipart_separator
Content-Type: text/vnd.webprofiles.wpm1+xml
WP-Content-URI: urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6

<!-- Content of the WebProfile with urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6 -->
--multipart_separator--

HTTP/1.0 200 OK
WP-Version: 1.0
WP-Collection: urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6; max-age=300

<!-- Adapted content and action performed: Temperature configured between 20°C and 30°C.
    Temperature control device ready -->

```

**5.2.1 WP-Version.** The WP-Version header merely notifies the other party about the version of the WebProfiles specification one uses. The user-agent sends this header to inform the server about WebProfiles support.

**5.2.2. WP-Accept.** In the step 2 of the Figure 4, the service provider agrees the WebProfile version and indicates in the WP-Accept header the list of context and configuration domains accepted for service adaptation. First, the MIME type of the accepted format for profiles is included (text/vnd.webprofiles.wpm1+xml), along with context and configuration domains in the standardized form of namespaces, via numbered parameters.

In the above example, the service provider informs the user-agent about two context domains (against which evaluate adaptation conditions) and one configuration domain (against which execute adaptation).

The HTTP response message includes the entity content referred by the request URI without adaptation, in the generic form, which is valid for the user-agent if the interaction ends at this point.

If the user-agent has some valid WebProfiles associating the context and configuration domains accepted by the server, those candidate WebProfiles are selected and checked against the P3P policy file that declares the intended use of the data by the service provider. After filtering, the user-agent gets the final list of validated profiles to send to the device.

**5.2.3. WP-Activate.** Now, the client can resend the original request including the validated WebProfiles. Since every validated WebProfile document must be included in the request message, the format of such is an HTTP POST multipart message where each part contains a particular WebProfile document along with description headers such as Content-Type, Content-Length, and the WP-Content-URI header (explained below).

Previous to each multipart section, a new response header WP-Activate is included to specify the URN UUIDs of the validated WebProfiles that must be used to perform service adaptation. For example, if two WebProfiles were selected, the header could be:

```

WP-Activate: urn:uuid:23adf57b-cfa2-11d0-
aad3-00a0c91e6bf6, urn:uuid:faef81d4-0c9-
11d0-a765-00a0c91ef5da

```

Every multipart section in the request message with a WebProfile content must include at least the Content-Type header (with the supported value of text/vnd.webprofiles.wpm1+xml) and a WP-Content-URI header that identifies unambiguously the associated WebProfile.

Note that the original HTTP GET request has been transformed to a HTTP POST request: URIs supporting WebProfiles adaptation should be accessible via POST requests to receive WebProfiles along with the request. An alternative solution could be



to include the complete WebProfiles in the headers as other technologies do such as [19].

However, we think that multipart POST messages are more suitable, clear and graceful for these tasks, and supporting POST requests are a usual feature for any URI, as well as a mechanism widely used by other protocols such as SOAP to convey data [2,10].

**5.2.4. WP-Content-URI.** The WP-Content-URI header is an entity header that associates a universally unique identifier to the accompanied entity. Its purpose is to identify unambiguously an information entity, so that can be referenced from other headers (mostly from WP-Activate and WP-Collection), but also declares the identification tag for the content entity that will be used by clients and servers.

Other header candidates for entity identification such as the ETag [13], Content-Location [13], Content-MD5 [20] or even the Content-Disposition [20] header were discarded because of inconvenience for universal identifying purposes as stated previously.

**5.2.5. WP-Collection.** In the step 7 of the Figure 4, the service provider issues a WP-Collection header conveying the URN UUIDs of the WebProfiles sent by the client in the request and found successful for service adaptation. The purpose of the WP-Collection header is to inform the client about the WebProfiles associated in the service provider, and used to establish the user-context and generate the adapted content. An example of two WebProfiles accepted would be:

```
WP-Collection: urn:uuid:23adf57b-cfa2-11d0-  
aad3-00a0c91e6bf6; max-age=300,  
urn:uuid:faef81d4-0c9-11d0-a765-  
00a0c91ef5da; max-age=600
```

The max-age parameter informs the user-agent about the period of time (number of seconds) that concrete profile is going to be active at the device. The client should actively renew its influence over the service provider, by sending a request containing the WP-Activate header listing the WebProfiles to renew before expiration.

This mechanism puts the charge of coping with rapidly changing contexts in the user-agent side, which is the unique party that initiates interaction in the HTTP model. This makes the user-agent explicitly aware of the period of influence of the adaptation, especially important in shared resources (referred by URIs) where users have to hand over the rights to each other, such as a TV set watched by several users at the time.

Finally, the service provider also includes the adapted content in the response.

At this point, the user-context is established, the device knows the user preferences and under which conditions must be activated without user explicit input. Those profiles can be sent once and again to different devices without user manual intervention to automatically adapt every device to his preferences.

### 5.3. Protocol usage

These proposed HTTP headers are intended to be applied in Ubiquitous Computing scenarios enabled by HTTP-based mechanisms as UPnP, strongly based in the HTTP protocol throughout all the communication processes.

We have developed a WebProfiles adaptation to UPnP as explained in [28]. During UPnP description and control phase, a surrounding device can inform about supported vocabularies and be influenced by the user's personal device embedded UPnP communication agent by attaching suitable WebProfiles in a POST message.

Although not yet implemented, P3P descriptions available in surrounding devices can be obtained by the user-agent and checked against the internal WebProfiles privacy policy.

## 6. Conclusion and future work

The WebProfiles model adds a convenient extension to the HTTP protocol in order to support automatic customization and adaptation in Ubiquitous Computing environments populated with devices and embedded services, where these are automatically configured to match user preferences and requirements.

Our current implementation takes the form of a background process for PocketPC which discovers surrounding devices using UPnP, and negotiates user preferences to adapt the environment, made up of small HTTP server-embedded devices.

The WebProfiles can be created by the user via UI wizards or inferred from past manual interactions with devices which can be recorded and analyzed by pattern-detection or heuristic tools.

The use of well-known standards such as HTTP, XML or XML Schemas guarantees the stability and coherence of the model itself, while retaining the extensibility that can be added by using accompanying web technologies such as HTTPS. The WebProfiles model relies also on P3P technology for validating the use of the preferences by the device against user privacy policy.

A more optimized method for WebProfiles updates at the server, via the WP-Collection and WP-Accept headers in message interactions could be implemented,

maybe exchanging only affected WebProfiles sections and not the entire document, so we are considering the use of delta encoding for HTTP [25].

The WebProfiles model shares some similarities with other semantic ubiquitous computing initiatives such as CoBrA [21] and Task Computing [27], but it is more focused on the concept of passive influence [5], instead of actively commanding the environment. Both CoBrA and Task Computing use semantic knowledge for reasoning and understanding user's context, while WebProfiles is based on XML information structures without semantic capabilities which makes it less powerful, but easier to implement in limited-capability devices.

However, we are planning to incorporate semantic technologies to express preferences using RDF [22] and OWL [23], as well as semantic rules expressed via the promising standard SWRL (SemanticWeb Rule Language) [26].

We foresee that the use of Semantic WebProfiles would allow the expression of context patterns and preferences by means of their real relationships and it is one of our future goals.

## 7. Acknowledgements

This work has been partially supported by the Department of Industry, Commerce and Tourism of the Basque Government under the SAIOTEK grant S-OD04UD02, and the Cathedra of Telefónica Móviles España at Deusto University, Bilbao, Spain.

## 10. References

- [1] UPnP Forum. UPnP Device Architecture v1.0. 2000. <http://www.upnp.org>.
- [2] Issarny, V., Sacchetti, D. et al. Developing Ambient Intelligence Systems: A Solution based on Web Services. *Journal of Automated Software Engineering*. 2004.
- [3] Kristol, D., and Montulli, L. RFC 2965: HTTP State Management Mechanism. IETF RFC. 2000.
- [4] St. Laurent, S. *Cookies*. Computing Mcgraw-Hill. 1998.
- [5] Vazquez, J.I., and López de Ipiña, D. An Interaction Model for Passively Influencing the Environment. *Adjunct Proceedings of the 2nd European Symposium on Ambient Intelligence* (Eindhoven, The Netherlands). 2004.
- [6] Hensley, P. et al. Implementation of OPS Over HTTP. 1997. <http://www.w3.org/TR/NOTE-OPS-OverHTTP.html>
- [7] P3P Specification Working Group. The Platform for Privacy Preferences 1.1 (P3P1.1) Specification. W3C Working Draft. 2004. <http://www.w3.org/TR/P3P11/>
- [8] Coyle, K. P3P: Pretty Poor Privacy? A Social Analysis of the Platform for Privacy Preferences (P3P). 1999. <http://www.kcoyle.net/p3p.html>
- [9] Little, M., Newcomer, E. and Pavlik, G. (eds.). Web Services Context Specification (WS-Context). OASIS Committee Draft v.0.8. 2004.
- [10] XML Protocol Working Group. SOAP Version 1.2 Part 0: Primer. W3C Recommendation. 2003.
- [11] Web Services Description Working Group. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft. 2004.
- [12] Franks, J. et al. RFC 2617: HTTP Authentication: Basic and Digest Access Authentication. IETF RFC. 1999.
- [13] Fielding, R. et al. RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1. IETF RFC. 1999.
- [14] Freed, N. and Borenstein, N. RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. IETF RFC. 1996
- [15] Rescorla, E. RFC 2818: HTTP Over TLS. IETF RFC. 2000.
- [16] Rivest, R. RFC 1321: The MD5 Message-Digest Algorithm. IETF RFC. 1992.
- [17] Leach, P., Mealling, M. and Salz, R. A UUID URN Namespace. Internet Draft. 2004. draft-mealling-uuid-urn-03.txt.
- [18] W3C Device Independence Working Group. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation.
- [19] P3P Specification Working Group. Platform for Privacy Preferences (P3P) Syntax Specification. W3C Working Draft 26 August 1999.
- [20] Troost, R., Dorner, S. and Moore, K. RFC 2183: Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field. IETF RFC. 1997.
- [21] Chen, H., Finin, T., and Joshi, A.. An Intelligent Broker for Context-Aware Systems. *Adjunct Proceedings of Ubicomp 2003*. 2003.
- [22] RDF Core Working Group. RDF Primer. W3C Recommendation. 2004. <http://www.w3.org/TR/rdf-primer/>

[23] Web Ontology Working Group. OWL Web Ontology Language Overview. W3C Recommendation. 2004. <http://www.w3.org/TR/owl-features/>

[24] Nielsen, H. et al. RFC 2774: An HTTP Extension Framework. IETF RFC. 2000.

[25] Mogul, J. et al. RFC 3229: Delta encoding in HTTP. IETF RFC.2002.

[26] Horrocks, I. et al. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Draft version 0.7. DARPA DAML Program, 2004. <http://www.daml.org/rules/>

[27] Masuoka, R., Labrou, Y. and Song, Z. Semantic Web and Ubiquitous Computing - Task Computing as an Example. *AIS SIGSEMIS Bulletin, Vol. 1 No. 3.* 2004.

[28] Vazquez, J.I., López de Ipiña, D. Empowering wireless UPnP devices with WebProfiles. *10th IFIP International Conference on Personal Wireless Communications - PWC'05.* 2005.