CC/PP and UAProf: Issues, Improvements and Future Directions

Mark H. Butler
Information Infrastructure Laboratory
HP Laboratories Bristol
HPL-2002-35
March 13th , 2002*

E-mail: mark-h_butler@hp.com

DELI, UAProf, CC/PP, delivery context, device independence, WAP

Different web-enabled devices have different input, output, hardware, software, and network capabilities. In order for a web server to provide optimized content to different clients it requires a description of the capabilities of the client known as the delivery context. Recently two new compatible standards have been created for describing delivery context: Composite Capabilities / Preferences Profile (CC/PP) created by the W3C and User Agent Profile (UAProf) created by the WAP Forum. This paper describes a number of issues, possible improvements and future directions for CC/PP and UAProf that have been identified in the course of work at HP Labs Bristol on device independence, and the creation DELI, an experimental open-source library that simplifies deploying these standards within servlets.

* Internal Accession Date Only Approved for External Publication
W3C Workshop on Delivery Context, 4-5 March 2002, Sophia-Antipolis, France

CC/PP and UAProf: Issues, Improvements and Future Directions

Dr Mark H. Butler (mark-h_butler@hp.com)
Hewlett Packard Laboratories, Bristol UK
1 February 2002
Position Paper for the W3C Workshop on Delivery Context

Abstract

Different web-enabled devices have different input, output, hardware, software, and network capabilities. In order for a web server to provide optimized content to different clients it requires a description of the capabilities of the client known as the delivery context. Recently two new compatible standards have been created for describing delivery context: Composite Capabilities / Preferences Profile (CC/PP) created by the W3C and User Agent Profile (UAProf) created by the WAP Forum. This paper describes a number of issues, possible improvements and future directions for CC/PP and UAProf that have been identified in the course of work at HP Labs Bristol on device independence[1,2,3] and the creation DELI[4], an experimental open-source library that simplifies deploying these standards within servlets.

1. Introduction

Recently two new compatible standards have been created for describing device delivery context: Composite Capabilities / Preferences Profile (CC/PP)[5] created by the W3C, and User Agent Profile (UAProf) created by the WAP Forum[6]. Both these standards are based on the W3C foundation for processing metadata, the Resource Description Framework (RDF)[7,8,9]. RDF aims to provide interoperability between applications that exchange machine-understandable information on the Web. RDF represents metadata as *models* that consist of a collection of *statements* about *resources*. Statements consist of a *subject*, a *predicate*, and an *object*, where subjects are always resources and objects are resources or literals. RDF models can be serialized using XML and most people first encounter CC/PP profiles in this form. However, viewing RDF in XML serialized form is deceptive; it is important to appreciate that the underlying RDF model is a graph where nodes represent resources or literals. The W3C RDF Validation Service[10] is a useful tool for investigating the relationship between the XML serialisation of RDF and the underlying RDF model.

CC/PP is a particular application of RDF metadata and is based on a description of device capabilities and user preferences known as a profile. Typically this profile is sent by the client to the server to guide the creation, adaptation or selection of content presented to the client device[11,12,13]. Essentially a CC/PP profile is constructed as a two level hierarchy: it has a number of *components* each possessing a number of *attributes*. UAProf is a variant of CC/PP created specifically for use with WAP phones that defines a specific vocabulary, plus a set of resolution rules to merge profiles and a protocol to transmit UAProf information in HTTP requests.

Recently HP Labs has released DELI[14], an open-source library that allows Java servlets to resolve HTTP requests containing delivery context information from CC/PP or UAProf capable devices. This paper aims to highlight a number of issues with CC/PP and UAProf that have arisen in the course of this work and to suggest possible improvements and future directions for these standards.

2. Profile Resolution

During the implementation of DELI, a number of issues have arisen that result from the decision to use RDF as the basis for CC/PP. Firstly we have found that the profile resolution mechanism specified in UAProf is difficult to implement in RDF. Both CC/PP and UAProf clients split up profile information in order to send it to the server in an efficient way. They do

this by using a standard profile, known as a *reference profile*, and a list of overrides specific to the requesting device known as a *profile diff*. Other devices in the communication path, such as proxies, may also add profile diffs. The process of reassembling the final profile from the reference profile(s) and profile diff(s) is known as *profile resolution*. CC/PP does not specify the exact mechanism for profile resolution, apart from requiring that default attribute values are always overridden by non-default attribute values. UAProf on the other hand specifies a set of *resolution rules* that apply to non-default values. Each attribute in a vocabulary is associated with a specific resolution rule that is applied when multiple attribute values are encountered. In UAProf these resolution rules are order dependent; for example, *locked* means take the first value encountered whereas *override* means take the last value encountered. Unfortunately these rules are difficult to implement in RDF, as RDF models do not have any implicit concept of ordering statements. Ordering must be done explicitly, e.g. using an RDF Sequence. Unlike RDF, XML does implicitly order elements in documents. As statements in an RDF model are unordered it is impossible to apply the UAProf resolution rules to a single RDF model. Possible solutions include representing each profile or profile-diff as a separate model and keeping track of the order of these models. This allows resolution to be performed between models. An alternative approach taken in DELI is to convert profiles to an intermediate data structure that stores attribute order before performing profile resolution. The problem could also be avoided by processing the CC/PP profile with an XML processor rather than an RDF processor. Unfortunately this causes other problems as discussed in the next section.

3. XML versus RDF parsers

It seems from discussions on the UAProf mailing list[15] that most people implementing server support for UAProf are using XML parsers rather than RDF parsers. DELI on the other hand, parses profiles using the Jena[16] RDF framework. Parsing RDF is non-trivial when using an XML parser[17] because identical RDF models can be serialized in many different ways. In UAProf this problem has been partly resolved by specifying a number of restrictions about what RDF expressions are allowed. For example UAProf only allows the use of XML elements to define properties whereas RDF allows the use of XML elements or XML attributes. One reason for avoiding RDF is that RDF tools have only recently become available. Several implementers have also cited efficiency as a concern, we have found that typical RDF parsers such as RDFFilter[18] and ARP[19] are respectively 5 and 20 times slower than standard XML parsers such as Xerces[20]. This may reflect the maturity of XML parsers in comparison with RDF parsers rather than any inherent inefficiency.

It is also common to see RDF being used incorrectly in profiles or even official schemas[21] or specifications. Typical errors include not stating that `ID` or `Resource` is in the RDF namespace, using non-standard namespace URIs for RDF or RDF schema, or syntactic errors in URIs such as pre-pending or appending spaces. People working with RDF are strongly urged to use the W3C validation service[10] which will identify most of these errors. Furthermore, greater priority needs to be given to updating public documents or schemas as otherwise errors are often unwittingly replicated. Otherwise there is a danger that CC/PP or UAProf will develop its own incompatible dialect of RDF.

4. Typing components

Another problem relating to the use of RDF in CC/PP is the way components are specified e.g.:

```
<prf:component>
  <rdf:Description rdf:ID="HardwarePlatform">
    <rdf:type
       rdf:resource =
"http://www.wapforum.org/profiles/UAPROF/ccppschema20010430#HardwarePlatform"/>
    <prf:BitsPerPixel>2</prf:BitsPerPixel>
   </rdf:Description>
</prf:component>
```

Many people seem to find it counter-intuitive that you need to declare that the component is called HardwarePlatform twice. It is important to note the first use of HardwarePlatform defines the instance name whereas the second use is to define the type. This is just as if we defined an object in Java e.g.

```
HardwarePlatform hardwarePlatform;
```

In Java we would not expect the complier to determine the object type from the instance name, but some CC/PP implementers omit the type statements from components. This means the CC/PP processor cannot recognise components as the `ID` is just a local name. Recent versions of DELI resort to inferring component type from attribute name when they encounter profiles that omit type statements. Hence it is not currently clear what purpose components serve, which leads the author to question whether they are necessary? If components are necessary, RDF already has a syntactic shortcut for such situations called the typedNode which allows us to rewrite the RDF above in a much more compact form as shown below. In the author's opinion this form is much more suitable for CC/PP and UAProf.

```
<prf:component>
  <prf:HardwarePlatform rdf:ID="HardwarePlatform">
     <prf:BitsPerPixel>2</prf:BitsPerPixel>
  </prf:HardwarePlatform>
</prf:component>
```

5. RDF Schemas and CC/PP Vocabularies

While implementing DELI it was found that RDF schema is not sufficiently expressive to describe CC/PP vocabularies. For example in UAProf each attribute is associated with a resolution rule and attribute type. This information is essential in order for the UAProf processor to perform profile resolution. Currently the UAProf schema overcomes this problem by storing this information in a comments field. It is possible to retrieve this information by parsing the comments field, as done in the current version of DELI, but really this information should be properly tagged so it can be retrieved directly by an XML or RDF parser. One way to do this would be to create a CC/PP-specific extension to RDF schema that allows additional vocabulary information to be included via a CC/PP schema namespace. This extension does not need to be complex - it could just be a standard mechanism for defining default property values for attributes so that values can be assigned to the resolution rule and attribute type of an attribute. The key issue here is that further work is necessary to make CC/PP a vocabulary independent standard. In the author's opinion, CC/PP should be sufficiently well specified so that it is possible to implement a CC/PP processor that when supplied with an appropriate CC/PP vocabulary schema, can process any possible CC/PP vocabulary. With the current generation of CC/PP processors, too much vocabulary specific knowledge is being hard-coded into CC/PP processors.

6. Processing CC/PP using XSLT

It is difficult to process CC/PP profiles using existing XML tools such as XSLT. For example XSLT is widely used in XML publishing frameworks to adapt XML documents to multiple output devices to support device independence. Therefore one possible use case for using CC/PP profile information is within XSLT stylesheets[22]. Recently the author has been working on incorporating CC/PP and UAProf support into the Apache Cocoon[23] XML publishing framework using DELI. This work has shown that it is easier to manipulate profiles in XSLT if they are 'flattened' first i.e. they just consist of a list of profile attributes with all extraneous RDF removed. This approach is not ideal as it means XSLT authors encounter profiles in a non-standard form. Other possible solutions include creating an extension for XSLT for querying RDF models.

7. Late conversion of CC/PP to RDF

Many of the previous points highlight difficulties that arise when using CC/PP because it is based on the XML serialisation of RDF rather than XML. A (controversial) solution to

several of these problems is to re-design CC/PP so that profiles are represented in a compact XML form. This would simplify the task of processing profiles using XML processors and avoid the creation of an incompatible dialect of RDF for CC/PP. In addition it would reduce the size of profiles, a desirable characteristic for low bandwidth networks. Furthermore such an approach would not rule out the use of RDF models with CC/PP; if we require RDF support, we can use a transform on the compact XML representation to transform it to RDF if so required.If this solution appears unattractive due to the effort invested in CC/PP, it is vital that we do due diligence to ensure we are using RDF correctly.

8. Dealing with multiple vocabularies

So why was RDF used as the basis of CC/PP? Unlike XML, RDF has been designed to cope with data from disparate sources using different vocabularies. RDF can cope with different vocabularies because it associates each vocabulary with a different namespace. When CC/PP was created, it was expected that the creation of multiple vocabularies for device profiles was unavoidable. RDF seemed to provide a way of addressing this problem. Unfortunately, processing multiple vocabularies presents additional problems beyond distinguishing between vocabularies. For example, at present, at least two vocabularies for UAProf exist[i], but the UAProf specification does not define how a processor should process a request that uses more than one UAProf vocabulary. Each vocabulary is associated with a different namespace so one possible interpretation could be for the processor to assume that the attributes, even if they have the same name, are totally different because they are in different namespaces. Other interpretations are possible. Current UAProf vocabularies contain some attributes that are identical in all vocabularies. Hence we might expect these attributes to be merged using conventional resolution rules. Other attributes are similar but have undergone changes e.g. they have moved component or have changed resolution rule. The required behaviour in these circumstances is even more uncertain. It seems that we need to decide how to merge different vocabularies and to define a way of describing this merge process.

Processing metadata described in multiple RDF namespaces is a key problem for the Semantic Web. Research is currently focussing on how ontologies can be used to map between metadata in multiple disparate namespaces. Ontologies may not be the ideal solution to all problems: they are complicated, based on work in artificial intelligence and still in at the research stage. Therefore an alternative and possibly better solution is the use of modular vocabularies. Here instead of replicating attributes so they exist in multiple namespaces for each vocabulary type, we create small "sub-vocabularies" of standard attributes with a single namespace. Specific vocabularies such as UAProf then reuse these sub-vocabularies rather than replicating attributes. For example in the current UAProf vocabularies both the November 1999 and May 2001 versions use the attribute `BitsPerPixel` in the component `HardwarePlatform`. In the author's opinion it would have been better to re-use the original `BitsPerPixel` in the original namespace than create a new version as this creates additional problems for processors as they need to map between the two versions. Sub-vocabularies would also reduce the total number of attributes in common use; something that can only simplify matters for content authors or server developers. One possible problem with sub-vocabularies is the possible proliferation of components of the same type. Currently you would need an instance of a component for each component and each namespace used by an attribute in the profile. This could result in multiple instances of the same component where each component is associated with a different namespace. This is another unnecessary complexity introduced by components.

[i] An aside: the reason for the previous statement being qualified by "at least" is that some profiles currently use namespace URIs that do not correspond to any available schema. Again, due diligence is required to ensure that profiles and schemas are correct and consistent. Although RDF schemas may not be necessary for some uses of RDF, they are necessary for profile resolution in CC/PP as discussed in Section 5.

9. Defining standards for attribute values as well as attributes

There is an additional problem for content authors with the way UAProf is currently used. Although the UAProf specifications define several features of each attribute they do not define the range of values or in the case of literals the set of possible value and the definitions of those values. For example, the UAProf specification states that possible example values for `Keyboard` are "`Disambiguating`", "`Qwerty`" and "`Keypad`". However it is possible that different phone manufacturers will use attribute values in different ways. For example one manufacturer might call a keyboard "`Keypad`" whereas another one might call it "`TelephonePad`" when they are both trying to describe the same functionality. This causes a problem for the content author or server developer as now they need to know how manufacturers have defined the values.

10. Representing complex alternatives in profiles

There is another standard for describing device capabilities that precedes CC/PP called media feature sets[24, 25, 26, 27, 28, 29, 30]. One interesting aspect of this standard is that it allows profile attributes to be joined explicitly by ANDs and ORs. Early drafts of the documents defining CC/PP also featured profiles with this type of structure, but the current version only features profiles that implicitly use ANDs and ORs i.e. all the attributes in a CC/PP profile represent constraints joined by ANDs except where an attribute contains multiple values. Here the processor decides whether to interpret the values as being joined by an AND or an OR. This means that CC/PP profiles can only possess ORs at leaf nodes i.e. you cannot have ORs of ANDs. Such profiles are limited in their expressive power. For example you might want to express that a device can accept landscape or portrait documents and these modes are associated with different screen resolutions. Alternatively you might want to associate MIME types with quality measures as in HTTP/1.1 content negotiation[31]. Note that this does not mean such profiles cannot be created in CC/PP; as CC/PP is based on RDF we can use standard RDF mechanisms to do this. However because profiles with this structure are not described explicitly in the CC/PP documents, processors will not able to process them. As already noted, the CC/PP specification needs to be sufficiently complete so that we can define general purpose processors that can process any CC/PP vocabulary.

There is one problem with using nested ANDs and ORs in profiles. In the media feature sets, it was necessary to normalise profiles to a canonical form known as "disjunctive normal form". This normalisation represents additional work at the profile resolution stage. If possible we should try to make profile resolution as efficient as possible. Therefore it may be desirable to determine whether it is reasonable to restrict the use of ANDs and ORs so that profiles come "pre-normalised" while gaining the extra expressability that such constructs provide.

11. Preference Ordering

Preference ordering is used in HTTP/1.1 content negotiation by both the browser and the content author in order to indicate a preference for a particular alternate when multiple alternates match the capabilities and requirements of the target device. An important use case for this is language. On a multilingual site, content may be available in several different languages. The user may understand several languages to varying degrees e.g. they may prefer French but be willing to accept English if French is not available. In HTTP/1.1 the browser can communicate this to the server by attaching "q" values to language alternatives and giving French a higher q value than English. UAProf uses an ordered set (a `Seq` construct in RDF) in order to specify language preference. This causes some additional complexities as demonstrated in the following situation: a phone has a default value of English in its reference profile, but the user sets the phone to French which is sent to the server using a profile diff. There is a potential problem here: as `CcppAccept-Language` uses the *append* resolution rule, the French attribute value will always come after the English value so the phone will still return English in preference to French. In order to avoid this problem, the phone must specify language as a default as defaults are always overridden regardless of resolution rule. Clearly

phone manufacturers need to carefully consider the interaction between preference ordering, defaults and resolution rules if they allow the user to change certain UAProf attributes.

12. CC/PP Protocols

Currently CC/PP does not have an official protocol, as protocols were not in the original scope of the working group. However there is a W3C note that describes a possible protocol for CC/PP based on HTTP-ex, the HTTP-extension framework[32],[33]. In the author's opinion the choice of HTTP-ex is unfortunate as it is an experimental protocol that is not widely supported. HTTP-ex has two important differences from HTTP/1.1: it defines a number of new methods e.g. M-GET (mandatory get) and it associates new request header fields with URI's via numerical namespaces.

The usefulness of M-GET is arguable. This instructs the server only to return content if it can interpret the enclosed CC/PP profile. If not it should return an error. Servers that are not HTTP-ex aware will also return an error, as they do not recognise the M-GET method. However the same functionality can be achieved without having to create new HTTP methods as demonstrated by the UAProf W-HTTP protocol. If a server is CC/PP aware, it can append an additional header to the response indicating if it has used the profile while processing the request. If a server is not CC/PP aware then it will not include this header. This allows a client to decide whether to try to display the content anyway or whether to report an error.

The usefulness of numerical namespaces is also arguable. HTTP-ex was proposed at a time when it seemed likely a large number of non-browser applications would use HTTP with non-standard header request fields. Numerical namespaces were seen as a way of preventing possible conflicts between these new request header fields. However the introduction of SOAP[34] has shifted the argument up a layer in many of these applications. This has resulted in decreased interest in HTTP-ex: for example even now there are very few HTTP-ex capable webservers available apart from the Jigsaw server[35].

DELI currently uses the W-HTTP UAProf protocol rather than the CC/PP-ex protocol. This is functionally equivalent to the CC/PP-ex protocol but omits the use of M-GET and numerical namespaces so is far easier to implement on existing servers. Therefore the author would prefer to see CC/PP and UAProf standardise on a single protocol derived from W-HTTP UAProf. Standard protocols are essential in order to support device independence.

13. Profile Resolution in CC/PP

It is important to note that the UAProf specification has a more complete treatment of profile resolution than CC/PP even though profile resolution is essential to both standards. If future versions of CC/PP specify standard profile resolution mechanisms, it seems desirable that they are compatible with UAProf. Unfortunately the current UAProf mechanisms are not ideal as they perform resolution based on numerical order. An alternative approach to resolution rules could involve making the parties involved explicit, i.e. for this attribute the proxy is more important than the device, or for this attribute the device overrides are more important that the proxy overrides etc. Future work could investigate whether explicitly identifying parties in this way is better than the implied ordering: for example we might expect proxies to be well behaved and place any profile diffs after the client device profile diff. If any intermediate device re-orders profile diffs, this may lead to unexpected behaviour when the profile is resolved.

14. Conclusions

This position paper has described some key issues identified with CC/PP and UAProf while implementing DELI. DELI demonstrates that it is possible to implement these standards in spite of the problems. But we still need to consider these issues described here to ensure these standards are widely adopted and to provide real and compelling benefits to users. Many of these issues are important if we wish to provide device independent content. Therefore it is

the author's hope that the relevant working groups will address some of these issues in future versions of these standards.

---

[1] HPL-2001-83 Current Technologies for Device Independence, http://www.hpl.hp.com/techreports/2001/HPL-2001-83.html
[2] Some comments on "Current Technologies for Device Independence", http://www-uk.hpl.hp.com/people/marbut/currTechExtra.htm
[3] HPL-2001-190 Implementing Content Negotiation using CC/PP and WAP UAProf, http://www.hpl.hp.com/techreports/2001/HPL-2001-190.html
[4] DELI: A Delivery context library for CC/PP and UAProf, http://www-uk.hpl.hp.com/people/marbut/DeliUserGuideWEB.htm
[5] Composite Capabilities / Preferences Profile, http://www.w3.org/Mobile/CCPP/
[6] Wireless Application Forum, http://www.wapforum.org/
[7] Resource Description Framework, http://www.w3.org/RDF/
[8] RDF Model and Syntax Specification, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
[9] RDF Schema Specification 1.0, http://www.w3.org/TR/2000/CR-rdf-schema-20000327/
[10] W3C RDF Validation service, http://www.w3.org/RDF/Validator/
[11] CC/PP: Structure and Vocabularies, http://www.w3.org/TR/CCPP-struct-vocab/
[12] CC/PP: Requirements and Architecture, http://www.w3.org/TR/2000/WD-CCPP-ra-20000721/
[13] CC/PP: Terminology and Abbreviations, http://www.w3.org/TR/2000/WD-CCPP-ta-20000721/
[14] DELI, http://www-uk.hpl.hp.com/people/marbut/deli/
[15] WAP Forum UAProf mailing list, http://mail.wapforum.org/wap-uaprof.html
[16] Jena, http://www.hpl.hp.com/semweb/jena-top.html
[17] Co-Parsing of RDF and XML, http://www-uk.hpl.hp.com/people/jjc/docs/r292.pdf
[18] RDF-Filter, http://rdf-filter.sourceforge.net/
[19] ARP, http://www.hpl.hp.com/semweb/arp.html
[20] Apache Xerces, http://xml.apache.org/xerces2-j/
[21] UAProf Schema, http://www.wapforum.org/profiles/UAPROF/ccppschema-20010330
[22] CC/PP Implementors Guide: Harmonization with exisiting vocabularies and content transformation, http://w ww.w3.org/TR/CCPP-COORDINATION/
[23] Apache Cocoon, http://xml.apache.org/cocoon/
[24] RFC 2506: Media Feature Tag Registration Procedure, ftp://ftp.isi.edu/in-notes/rfc2506.txt
[25] RFC 2533: A Syntax for Describing Media Feature Sets, ftp://ftp.isi.edu/in-notes/rfc2533.txt
[26] RFC 2534: Media Features for Display, Print, and Fax, ftp://ftp.isi.edu/in-notes/rfc2534.txt
[27] RFC 2531: Content Feature Schema for Internet Fax, ftp://ftp.isi.edu/in-notes/rfc2531.txt
[28] A revised media feature set matching algorithm, http://www.ics.uci.edu/pub/ietf/http/draft-klyne-conneg-feature-match-00.txt
[29] MIME content types in media feature expressions, http://community.roxen.com/developers/idocs/drafts/draft-ietf-conneg-feature-type-03.html
[30] Indicating media features for MIME content, http://community.roxen.com/developers/idocs/drafts/draft-ietf-conneg-content-features-03.html
[31] RFC 2616: HTTP 1.1 Content Negotiation, page 70-73, ftp://ftp.isi.edu/in-notes/rfc2616.txt
[32] CC/PP exchange protocol using HTTP Extension Framework, http://www.w3.org/TR/NOTE-CCPPexchange
[33] Content Negotiation Header in HTTP Scenarios, http://search.ietf.org/internet-drafts/drafts-hjelm-http-cnhttp-scenarios-00.txt
[34] W3C Webservices activity, http://www.w3.org/2002/ws/
[35] Jigsaw Server, http://www.w3.org/Jigsaw/