# User Agent Profile

Approved Version 2.0 – 06 Feb 2006

## Open Mobile Alliance

OMA-TS-UAProf-V2_0-20060206-A

# Contents

# Figures

# 1. Scope

The *User Agent Profile* specification is concerned with capturing classes of device capabilities and preference information. These classes include (but are not restricted to) the hardware and software characteristics of the device as well as information about the network to which the device is connected. The user agent profile contains information used for *content formatting purposes.* A user agent profile is distinct from a *user preference profile* that would contain application-specific information about the user for content *selection* purposes. For example, a user preference profile might designate whether the user is interested in receiving sports scores and, if so, the particular teams. The specification of user preference profiles is beyond the scope of this document.

In accordance with Composite Capability/Preference Profiles (CC/PP) [CCPP], the user agent profile schema is defined using an RDF schema and vocabulary [RDF-Schema]. This document defines the structure of this schema in terms of the class definitions and semantics of attributes for devices adhering to the WAP standard. Extensibility guidelines are also offered to assist in extending the schema with new components.

Regarding the user agent profile, this document assumes that:

- The information contained within the profile is provided on behalf of the user who will be receiving the content contained in the associated WSP or HTTP response. Profiles may also be used by the Push Proxy Gateway (PPG).
- The CC/PP repository storing the profile is secure, meaning that it does not permit unauthorized modification to stored profile information. The functionality associated with this repository may be implemented in a WAP gateway or intermediate proxy or as a separate standalone element in the network.
- WSP/HTTP headers are generally not encrypted. Because this specification does not define any security mechanisms, care must be taken when including user-confidential information[1] in the profile unless the profile is transmitted over an end-to-end secure channel (e.g. TLS/WTLS to the origin server).
- An implicit chain of trust exists between the client and origin server. The integrity of the profile is maintained (in other words, not compromised) as it is transmitted through or cached within the network. It is assumed that the network elements that contribute property descriptions to the profile are trusted. Network elements will not assemble a "history" about users by tracking the deltas in their profile over time.
- To ensure the integrity of the profile, lower-level mechanisms, such as TLS or WTLS, must be used.

Use of the profile by the origin server is intended to optomize the content to the characteristics of the device, and the users' preferences. If the profile were discarded, corrupted, or otherwise inaccessible, the origin server should still provide content to the client.

It should be noted that user agent profiles are not statements of conformance but rather, a content negotiation tool to be used to adapt content between a client and content provider.

[1] The definition of "*confidential*" varies among different users, cultures, and service providers. It is therefore impossible to define a profile schema that meets the security needs of all users while still conveying any useful information.

## 1.1 Relationship to Other Standards

This specification builds upon and coexists with numerous WAP and Internet standards. These relationships are summarized below:

- Composite Capability/Preferences Profiles (CC/PP) [CCPP]: This specification defines the Capabilities and Preference Information (CPI) structure according to the structure mandated by the CC/PP specification.
- Resource Description Framework (RDF) [RDF], [RDF-Schema]: The CC/PP specification uses the RDF syntax to represent the CPI. In designing or extending the schema, a schema designer must be familiar with the RDF concepts. MIME type for RDF is application/rdf+xml as specified in [RFC3023].
- Wireless Session Protocol (WSP) [WSP]: One of the mechanisms used to transport CPI over wireless networks is within WSP headers
- CC/PP Exchange Protocol over HTTP [CCPPex]: Previous versions of UAProf [UAProf1] allowed for the transmission of CPI using the CC/PP Exchange Protocol over HTTP which, in turn, defines headers in HTTP 1.1 [HTTP] with the HTTP Extension Framework [HTTPext].
- HTTP [HTTP]: Optionally, the CPI can be transmitted over Wireless Profiled HTTP [W-HTTP] from the mobile terminal to the origin server. In this case, proprietary headers specified in this specification is used to convey the information.
- WAP Push Access Protocol (PAP) [WAP-PAP]: This protocol is used by push origin servers to retrieve the CPI from the Push Proxy Gateway (PPG).

# 2. References

## 2.1 Normative References


| | |
|---|---|
| CCPP | "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies", G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, URL: http://www.w3.org/TR/CCPP-struct-vocab |
| CCPPex | "CC/PP exchange protocol based on HTTP Extension Framework", H. Ohto, J. Hjelm, June 1999, URL: http://www.w3.org/TR/NOTE-CCPPexchange |
| CREQ | "Specification of WAP Conformance Requirements". WAP Forum™. WAP-221-CREQ-20000915-a, URL: http//www.openmobilealliance.org/ |
| HTTP | "Hypertext Transfer Protocol HTTP/1.1", RFC2616, R. Fielding, et al., June 1999, URL: ftp://ftp.isi.edu/in-notes/rfc2616.txt |
| HTTPext | "An HTTP Extension Framework", RFC 2774, H. Nielsen, P. Leach, and S. Lawrence, February 2000, URL: ftp://ftp.isi.edu/in-notes/rfc2774.txt |
| RDF | "RDF Semantics", World Wide Web Consortium, P. Hayes, URI: http://www.w3.org/TR/rdf-mt |
| RDF-Schema | "RDF Vocabulary Description Language 1.0: RDF Schema", World Wide Web Consortium, D.Brickley, R.V. Guha, URL: http://www.w3.org/TR/rdf-schema |
| RDF-Syntax | "RDF/XML Syntax Specification (Revised)", World Wide Web Consortium, D.Beckett, URL: http://www.w3.org/TR/rdf-syntax-grammar |
| RFC1321 | "The MD5 Message-Digest Algorithm",RFC 1321, R. Rivest, April 1992, URL: ftp://ftp.isi.edu/in-notes/rfc1321.txt |
| RFC3066 | "Tags for the Identification of Languages", RFC 3066, H. Alvestrand, January 2001, URL: ftp://ftp.isi.edu/in-notes/rfc3066.txt |
| RFC2119 | "Key words for use in RFCs to Indicate Requirement Levels", RFC2119, S. Bradner, March 1997, URL: http://www.ietf.org/rfc/rfc2119.txt |
| RFC2045 | "Multipurpose Internet Mail Extensions (MIME) Part One", RFC2045, N. Freed et al., November 1996, URL: ftp://ftp.isi.edu/in-notes/rfc2045.txt |
| RFC2234 | "Augmented BNF for Syntax Specifications: ABNF", RFC2234, D. Crocker, Ed., P. Overell. November 1997, URL: http://www.ietf.org/rfc/rfc2234.txt |
| RFC2396 | "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, T. Berners-Lee, R. Fielding, and L. Masinter, August 1998, URL: ftp://ftp.isi.edu/in-notes/rfc2396.txt |
| WAE | "Wireless Application Environment Specification", WAP Forum™, WAP-236-WAESpec, URL: http://www.openmobilealliance.org/ |
| WDP | "Wireless Datagram Protocol", WAP Forum™, WAP-259-WDP, URL: http://www.openmobilealliance.org/ |
| W-HTTP | "WAP Wireless Profiled HTTP", WAP Forum™. WAP-229-HTTP, URL: http://www.openmobilealliance.org/ |
| WSP | "Wireless Session Protocol Specification", WAP Forum™, WAP-230-WSP, URL: http://www.openmobilealliance.org/ |
| WTA | "Wireless Telephony Application Specification", WAP Forum™, WAP-169-WTA, URL: http://www.openmobilealliance.org/ |
| XML | "Extensible Markup Language (XML) 1.0 (Second Edition)", World Wide Web Consortium Recommendation, Tim Bray et al., 6 October 2000, URL: http://www.w3.org/TR/2000/REC-xml-20001006 |
| XML-NS | "Namespaces in XML", W3C Recommendation, T. Bray, D. Hollander, A. Layman, January 1999, URL: http://www.w3.org/TR/1999/REC-xml-names |
| XML-Schema | "XML Schema Part 1: Structures", World Wide Web Consortium, Recommendation, H. Thompson, et.al., 2 May, 2001 |

## 2.2 Informative References


BLT "Specification of the Bluetooth System, profiles version 1.1", Bluetooth SIG, Inc., February 2001. URL: http://www.bluetooth.com/developer/specification/Bluetooth_11_Profiles_Book.pdf

DLOTA "Generic Content Download Over the Air Specification", Open Mobile Alliance™, OMA-Download-OTA-v1_0. http://www.openmobilealliance.org/

DRM "Digital Rights Management", Open Mobile Alliance™, OMA-Download-DRM-v1_0. http://www.openmobilealliance.org/

DRMREL "Rights Expression Language", Open Mobile Alliance™, OMA-Download-DRMREL-v1_0. http://www.openmobilealliance.org/

EMN "E-Mail Notification", Open Mobile Alliance™, OMA-Push-EMN-v1_0, URL: http://www.openmobilealliance.org/

IANA "Internet Assigned Numbers Authority", URL: http://www.iana.org

JCP "Java Community Process", URL: http://www.jcp.org/

MexE "3GPP TS 23.057: Mobile Execution Environment (MexE); Stage 2", URL: http://www.3gpp.org

PICT "Pictogram Specification", The WAP Forum™, WAP-213-Pictogram, URL: http://www.openmobilealliance.org/

P3P "The Platform for Privacy Preferences 1.0 (P3P1.0) Specification", World Wide Web Consortium Recommendation, L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, J. Reagle, URL: http://www.w3.org/TR/P3P

RDF-Val "RDF Validation Service", World Wide Web Consortium, URL: http://www.w3.org/RDF/Validator/

RFC2703 "Protocol-independent Content Negotiation Framework", RFC2703, G. Klyne, September 1999, URL: ftp://ftp.isi.edu/in-notes/rfc2703.txt

RFC3023 "XML Media Types", RFC3023,. M. Murata, S. St.Laurent, D. Kohn, January 2001, URL: ftp://ftp.isi.edu/in-notes/rfc3023.txt

UA-Attrs "Client-specific Web Services by using User Agent Attributes", Tomihisa Kamada, et. Al, December 1997, URL: http://www.w3.org/TR/NOTE-agent-attributes

UAProf1 "User Agent Profile Specification", WAP Forum™, WAP-174-UAProf., URL: http://www.openmobilealliance.org/

WAPARCH "WAP Architecture Specification", WAP Forum™, WAP-210-WAPArch, URL: http://www.openmobilealliance.org/

WAP-PAP "Push Access Protocol Specification", WAP Forum™, WAP-247-PAP, URL: http://www.openmobilealliance.org/

WAP-PushArch "Push Architectural Overview", WAP Forum™, WAP-250-PushArch, URL: http://www.openmobilealliance.org/

WAP-PushOta "Push OTA Protocol Specification", WAP Forum™, WAP-235-PushOTA, URL: http://www.openmobilealliance.org/

WINA "WAP Interim Naming Authority", WAP Forum™, URL: http://www.openmobilealliance.org/wina

WTAI "Wireless Telephony Application Interface Specification", WAP Forum™, WAP-170-WTAI, URL: http://www.openmobilealliance.org/

WMLStdLib "WMLScript Standard Libraries Specification", WAP Forum™. WAP-194-WMLSL, URL: http://www.openmobilealliance.org/

XHTML "XHTML™ 1.0: The Extensible HyperText Markup Language – A Reformulation of HTML 4 in XML 1.0", Steve Pemberton et al, 26 January 2000, URL: http://www.w3.org/TR/xhtml1/

XMOD "Modularization of XHTML", World Wide Web Consortium Recommendation, Murray Altheim et al, 10 April 2001, URL: http://www.w3.org/TR/xhtml-modularization/

# 3. Terminology and Conventions

## 3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction” are normative, unless they are explicitly indicated to be informative.

## 3.2 Definitions

| | |
|---|---|
| **Attribute** | A UAProf attribute refers to the data elements describing the CPI and is represented as an RDF property element. Each UAProf attribute is associated with a single value or a list of values or resources. |
| **CC/PP Repository** | A server that stores CPI persistently in a form that may be referenced by and incorporated into a profile. A CC/PP repository is typically a Web server that provides CC/PP profiles in response to HTTP requests. |
| **Component** | Elements of a high level classification of the information in the CPI. For UAProf, these include *HardwarePlatform*, *SoftwarePlatform*, *NetworkCharacteristics*, *WAP*, *MexE*, *BrowserUA and PushComponent.* |
| **CPI** | Capabilities and Preference Information pertaining to the capabilities of the device, the operating and network environment, and user’s personal preferences for receiving content and/or resource. CPI are represented by means of a UAProf Profile. |
| **Gateway** | Software that is capable of bridging disparate network protocols. For the purposes of this specification, “gateway” refers to protocol bridging functionality, which may exist in a stand-alone gateway or may be co-located with a proxy or origin server. |
| **Home Location Register** | A permanent database used in cellular networks. The HLR is located on the SCP (Signal Control Point) of the cellular provider of record, and is used to identify/verify a subscriber; it also contains subscriber data related to features and services. |
| **Origin Server** | Software that can respond to requests from a WAP terminal by delivering appropriate content or error messages. The origin server may receive requests via either WSP or HTTP. Application programs executing on the origin server can use UAProf to deliver content that is tailored in accordance with the CPI that can be found within the provided Profile. For the purpose of this specification, “origin server” refers to content generation capabilities, which may physically exist in a stand-alone Web server or may be co-located with a proxy or gateway. |
| **Profile** | An instance of the UAProf schema that describes capabilities for a specific device and network configuration. A profile need not have all the attributes identified in the vocabulary/schema. |
| **Profile repository** | The profile repository is an origin server that stores profile resources. Profiles are treated as regular static or dynamic resources served using the access scheme specified as a part of the URI identifying the profile of a particular device. To increase speed, the origin server can use a cache to reduce the number of downloads from profile repositories. It is expected that profile repositories would typically be run by device manufacturers or software companies providing information about their user agents. |
| **Property** | An RDF property represents a specific aspect, characteristic, capability or relation (meta data) used to describe a resource. It is denoted using RDF property element syntax. |
| **Proxy** | Software that receives HTTP requests and forwards that request toward the origin server (possibly by way of an upstream proxy) using HTTP. The proxy receives the response from the origin server and forwards it to the requesting client. In providing its forwarding functions, the proxy may modify either the request or response or provide other value-added functions. For the purposes of this specification, “proxy” refers to request/response forwarding functionality, which may exist in a stand-alone HTTP proxy or may be co-located with a gateway or origin server. |
| **Resource** | An object or element being referred to in a UAProf expression is a resource. A UAProf resource is typically identified by a URI. |
| **Schema** | An RDF schema denotes resources which constitute the particular unchanging versions of an RDF vocabulary at any point in time. It is used to provide semantic information (such as organization and relationship) about the interpretation of the statements in an RDF data model. It does not include the values associated with the attributes. |

| | |
|---|---|
| **Server** | Software which respond to HTTP requests. This software may reside on the mobile terminal. The requests may be for CPI or use CPI as meta data. |
| **User** | An individual or group of individuals acting as a single entity. The user is further qualified as an entity who uses a device to request content and/or resource from a server. |
| **User agent** | A program, such as a browser, running on the device that acts on a user's behalf. Users may use different user agents at different times. |
| **Vocabulary** | A collection of attributes that adequately describe the CPI. A vocabulary is associated with a schema. The vocabulary for UAProf includes attributes pertaining to the device capabilities and network characteristics. |

## 3.3 Abbreviations

| | |
|---|---|
| **CCQ** | Client Capability Query |
| **CC/PP** | Composite Capability/Preferences Profiles |
| **CC/PPex** | CC/PP Exchange Protocol |
| **CC/PP-http** | CC/PP Exchange Protocol over HTTP |
| **CC/PP-WSP** | CC/PP Exchange Protocol over WSP |
| **CPI** | Capability and Preference Information |
| **DRM** | Digital Rights Management |
| **EMN** | E-Mail notification |
| **HTML** | Hyper-Text Markup Language |
| **HTTP** | Hyper-Text Transfer Protocol |
| **IANA** | Internet Assigned Numbers Authority |
| **LWS** | Linear White Space |
| **OTA** | Over the Air i.e. in the radio network |
| **PAP** | Push access protocol |
| **PI** | Push Initiator |
| **ppg** | Push Proxy Gateway |
| **P3P** | Platform for Privacy Preferences Project |
| **RDF** | Resource Description Framework |
| **UAProf** | User Agent Profile |
| **URI** | Uniform Resource Identifier |
| **URL** | Uniform Resource Locator |
| **W3C** | World Wide Web Consortium |
| **WAP** | Wireless Application Protocol |
| **W-HTTP** | Wireless Profiled HTTP |
| **WML** | Wireless Markup Language |
| **WSP** | Wireless Session Protocol |
| **WTA** | Wireless Telephony Application |
| **W-TCP** | Wireless Profiled TCP |
| **WTLS** | Wireless TLS |
| **XHTML** | Extensible HyperText Markup Language |
| **XML** | Extensible Markup Language |

# 4. Introduction

*This section is informative.*

As WAP-enabled devices come of age, an assumption of device homogeneity is no longer valid. In particular, mobile devices can be expected to have an increasingly divergent range of input and output capabilities, network connectivity, and levels of language support. Moreover, users may have content presentation preferences that also cannot be transferred to the server for consideration. As a result of this device heterogeneity and the limited ability of users to convey their content presentation preferences to the server, clients may receive content that they cannot store, that they cannot display, that violates the desires of the user, or that takes too long to convey over the network to the client device.

Work is ongoing in the World-Wide Web Consortium (W3C) to define mechanisms for describing and transmitting information about the capabilities of Web clients and the display preferences of Web users. The Composite Capabilities/Preferences Profile (CC/PP) specification [CCPP] defines a high-level structured framework for describing this information using the Resource Description Framework (RDF) [RDF]. CC/PP profiles are structured as named "components," each containing a collection of attribute-value pairs, or properties. Each component may optionally provide a default description block containing either a set of default values for attributes of that component or a URI that refers to a document containing those default values. Any attributes explicitly provided in the component description therefore override the default values provided in the default description block or through that URI. The CC/PP specification does not mandate a particular set of components or attributes, choosing instead to defer that definition to other standards bodies. The mechanism by which the profile is transported between the mobile terminal, WAP Gateway and origin server is defined in this specification

The User Agent Profile (UAProf) specification extends WAP 2.0 to enable the end-to-end flow of a User Agent Profile (UAProf), also referred to as ***Capability and Preference Information*** (CPI), between the WAP client, the intermediate network points (proxies and gateways), and the origin server. It seeks to interoperate with the emerging standards for Composite Capability/Preference Profile (CC/PP) [CCPP] distribution over the Internet. It uses the CC/PP model to define a robust, extensible framework for describing and transmitting CPI about the client, user, and network that will be processing the content contained in a WSP/HTTP response. The specification defines a set of components and attributes that WAP-enabled devices may convey within the CPI.[2] This CPI may include, but is not limited to:

- hardware characteristics (screen size, color capabilities, image capabilities, manufacturer, etc.),
- software characteristics (operating system vendor and version, support for MexE, list of audio and video encoders, etc.),
- application/user preferences (browser manufacturer and version, markup languages and versions supported, scripting languages supported, etc.),
- WAP characteristics (WML script libraries, WAP version, WML deck size, etc.), and
- network characteristics (bearer characteristics such as latency and reliability, etc.).

As a request travels over the network from the client device to the origin server, each network element may optionally add additional profile information to the transmitted CPI. These additions may provide information available solely to that particular network element. Alternatively, this information may override the capabilities exposed by the client, particularly in cases where that network element is capable of performing in-band content transformations to meet the capability requirements of the requesting client device.

Origin servers, gateways, and proxies can use the CPI to ensure that the user receives content that is particularly tailored for the environment in which it will be presented. Moreover, this specification permits the origin server to select and deliver services that are appropriate to the capabilities of the requesting client. Finally, it is expected that this specification will be used to enhance content personalization based on user preferences, and other factors, as specified by the Platform for Privacy Preferences Project (P3P) [P3P].

[2] Though a set of well-known components and attributes are defined within this specification, individual implementors are free to provide additional components and attributes with their CPI. However, most origin servers or proxies are unlikely to properly interpret those extensions unless they are standardized by another standards body.

## 4.1 UAProf and Web Convergence

UAProf and CC/PP have a common ancestry. However the development and release cycle differences have led to some divergence which has created several issues including: interoperability problems, perception and additional complexity in implementations.

This version of the UAProf specification addresses some of these issues. For example, UAProf's defaults property (formerly defined as the **prf:defaults** property) which is semantically equivalent to CC/PP's defaults property has been replaced with an explicit adoption of CC/PP's defaults mechanism (via CC/PP's **ccpp:defaults** property).

The W3C's RDF language has continued to evolve since the UAProf work began. Although the UAProf specification has recently been updated to include several changes to align its RDF usage with the W3C's specification, it is imperative for UAProf to continue to adopt relevant changes being proposed by the W3C. In particular this version of UAProf adopts advances in the RDF Schema and datatyping language that enable automatic processing of UAProf's schema and increase interoperability.

# 5. End-to-End Architecture

*This section is informative.*

This section provides an overview of the end-to-end architecture.

WAP clients can connect to an origin server using the legacy WAP stack's WSP [WSP] or Wireless Profiled HTTPW-HTTP]. WSP clients may connect to servers using a WAP gateway, or directly to an origin server that provides support for that protocol. W-HTTP [W-HTTP] clients can connect to an origin server directly, or using a feature or performance enhancing proxy.

This specification provides for the end-to-end specification, delivery, and processing of composite capability information from the device. The information is specified on the client device, optionally enhanced with information provided with a particular request, optionally combined with other information available over the network, and made available to the origin server. Over the Internet, this specification assumes the use of the CC/PP [CCPP], HTTP 1.1 [HTTP], and optionally the CC/PP Exchange Protocol over HTTP [CCPPex], and HTTP 1.1 [HTTP] with the HTTP Extension Framework [HTTPext].

A client can connect to the origin server as described in the overall WAP Architecture specification [WAPARCH], the nodes involved in this specification are depicted in Figure 5.1. Note, that the protocol used to retrieve information stored in the profile repository is not specified in this specification.

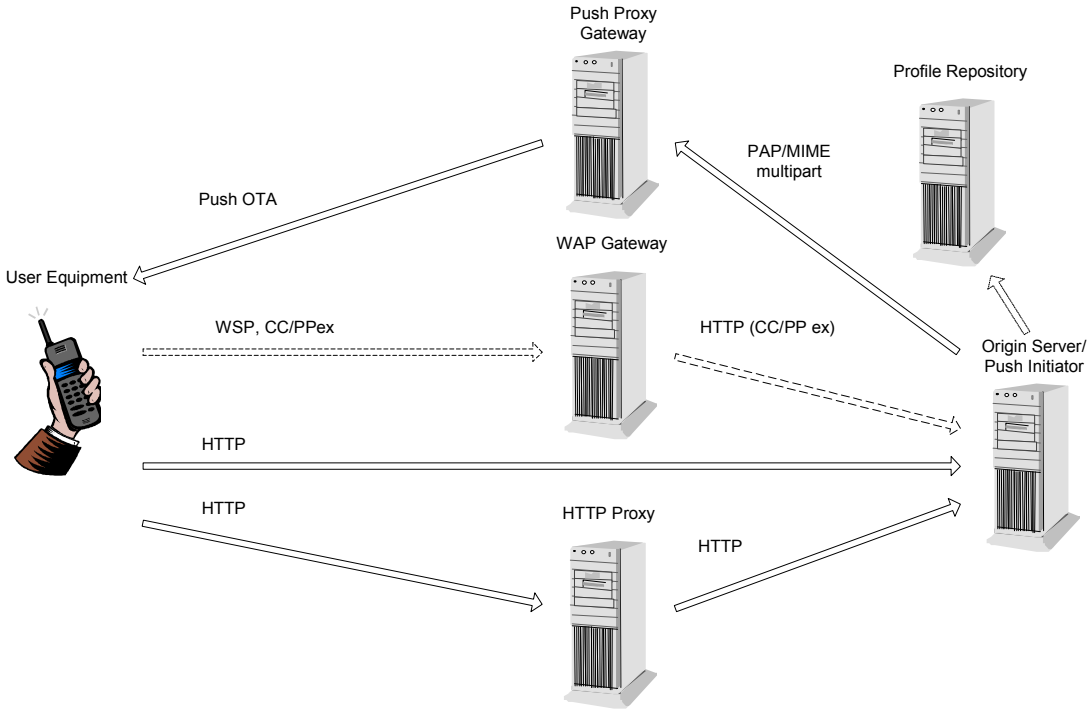


**Figure 5.1:UAProf end-to-end architecture.**

# 6. Usage Scenarios

*This section and its subsections are informative.*

This Section describes several scenarios in which CPI is conveyed and used to support information delivery to a WAP-enabled client. The description is divided into three sections, describing WAP/WSP and HTTP clients, and common usage scenarios.

## 6.1 WAP/WSP Clients

### 6.1.1 Opening a WSP Session and Establishing an Initial UAProf

Upon opening a WSP [WSP] session with a WAP gateway, the UAProf-aware client conveys its profile information using *Profile* and *Profile-Diff* headers within the WSP **Connect** request. Upon receiving the profile, a WAP gateway that is aware of the UAProf capability responds with a *Profile-Warning* header value of 100 ("OK"). This header signals to the client that the CPI is being cached by the WAP gateway and will be effective for the lifetime of the session. The client device may update the CPI at any time during the session lifetime.

If the client does not receive the "OK" *Profile-Warning* header in the WSP **Connect** response, it assumes that the gateway does not support this UAProf specification and therefore that the CPI is not being cached by the WAP gateway. The client device may neither convey nor update the CPI during the session lifetime.

### 6.1.2 Updating the UAProf During an Active WSP Session

While a UAProf-aware session is established, the client may update the active UAProf at any time. To do this, the client transmits a WSP Session **Resume** message to the WAP gateway, said message containing *Profile* and *Profile-Diff* headers with the new CPI. Upon updating the cached headers, the gateway responds with a *Profile-Warning* header value of 100 ("OK"). All future requests issued on the WSP session will be associated with the newly cached *Profile* and *Profile-Diff* headers.

### 6.1.3 Resuming a Suspended WSP Session

To resume a suspended WSP session, the client initiates a standard WSP **Resume** request. The session, once resumed, retains all of the cached header state at the gateway, including the *Profile* and *Profile-Diff* headers containing the CPI.

The lifespan of a WSP session is not tied to power cycles of the device. In many situations, therefore, the client's capabilities may change significantly while a WSP session is suspended. This is the case, for example, if hardware is added or removed from the device while it is turned off. It may also occur if the WSP session is held by a smart card that may be moved to a new device while the session is suspended. In these situations, the client device must update the cached WSP *Profile* and *Profile-Diff* headers upon resuming the session. These updated headers are conveyed in the WSP **Resume** request and cached at the WAP gateway, in the same manner described in Section 6.1.2.

### 6.1.4 Suspending an Active WSP Session

To suspend a WSP session, the client initiates a standard WSP **Suspend** request. As long as the session is established, the WAP gateway must cache all negotiated headers, including the *Profile* and *Profile-Diff* headers associated with the WSP session. However, should the gateway choose to discard the session, it may also discard these cached headers. To resume the session, the client device follows the steps outlined in Section 6.1.3.

### 6.1.5 Issuing a WSP Request for Content

To request content during a UAProf-aware WSP session, the client issues a standard WSP request to the WAP gateway. The WAP gateway is responsible for forwarding this WSP request to the designated origin server (typically via HTTP). In forwarding the request to the origin server, the WAP gateway must include the CPI associated with the WSP session over which the request was conveyed. The origin server receives the HTTP request (which may have been modified by one or more intermediate HTTP proxies), resolves the CPI, and generates a response along with a *Profile-Warning* header indicating whether the CPI was honored as the content was generated.

The HTTP response may itself be modified by intermediate HTTP proxies to better meet the needs of the requesting client. The WAP gateway forwards the content to the client device over WSP, encoding the *Profile-Warning* header for efficient transmission over the wireless network.

#### 6.1.5.1 CPI Provided by the WAP Gateway

As it forwards the request, the WAP gateway may optionally add profile information, reflecting information only available to the gateway. For instance, if a network operator controls the gateway, then the gateway may provide additional network information (such as from a Home Location Register (HLR)) that is not otherwise available to the requesting client. Similarly, the WAP gateway may add information to the profile to override information provided by the requesting device/user. These overrides may, for example, reflect policies in place by the network or gateway operator.

#### 6.1.5.2 Overriding the CPI Within a Single Request

When issuing a WSP request, the client may provide additional information to override or augment the basic CPI already cached at the WAP gateway. This additional information is only applied during the scope of the associated request, and it is not used by the gateway during subsequent requests.

To augment the profile during a request, the client includes *Profile* and/or *Profile-Diff* headers with the WSP request. As it generates an HTTP request, the WAP gateway overrides the cached WSP *Profile* and *Profile-Diff* headers with any headers provided in the WSP request. As described above in Section 6.1.5.1, the gateway may also add additional information to the forwarded profile.

## 6.2 Wireless Profiled HTTP Clients

The UAProf capable handset may transmit the CPI data in the *x-wap-profile* and *x-wap-profile-diff* headers with each HTTP request that is made. The user agent should also be capable of processing the *x-wap-profile-warning* header, which indicates whether the CPI was used when formulating the response.

## 6.3 Push environment

The WAP Push environment [WAP-PushArch, WAP-PAP] uses UAProf to allow the *push-initiator* (e.g. the application generating push messages) to adapt the content of push-messages it generates based on the target device's capabilities and preferences information (CPI). Furthermore, the push-initiator (PI) can supply a constraining UAProf setting which is matched by the Push Proxy Gateway (PPG) against the target device's CPI in determining necessary and possible content transformations respectively discarding the push-message request altogether.

### 6.3.1 Client Capability Query

In order to inform the PI on the target device's CPI, the PI can request these from the PPG using a client capability query (CCQ) [WAP-PAP]. Based on the target device's PAP target address, PPG retrieves the corresponding, currently valid UAProf formatted CPI and includes this capability entity as part of the CCQ-response as described in [WAP-PAP].

The PI, being aware of the current CPI of a target device can use application specific logic to tailor the content of subsequent push-messages it generates.

### 6.3.2 Profile Matching

The PI can append a capability entity to an outgoing push-message that is formatted as a MIME multi-part related message.

Upon reception of the push-message's capability entity, PPG will match this profile requirement against the currently valid CPI of the target device. (If multiple target devices are addressed, the matching process is executed in turn for each individual target address.) If the required profile matches the target device profile settings, the message is forwarded. In order for such matching to occur, PPG may apply appropriate transformations to the push-message's content. However, such transformations are at the discretion of a particular PPG implementation, there is no means for the PI to control the content transformations applied at the PPG. If no match can be achieved with the relevant client capabilities, the push-message is rejected using the appropriate WAP Push status reporting [WAP-PAP].

As one particular example, consider the PI requiring the target device to support audio output. It thus will include the UAProf SoundOutputCapable hardware platform component (see Appendix A) attribute requiring it to be "Yes". If the PPG determines that the target device(s) does currently not support sound output, it will reject the push-message.

Matching of required properties against the currently valid CPI can only be done at the PPG once all possible or necessary content transformations have been applied. As an example, consider the WapPushMsgSize attribute: although the PI may indicate a value for this attribute, it is only indicative since the PI does not have knowledge of the push-message size after header compression and WML tokenization have taken place (assuming WSP is used OTA). Furthermore, PPG must validate the push-message size against the currently valid CPI and not the possibly outdated values precedently queried by the PI.

### 6.3.3 Push OTA

The push OTA protocol [WAP-PushOta] is designed to run on top of either WSP or HTTP. The two protocol variants are referred to as "OTA-WSP" and "OTA-HTTP" respectively. When OTA-WSP is used the profile and profile-diff headers are conveyed from the terminal to the Push Proxy Gateway (PPG) within the connect request during push session establishment. If the PPG supports UAProf a profile-warning header with status code 100 will be included in the response to indicate that the profile and profile-diff headers were correctly understood. Hence, the profile and profile-diff headers are used as request headers in this case.

On the other hand, when OTA-HTTP is utilised the PPG instead sends a registration request to the terminal to obtain its capability and preference information. This is accomplished using the HTTP OPTIONS method; the PPG sends an OPTIONS request to the terminal whereupon the terminal includes its CPI headers in the response. The CPI headers defined in [WAP-PushOta] reflects the most common capability information. The profile and profile-diff headers may also be included among the CPI headers to provide more extensive information. The headers may even replace the defined CPI headers if the PPG indicates that it supports UAProf by including a profile-warning header with warning code 200 in the request. Consequently, the profile and profile-diff headers (see Section 8) are used as response headers when OTA-HTTP is used.

## 6.4 Profile resolution

### 6.4.1 Resolving Attribute Values in the CPI

An origin server or proxy that needs to determine the correct values for CPI attributes must resolve the profile. This resolution process applies a collection of default attribute values and then applies appropriate overrides to those defaults. Because different network elements may provide additional (or overriding) profile information, the resolution process must apply this additional information to determine the final attribute values.

The User Agent Profile is constructed in three stages:

- Resolve all indirect references by retrieving URI references contained within the profile
- Resolve each *Profile* and *Profile-Diff* document by first applying attribute values contained in the default URI references and by second applying overriding attribute values contained within the category blocks of that *Profile* or *Profile-Diff*.
- Determine the final value of the attributes by applying the resolved attribute values from each *Profile* and *Profile-Diff* in order, with the attribute values determined by the resolution rules provided in the schema. Where no resolution rules are provided for a particular attribute in the schema, values provided in profile diffs are assumed to override values provided in previous profiles or profile diffs.

# 7. Composite Profile Segments and Attributes

*This section is normative, all examples are informative. Throughout this section, sentences in Italics are also informative.*

The CC/PP framework [CCPP] enables information about the capabilities and characteristics of a device and network to be communicated to web servers and gateways/proxies so that suitable content is rendered to the device. These capabilities and characteristics are referred to as attributes, and together form a vocabulary. The syntax for attributes values and, to some extent, the semantics are identified in a schema for that vocabulary. A profile is an instance of the schema and contains one or more attributes from the vocabulary. The attributes in the schema are classified into one of several components, each of which represents a distinguished set of characteristics. Components are typed and may contain a default attribute that refers to the default setting of the various component attributes. The default values are overridden by explicitly including a CC/PP attribute in a profile.

CC/PP is based on the Resource Description Framework (RDF) [RDF]. RDF is an XML application (see [RDF-Syntax]) and specifies a context free grammar that well-formed RDF documents must respect. Note that not all RDF compliant documents can be described using a Document Type Definition (DTD); certain RDF features cannot be captured through a DTD. Furthermore, RDF allows various notations to convey identical meaning. For instance an XML element or an XML element attribute can be used to represent an RDF property. This specification restricts the usage of RDF in order to ease the manipulation of RDF compliant UAProf documents.

## 7.1 Schema Layout

The definition of a UAProf schema is governed by the following rules:

- The schema MUST correspond to a vocabulary which MUST be specified in conformity with the RDF [RDF-Syntax] standard and for which the unique XML namespace [XML-NS] name “prf”, prefixed with the XML namespace prefix “xmlns:”, MUST serve as the unambiguous identifier.
- Other reserved UAProf schema XML namespace prefixes are “rdf” and “rdfs” which MUST be used to identify the RDF and the RDF Schema namespace prefixes, respectively.
- All XML namespace declarations used within the UAProf schema definition MUST be declared as attributes of the **rdf:RDF** root element. Namespace aliases MUST NOT be introduced within nested XML elements of the UAProf schema.
- The UAProf schema MUST consist of one or more CC/PP components, each describing a set of properties (or attributes) within one or more RDF description elements.
- Each component MUST have a **rdf:type** property with a **rdf:resource** attribute whose values is set to the absolute URI for the RDFS Class object and an **rdfs:label** property whose value is unique across the set of components introduced by the UAProf schema.
- Attribute values are typed (see Appendix A for more information about UAProf’s datatypes) and MUST obey their corresponding syntax. Attributes of type Literal MUST be interpreted in a case-sensitive manner. Leading and trailing white space MUST be discarded. Embedded linear white space is processed according to LWS rules of HTTP/1.1 [HTTP].
- Descriptions to override the default values MAY be included in the schema’s component descriptions. The final value of an instance of a profile’s attribute is resolved based on the resolution rule associated with the attribute.

## 7.2 User Agent Profile Components

The schema for WAP User Agent Profiles consists of description blocks for the following key components:

**HardwarePlatform:** A collection of properties that adequately describe the hardware characteristics of the terminal device. This includes, the type of device, model number, display size, input and output methods, etc.

**SoftwarePlatform:** A collection of attributes associated with the operating environment of the device. Attributes provide information on the operating system software, video and audio encoders supported by the device, and user’s preference on language.

**BrowserUA:** A set of attributes to describe the HTML browser application

**NetworkCharacteristics:** Information about the network-related infrastructure and environment such as bearer information. *These attributes can influence the resulting content, due to the variation in capabilities and characteristics of various network infrastructures in terms of bandwidth and device accessibility.*

**WapCharacteristics:** A set of attributes pertaining to WAP capabilities supported on the device. This includes details on the capabilities and characteristics related to the WML Browser, WTA [WTA], etc.

**PushCharacteristics:** A set of attributes pertaining to Push specific capabilities supported by the device. This includes details on supported MIME types, the maximum size of a push-message shipped to the device, the number of possibly buffered push-messages on the device, etc.

Additional components can be added to the schema to describe capabilities pertaining to other user agents such as an Email application or hardware extensions. See Section 7.8 for details.

# 7.3 Attributes

A profile attribute MUST belong to one and only one component of the schema. If an origin server application is interested in a set of attributes that spans multiple components, it MUST parse the complete profile to obtain the necessary information.

Profile attributes MUST be defined using a traditional name and value pair syntax. The first half of the name-value pair describes the attribute, and the other half provides the value itself. The RDF property descriptions MUST be unique and unambiguous in both semantics and value.

Within an RDF component description block, the attribute description MUST be either embedded lexically inline to denote the value or expressed as an RDF resource (using indirect or remote reference such as URIs) that must be resolved to obtain the full description.

UAProf is an application of CC/PP and therefore it inherits the syntax and semantics of CC/PP's "defaults" mechanism as defined in [CCPP]. An instance of a **ccpp:defaults** property may point to a collection of externally-defined attributes and should therefore be described as a URI resource (using the **rdf:resource** attribute). An instance of a **ccpp:defaults** property may also contain inline attributes as defined in [CCPP].

Attributes with composite or multiple values MUST be described as RDF resources. A RDF container (Bag or Sequence) MUST be used to describe a list of values (ordered or unordered) associated with a given attribute. This specification constrains CC/PP to the extent that RDF containers MUST NOT contain elements that are RDF containers themselves. For example, the *InputCharSet* attribute would be expressed as follows:

```
<prf:InputCharSet>
   <rdf:Bag>
      <rdf:li>US-ASCII</rdf:li>
      <rdf:li>Shift_JIS</rdf:li>
   </rdf:Bag>
</prf:InputCharSet>
```

The server parsing and interpreting the profile MAY carry out validation of the attribute descriptions in terms of units and value.

The value of an attribute with multiple descriptions MUST be resolved as follows:

- The description of the attribute within the default description block MUST be resolved first
- Any other description of the attribute identified in a subsequent instantiation of the attribute MUST override the default description
- Where multiple descriptions of the attribute exist outside the default description block, the ultimate value of the attribute MUST be determined by the resolution rules for that attribute. An attribute MUST be associated with one of the following three resolution rules:

1. **Locked:** The final value is determined by the first occurrence of the attribute outside the default description block. Subsequent occurrences MUST be ignored.

2. **Override:** The final value equals the last occurrence of the attribute with a description element.

3. **Append:** The final value is a list of all the occurrences of the attribute

   For example, the *ModelNumber* attribute has a resolution semantic identified as *Locked*. This implies that the first non-default occurrence of the attribute is the final value or that subsequent instantiations of the attribute occurrence cannot change the value of that attribute. On the other hand, an attribute such as *ColorCapable* has a resolution semantic of *Override* which means that of the multiple occurrences of the attribute, only the last determines the final value. Finally, an attribute with a resolution semantic *Append* has a final value that includes all the occurrences for that attribute in the profile.

An attribute (RDF property) MUST closely represent the semantics of the capability or characteristics being represented and MUST follow the naming conventions identified in [RDF] and [RDF-Schema]. For example, the *SoundOutputCapable* attribute indicates whether or not the device supports sound output.

## 7.4 Profile

A profile is an instance of the schema. The following, RDF and CC/PP compliant, syntax rules govern the formation of a profile:

- The root element of the profile, labeled **rdf:RDF**, MUST have one and only one child element and that child element MUST be an **rdf:Description** element. Furthermore, this **rdf:Description** element MUST be identified with an invariant **rdf:ID** attribute, set to e.g. "MyProfile".
- Profiles MUST use the XML namespace prefix "*rdf*" to refer to the RDF namespace and "*prf*" for the UAProf schema namespace. These namespace declarations MUST be included as XML element attributes with the **rdf:RDF** root element and thus apply to the entire scope of the UAProf profile.
- Aliases for the RDF and UAProf schema namespace prefixes MUST NOT be used.
- All profile properties (or attributes) MUST be represented as RDF property elements and MUST NOT be represented using the abbreviated syntax as defined in [RDF-Syntax]. The components in the profile are instances of the component classes identified in the UAProf schema. They MUST be identified as such by means of the **rdf:type** property of which the *resource* attribute value matches the name of the component type in the UAProf schema. For example, *MyNetworkChar* is an instantiated component of type *NetworkCharacteristics*. The profile must therefore include an RDF description such as

```
  <rdf:Description rdf:ID="MyNetworkChar">
    <rdf:type
rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-
20021212#NetworkCharacteristics"/>
```

- Profile parsers MUST parse the profile based on the type of the component and not based on the component's name (since the name can vary, but the type is defined in the UAProf schema.) This applies to the merging of the various delta profiles (e.g. *Profile-diff*) that are generated or appended by different network elements.
- Each component in the profile MUST contain one or more attributes identified in the base vocabulary. It is not required for a profile to contain all the attributes in the base vocabulary. In other words, the profile instantiates a subset of the attributes in the vocabulary.
- For a given component type there MUST NOT be more than one component instance. Components MAY be fragmented within the same profile, that is the attributes of the same component instance MAY be spread across several **rdf:Description** elements. All component fragments that combine into the component instance of a particular type MUST have the same name (**rdf:ID** attribute). The **rdf:type** attribute MUST be specified with each individual fragment and MUST have the same value.
- In a fragmented component only the first fragment MAY contain a defaults reference (i.e. **ccpp:defaults** property); all subsequent fragments, if any of the same type, MUST NOT contain a defaults reference.
- A defaults URL MUST refer to a UAProf profile which describes exactly one UAProf component. The component's **rdf:type** attribute MUST be specified within every default profile. Both, the referring component and the referred default component MUST have the same **rdf:type** attribute value. Defaults MUST NOT be nested, that is the referred default document itself MUST NOT contain a **ccpp:defaults** property.

## 7.5 Profile Merging

This section describes the merging operation of two profiles A and B. Note that the merging operation is not commutative, that is, the result of the merging of A onto B may differ from the result of merging B onto A.

The order of the merging is also important if more than two profiles are to be merged. In this case, the merging operation must be applied repeatedly by merging the 2nd profile onto the 1st profile, merging the 3rd profile onto the result of the previous step, and so forth until all profiles have been merged. Note that the relative order of the profiles to be merged MUST NOT be changed. The above merging process is equivalent to merging the $n^{th}$ profile onto the $(n-1)^{th}$ profile, merging the result of the previous step onto the $(n-2)^{th}$ profile, and so forth until all profiles have been merged.

Two profiles A and B are merged as follows:

1. A and B are normalized. A profile is in its normal form if no fragmented component instances exist and the default references are resolved in all components. If the component contains a **ccpp:defaults** element, the default property values MUST be loaded first. If the component is fragmented, the component attributes are compiled from all fragments in the order of their occurrence observing the resolution policies. These attribute values then always override eventual default values. A normalized UAProf profile therefore consists of at most six component instances without **ccpp:defaults** elements. Note that this procedure in principle would allow handling nested defaults. However, section 7.4 explicitly prohibits nesting of defaults to avoid unnecessary complexity.
2. All corresponding components are merged. Two components $C_A$ and $C_B$ of two profiles A and B correspond if they are instances of the same component type as indicated by the **rdf:type** attributes. Property values that are only specified in the component $C_A$ (but not in $C_B$) are inserted into the result component as is. Property values that are only specified in the component of $C_B$ (but not in $C_A$) are inserted into the result component as is. Property values that are specified in both components are merged according to the resolution policy associated with the particular property:

- If no resolution policy is specified or the resolution policy is **Override**, the property as specified in component $C_B$ is inserted into the result component.
- If the resolution policy is **Locked**, the property as specified in component $C_A$ is inserted into the result component.
- If the resolution policy is **Append**, the property value list specified in component $C_B$ is appended to the property value list specified in component $C_A$, and the property with the value of the concatenated list is inserted into the result component.

3. All components of A without a corresponding component in B, are appended to the result profile as is. All components of B without a corresponding component in A, are appended to the result profile as is.

## 7.6 User Agent Profile Schema and Base Vocabulary

The UAProf schema is discussed in Appendix A.

## 7.7 Profile Example in RDF

This section is intended to help the reader better understand the implementation and use of the specified schema and vocabulary for User Agent Profiles. The reader is expected to be familiar with RDF specifications [RDF], [RDF-Schema].

### 7.7.1 Profile Example 1

The following profile contains attributes from a single vocabulary.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
  <!ENTITY ns-rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!ENTITY ns-prf  'http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-
YYYYMMDD#'>
  <!ENTITY prf-dt  'http://www.openmobilealliance.org/tech/profiles/UAPROF/xmlschema-
YYYYMMDD#'>
]>
```

```
<rdf:RDF xmlns:rdf="&ns-rdf;"
         xmlns:prf="&ns-prf;">

  <rdf:Description rdf:ID="MyDeviceProfile">
    <prf:component>
      <rdf:Description rdf:ID="HardwarePlatform">
        <rdf:type rdf:resource="&ns-prf;HardwarePlatform"/>
        <prf:ScreenSizeChar rdf:datatype="&prf-dt;Dimension">15x6</prf:ScreenSizeChar>
        <prf:BitsPerPixel rdf:datatype="&prf-dt;Number">2</prf:BitsPerPixel>
        <prf:ColorCapable rdf:datatype="&prf-dt;Boolean">No</prf:ColorCapable>
        <prf:TextInputCapable rdf:datatype="&prf-dt;Boolean">Yes</prf:TextInputCapable>
        <prf:ImageCapable rdf:datatype="&prf-dt;Boolean">Yes</prf:ImageCapable>
        <prf:Keyboard rdf:datatype="&prf-dt;Literal">PhoneKeypad</prf:Keyboard>
        <prf:NumberOfSoftKeys rdf:datatype="&prf-dt;Number">0</prf:NumberOfSoftKeys>
      </rdf:Description>
    </prf:component>

    <prf:component>
      <rdf:Description rdf:ID="SoftwarePlatform">
        <rdf:type rdf:resource="&ns-prf;SoftwarePlatform"/>
        <prf:AcceptDownloadableSoftware rdf:datatype="&prf-
dt;Boolean">No</prf:AcceptDownloadableSoftware>
        <prf:CcppAccept-Charset>
          <rdf:Bag>
            <rdf:li rdf:datatype="&prf-dt;Literal">US-ASCII</rdf:li>
            <rdf:li rdf:datatype="&prf-dt;Literal">ISO-8859-1</rdf:li>
            <rdf:li rdf:datatype="&prf-dt;Literal">UTF-8</rdf:li>
            <rdf:li rdf:datatype="&prf-dt;Literal">ISO-10646-UCS-2</rdf:li>
          </rdf:Bag>
        </prf:CcppAccept-Charset>
      </rdf:Description>
    </prf:component>
  </rdf:Description>
</rdf:RDF>
```

## 7.7.2 Profile Example 2

The following profile includes attributes from UAProf's core vocabulary as well as attributes from another vocabulary.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
  <!ENTITY ns-rdf  'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!ENTITY ns-prf  'http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-
YYYYMMDD#'>
  <!ENTITY prf-dt  'http://www.openmobilealliance.org/tech/profiles/UAPROF/xmlschema-
YYYYMMDD#'>
  <!ENTITY ns-mms  'http://www.openmobilealliance.org/tech/MMS/ccppschema-YYYYMMDD#'>
]>

<rdf:RDF xmlns:rdf="&ns-rdf;"
         xmlns:prf="&ns-prf;"
         xmlns:mms="&ns-mms;">

  <rdf:Description rdf:ID="MyDeviceProfile">
    <prf:component>
      <rdf:Description rdf:ID="HardwarePlatform">
        <rdf:type rdf:resource="&ns-prf;HardwarePlatform"/>
        <!-- see the previous definition of the HardwarePlatform component -->
      </rdf:Description>
    </prf:component>
```

```
     <prf:component>
      <rdf:Description rdf:ID="MMSCharacteristics">
        <rdf:type rdf:resource="&ns-mms;MMSCharacteristics"/>
        <mms:MmsMaxMessageSize rdf:datatype="&prf-dt;Number">2048</mms:MmsMaxMessageSize>
        <mms:MmsVersion rdf:datatype="&prf-dt;Literal">2.0</mms:MmsVersion>
        <mms:MmsCcppStreamingCapable rdf:datatype="&prf-
dt;Boolean">Yes</mms:MmsCcppStreamingCapable>
      </rdf:Description>
    </prf:component>
  </rdf:Description>
</rdf:RDF>
```

# 7.8 Extensions to the Schema/Vocabulary

While this specification provides a base vocabulary of property descriptions for User Agent Profiles, it is anticipated that implementors of new applications or device capabilities may, in the future, have a need for expressing new or additional attributes in the profile. The use of RDF enables an extensibility mechanism for CC/PP-based schemas that addresses the evolution of new types of devices, applications, or hardware/software. The vocabulary extensions will constitute a different RDF schema and will have a corresponding RDF model and syntactic representation. A particular profile may compose attributes from multiple schemas, namely the base vocabulary schema and the vocabulary extension schema.

The new schema/vocabulary MUST adhere to the following guidelines and recommendations for compliance with this specification and the CC/PP framework:

- New schemas and vocabularies MUST be uniquely identified using well-defined XML namespaces [XML-NS].
- Since RDF supports interoperability of schemas, the new schemas MUST NOT contain the same attributes (name and semantics) as specified in the base vocabulary. *Note that profiles can be composed from attributes from multiple schemas*. Moreover, extended attributes SHOULD NOT use attribute names that are reserved for future use by the base vocabulary (see Appendix B for more information).
- The schema vocabulary designer SHOULD have a mental model of the RDF document as an RDF graph, and should take care to express and verify the intuitions against common RDF tools. *This will eliminate potential ambiguity regarding the schema and its semantics, and interpretation by tools.*
- Since RDF's abbreviated syntax is not permitted for profiles, schema extensions also MUST NOT use the abbreviated syntax defined in [RDF-Syntax].
- Control characters and binary bytes MUST be encoded in conformance with XML [XML] syntax specifications.
- The following naming conventions MUST be adhered to while defining new profile attributes and components:

  All components and attributes MUST start with an upper case letter.

  Multiple word names MUST be described as one word, with the first letter in the second word in upper case. There MUST NOT be any separator or underscore between the words. For example, Software Platform must be identified as *SoftwarePlatform*.
- Cyclic references (URIs) MUST NOT be used during schema design.

## 7.8.1 Addition of Components

In defining components within the new schema, the designer MUST apply the following rules:

- The components used in a standardized base vocabulary MUST not only contain enough information to meet the needs of a majority of current content providers but also allow for the meaningful introduction of future device capabilities into the profile.
- As long as the new attributes fall within the realm of one of the base profile components (HardwarePlatform, SoftwarePlatform, NetworkCharacteristics, WAPCharacteristics, PushCharacteristics, and BrowserUA), the designers of new schemas must add those attributes to these defined components (instead of creating new components).
- New applications or user agents may need to assert their capabilities and preferences in a new component. The schema designer MUST uniquely define the attributes to be included in the vocabulary for the component. An

attribute definition includes identifying the semantic description, the attribute's type the attribute's resolution rule and sample values.

- The schema for the new components MUST follow the general schema layout for the core profile components. Therefore, the nested description for each component MUST incorporate the following structure:

  A subordinate description block to identify default attributes if any, preferably referenced by a resource URI.

  Any overrides/modifications to the default descriptions outside the Default subordinate block.

### 7.8.2 Addition of Attributes

In defining attributes within the new schema, the designer MUST apply the following rules:

- The attributes described in the vocabulary MUST be atomic and semantically unambiguous. The names used to define/represent the attributes MUST be unique within the namespace.
- To preserve the simplicity of the schema, the use of complex data types such as containers as well as schema validation conditions such as value ranges, constraints and units SHOULD be minimized.
- For values that are complex data types (such as lists), RDF containers MUST be used. The **rdf:resource** construct MUST be used, where appropriate, to indicate to the RDF parser that the value of an attribute is indeed a resource and not a literal.
- The rule for resolving multiple descriptions of an attribute MUST be specified as part of the semantics. The rule must specify a *Locked*, *Override*, or *Append* treatment for value resolution. If no rule is specified, a default rule of *Override* is assumed for the attribute.
- Escape control characters and binary bytes MUST be in conformance with XML syntax [XML].

# 8. User Agent Profile Transport

*This section is normative, unless explicitly stated otherwise. Examples are informative.*

This section defines the transport mechanisms for User Agent Profile data.

From the mobile client to the WAP gateway/proxy, the User Agent profile data may be transferred over one of two protocol variants :

- Wireless Profiled HTTP; hereafter referred to as W-HTTP, or
- a combination of WSP and HTTP 1.1; hereafter referred to as WSP

Additionally a mobile client may transfer its user profile data directly to the origin server using W-HTTP where no WAP gateway/proxy is being used.

A client MUST at least support one of these mechanisms in order to support UAProf. Each mechanism is functionally equivalent but a major difference is that CC/Ppex is not used in the W-HTTP variant.

From the WAP gateway/proxy to the origin server the following applies: A WAP proxy (receiving W-HTTP formatted CPI information) MUST use the W-HTTP transport method towards the origin server. A WAP gateway (receiving WSP formatted CPI information) SHOULD use the W-HTTP transport method as specified in section 8.2.3.3 towards the origin server. Alternatively, CC/Ppex over HTTP as specified in section 8.2.3.4 MAY be used. The use of HTTPex is not recommended. Thus, an origin server MUST be prepared to handle CPI information in either W-HTTP or CC//Ppex over HTTP format.

## 8.1 Transport Over W-HTTP

In the case where the mobile terminal supports wireless profiled HTTP [W-HTTP] the profile is transported using meta data defined by this specification. The CC/PP Framework remains unaltered. The defined mechanism provides a functional equivalent for the CC/PP exchange protocol [CCPPex] but the definition of the syntax and semantics of the transport remains in this specification.

### 8.1.1 Using W-HTTP to transport CC/PP

The following extension headers are defined to transport CC/PP in W-HTTP. The defined extension headers are considered to be end to end headers. The headers are defined as general headers because they may be used in requests and responses in the push and pull use cases defined in section 6.

#### 8.1.1.1 X-WAP-PROFILE

The *x-wap-profile* header is a general header field which MUST contain the following :

- a URI referencing the CPI or
- a reference to a profile difference, transported using the *x-wap-profile-diff* or
- a combination of multiple instances of these two types of data.

Note: The URI referencing the CPI is generally expected to resolve to a specific URL.

This data MAY be generated by the mobile terminal or attached by an intermediary point in a request to an origin server. In the case of Push this header MAY be generated as a response to a request. In the case of Push this data may be cached. However this header MUST be present in any request or response when UAProf is used. The ABNF [RFC2234] format of the header is:

*Header Name:* `x-wap-profile`

*Description:* List of supported references to CPI data pertaining to the terminal or intermediate performance enhancing proxies.

*Format:*

```
x-wap-profile       = "x-wap-profile" ":" 1#reference

reference           = <"> ( absoluteURI | profile-diff-name ) <">

absoluteURI         = <an absolute URI as defined by RFC2396>

profile-diff-name   = profile-diff-seq "-" profile-diff-digest

profile-diff-seq    = ("1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9")
                      *DIGIT

profile-diff-digest = *OCTET ";" < MD5 message digest encoded by base64 >

DIGIT               = <any US-ASCII digit "0"..."9">
```

*Default:* None

The document referred to by absoluteURI is a profile that MAY contain several component instances, each of a different type. Components MAY contain a **ccpp:defaults** element.

The *x-wap-profile* header MAY contain references to instances of the *x-wap-profile-diff* header (defined in the following section ). Each reference contains two parts, the sequence number and the profile-digest. The sequence number is used to determine the order of how the *x-wap-profile-diff* headers should be applied and the digest is used to validate that the profile-desc in the *x-wap-profile-diff* header value is correct.

The computation of the MD5 message digest takes place over the profile description defined in section 8.1.1.2, consisting of an XML/RDF document. It is introduced for the efficiency of the cache table look up in gateways, proxies and user agents and to ensure integrity of the profile description. Prior to the computation of the MD5 digest, the profile description is normalized as follows:

- Leading and trailing white spaces are eliminated. (white space as defined in RFC 2616 section 2.2: LWS)
- All non-trailing or non-leading linear white space (LWS) contained in the profile description, including line folding of multiple HTTP header lines, is replaced with one single space (SP) character. Note: This implies that property values, represented as XML attributes or XML element character data, MUST be adhering to white space compression as mandated in RFC 2616 section 2.2.

### 8.1.1.2 X-WAP-PROFILE-DIFF

The *x-wap-profile-diff* header is a general header and MAY be generated by the mobile terminal or an intermediate proxy to enhance or alter the CPI. There may be multiple profile differences, each profile difference must also have a reference in the *x-wap-profile* header which indicates the order in which differences should be applied. This header contains two parts, a sequence identifier and the entity which represents the part of the CC/PP description that is being enhanced. This header MAY be present in a request or response. In the case of Push this data may be cached.

*Header Name:* `x-wap-profile-diff`

*Description:* This header contains additional profile information which should be applied to the CPI prior to serving any content.

*Format:*

```
x-wap-profile-diff  = "x-wap-profile-diff" ":" profile-diff-seq
                      ";" profile-desc

profile-diff-seq    = ("1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9")
                      *DIGIT
```

```
profile-desc        =<XML document containing profile subset of
                      the UAProf schema >

DIGIT               = <any US-ASCII digit "0"..."9">
```

*Default:* None

The document specified in profile-desc is a profile that MAY contain several component instances, each of a different type. Components MAY contain a **ccpp:defaults** element.

The actual profile referred to in the *x-wap-profile* header URI is merged with its correlated *x-wap-profile-diff* header values in the order in which they occur. This process is applied to all URIs contained in the *x-wap-profile* header. Finally, the resulting profiles are merged in the order in which the corresponding URIs were defined in the *x-wap-profile* header.

The RDF/XML document that forms the "*x-wap-profile-diff*" header value MUST adhere to linear white space (LWS) substitution rules as per section 2.2 of RFC 2616.

### 8.1.1.3 X-WAP-PROFILE-WARNING

The *x-wap-profile-warning* header is a general header. Its presence indicates the level to which the response has been tailored in relation to profile data that has been supplied in the request. This header MAY be present in a request or response.

*Header Name:* `x-wap-profile-warning`

*Description:* This header is used by the server to indicate whether the CPI has been honoured when the response to the request was generated.

*Format:*

```
x-wap-profile-warning = "x-wap-profile-warning" ":" warning-code

warning-code          = 200 | 201 | 202 | 203 | 500
```

*Default:* None

The warning codes that are defined fall into the following categories :

- 1xx – reserved
- 100 – reserved
- 2xx – indicates whether the content has been adapted depending on the profile
- 5xx – indicates the server is incapable of processing CPI.

The *x-wap-profile-warning* may have the following values.

**200** Not applied

This value MUST be included if the content has not been tailored, and is sent in a representation which is the only representation available in the server.

07 Content selection applied

MUST be included if the included content has been selected from one of the representations available.

07 Content generation applied

MUST be included if the content has been tailored or generated as a result of applying the included profile.

07 Transformation applied

MUST be added by an intermediate proxy if it applies any transformation changing the content-coding based on the CPI data.

**500** Not Supported

Indicates that the entity sending this warning code does not support UAProf.

## 8.1.2 Protocol Procedures

### 8.1.2.1 Over The Air

In this transport variant the headers and their values are not compressed for over the air transmission. It is recommended that the hardware and software manufacturer only use the *x-wap-profile* header to indicate an `absoluteURI` as the reference for CPI for the mobile terminal when transmitted over the air. However this specification does not preclude the use *x-wap-profile-diff* in this case.

### 8.1.2.2 Combining X-WAP-Profile and X-WAP-Profile-Diff

If the *x-wap-profile-diff* header is included the *profile-diff-seq* MUST match a sequence number in the *x-wap-profile* header otherwise the associated profile-diff-desc MUST NOT be processed. The reference to each *x-wap-profile-diff* header contains two parts, a sequence number which governs the order in which the *x-wap-profile-diff* headers are processed and an MD5 message digest which is used to validate the profile-desc which is contained in the *x-wap-profile-diff*. If either the sequence or the MD5 validation does not match, the particular profile-diff MUST be ignored.

If the *x-wap-profile-diff* header is added by an intermediate proxy, it MUST NOT alter the existing sequence of *x-wap-profile-diff* headers, the proxy MUST append using the next available sequence number in numeric order.

The digest associated with an *x-wap-profile-diff* MUST be generated by applying MD5 message digest algorithm [RFC1321] and Base64 algorithm, section 6.8 in the MIME specification [RFC2045] to the corresponding *profile-desc* part of the header field-value. The MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit “fingerprint” or “message digest” of the input. The Base64 algorithm takes as input arbitrary binary data and produces as output printable encoding data of the input.

### 8.1.2.3 Relationship with HTTP Headers

The profile information referred to in the *x-wap-profile* and *x-wap-profile-diff* header does not supersede HTTP request or response header information.

### 8.1.2.4 Caching

The *x-wap-profile-warning* header MUST NOT be used for cache control purposes. If a server wishes to indicate a caching dependency based on these headers then it should use the Vary header as defined in section 14.44 of the HTTP 1.1 specification [HTTP].

### 8.1.2.5 Examples

This section is informative

| Use Case : HTTP Request | |
|---|---|
| *Mobile Terminal* | *Performance Enhancing Proxy* |
| Request → Origin Server | |
| GET http://anyuri/ HTTP/1.1<br>Host: plaintext<br>x-wap-profile: “http://profilerepository/”<br>… | GET http://anyuri/ HTTP/1.1<br>Host: plaintext<br>x-wap-profile: “http://profilerepository/”, “1-uKdjJHuhjHUuj”<br>x-wap-profile-diff:1; <?xml…./><br>… |
| *Mobile Terminal* | *Origin Server* |
| | Response ← |
| | HTTP/1.1 200 OK<br>Date:<br>x-wap-profile-warning:202<br>… |

| Use Case : Push Proxy Gateway Request for Profile | |
|---|---|
| *Mobile Terminal* | *Push Proxy Gateway* |
| | ← Request |
| | OPTIONS * HTTP/1.1<br>Host:<br>… |
| Response → | |
| HTTP/1.1 204 OK<br>Date:<br>x-wap-profile: “http://profilerepository/” | |

| Use Case : Push of Tailored Content | |
|---|---|
| *Mobile Terminal* | *Push Proxy Gateway* |

| | |
|---|---|
| | ← Request |
| | POST /wappush HTTP/1.1<br>Host:<br>x-wap-profile-warning:200<br>… |

# 8.2 User Agent Profile Transport Over WSP

This section specifies how Profiles are transported over WSP. Section 8.2.1 is an introduction and is not normative. The CC/PP Exchange Protocol over WSP, referred to as CC/PP-WSP, is specified in the normative Sections 8.2.2 and 8.2.3.

## 8.2.1 Introduction

### 8.2.1.1 The CC/PP Framework and the CC/PP Exchange Protocol Over HTTP

The Composite Capability/Preference Profiles (CC/PP) defines a framework for content negotiation [CCPP]. Section 7 defines a vocabulary of categories and attributes for WAP-enabled devices.

To transport CC/PP documents, or references to such profiles, over the Internet, the CC/PP Exchange Protocol Over HTTP MAY be used. It is specified in [CCPPex]. The CC/PP Exchange Protocol Over HTTP, referred to as CC/PP-HTTP, is used over HTTP [HTTP] and uses the HTTP Extension Framework [HTTPext]. The mapping onto WSP is specified in this section and is sometimes referred to as CC/PP-WSP.

The CC/PP Exchange Protocol specifies two new request header fields (*Profile*, *Profile-Diff*) and one new response header field (*Profile-Warning)*. The *Profile* header is used to transport one or many Profile identities, URIs, from the client to the server. This set of Profiles is used to construct the Composite Profile Information (CPI). The *Profile-diff* header is used to transport changes to the CPI. This means that the *Profile-Diff* header must always be used together with and referenced by the *Profile* header. The *Profile-Warning* header is used by the server to notify the client whether the request to use Profiles was fulfilled, partly fulfilled, or not fulfilled. To extend the HTTP protocol with the new header in a structured way, the HTTP Extension Framework MAY be used [HTTPext], however the headers defined in Section 8.1 SHOULD be used to transport CPI.

### 8.2.1.2 Using WSP to Transport CC/PP Profiles

The WSP protocol has some features that cannot be found in HTTP. To reduce the size of request messages the WSP client can cache headers in the gateway for the lifetime of a WSP session. The cached headers are called session headers and are sent to the gateway during session establishment. The client can use the **Resume** operation of WSP to update the session headers during the session. At any time the client or the gateway can terminate the session and establish a new one, with new session headers. Moreover, the client MAY provide additional headers with each request; these request headers are merged with the cached headers (and, possibly, other WSP headers) to generate the final CPI that is transmitted over HTTP.

WSP uses *Profile* and *Profile-Diff* headers to convey the CPI. A *Profile* header contains a single URL, referencing an externally accessible CPI document. The *Profile-Diff* header contains a CPI document. Multiple *Profile* and *Profile-Diff* headers MAY be cached by the gateway and/or included with a request.

The WAP gateway combines request headers with cached session headers to create HTTP requests [WAE]. The following list summarizes the WSP header management:

- If one or many *Profile* headers are cached in the server, the client can override all of them within the scope of a particular request by sending one or more *Profile* headers in a single request message
- If one or many *Profile* headers, but no *Profile-Diff* headers, are cached in the gateway, the client can append one or many *Profile-Diff* headers within the scope of a particular request by including them in a request message.
- If one or many *Profile-Diff* headers are cached in the gateway, the client can override all of them within the scope of a particular request by sending one or more *Profile-Diff* headers in a single request message.

- If one or many *Profile-Diff* headers, but no *Profile* headers, are cached in the gateway, the client can append one or many *Profile* headers within the scope of a particular request by including them in a request message.
- By using the Resume function, the client can update the session headers, without establishing a new session[3].

#### 8.2.1.3 Differences Between CC/PP-HTTP and CC/PP-WSP

Among the differences between CC/PP-HTTP and CC/PP-WSP:

- In the CC/PP-WSP *Profile-Warning* response header, the warning text is not included.
- In CC/PP-HTTP, multiple profile references are transmitted in one *Profile* header. The header may reference both external profiles (via a URL) or embedded profiles (via a URN containing an MD5 checksum of the embedded profile). Embedded profiles are associated with custom headers computed from the profile's MD5 checksum. In WSP, a *Profile* header can only transmit one profile reference, but multiple *Profile* headers can be transmitted in the same WSP header; the *Profile* header only references external profiles via URL. In addition, multiple *Profile-Diff* headers, each containing an embedded profile, may be transmitted in the same WSP header. No functionality is lost, because the WAP gateway is capable of generating the MD5 checksums for the *Profile-Diff* documents and constructing a complete *Profile* header for transmission over HTTP.

### 8.2.2 Structure and Encoding of Header Fields

#### 8.2.2.1 The Profile Header

The syntax of the Profile header SHOULD conform to the production of [**1.**].

**[1.]** Profile = Profile-field-name Profile-field-value

**[2.]** Profile-field-name = Short-integer (as defined in [WSP])

**[3.]** Profile-field-value = Uri-value (an absolute URI per [RFC2396])

**Example:**

0x35 http://anyco.com/anypda 0x00

In the above example the profile URI is http://profile.anyuri.com/anypda. Note that 0x00 signals the end of the URI-value string.

#### 8.2.2.2 The Profile-Diff Header

The syntax of the *Profile-Diff* header MUST conform to the production of [**4.**].

**[4.]** Profile-diff = Profile-diff-field-name Profile-diff-field-value

**[5.]** Profile-diff-field-name = Short-integer (as defined in [WSP])

**[6.]** Profile-diff-field-value = CCPP-profile

**[7.]** CCPP-profile = a RDF/XML document as described in section 8.1.1.2

**Example:**

0x36 <?xml …/>

[3] This is not possible in WSP 1.0.

### 8.2.2.3 The Profile-Warning Header

The syntax of the *Profile-Warning* header MUST conform to the production of [**8.**].

**[8.]** Profile-warning = Profile-warning-field-name Profile-warning-value

**[9.]** Profile-warning-field-name = Short-integer (as defined in [WSP])

**[10.]** Profile-warning-value = Warn-code | ( Value-length Warn-code Warn-target *Warn-date)

**[11.]** Warn-code = Short-integer

Status codes (and corresponding Short-integer values) are 100 (0x90) | 101 (0x91) | 102 (0x92) | 200 (0xa0) | 201 (0xa1) | 202 (0xa2) | 203 (0xa3) [CCPPex]

**[12.]** Warn-target = Uri-value | host [ ":" port ]

Uri-value is an absolute URI per [RFC2396]

**[13.]** Warn-date = Date-value (per the HTTP-date rule in [HTTP])

**Example 1:**

0x37 0x16 0x92 http://anyco.com/pda 0x00

In the above example:

- The length is 22 octets (0x16)
- The profile warning code is 102 (encoded as 0x92); and
- The profile URI is http://profile.anyuri.com/anypda
- No date is provided

**Example 2:**

0x37 0x92

In the above example, only the warning code is sent.

## 8.2.3 Protocol Procedures

### 8.2.3.1 Session Establishment

The following procedure is used to establish a WSP session that uses User Agent Profiles:

**[14.]** To indicate support for User Agent Profiles, the client MUST include the *Profile* header in the connect message.

**[15.]** **If the server responds with the Profile-Warning Header (set to warning code 100), the gateway MUST forward this to the client.**

**[16.]** If the client does not receive the *Profile-Warning* header with the warning code 100, it MUST NOT send any additional User Agent Profile headers during the WSP session.

During the session the client can use the Resume procedure to update the session header [WAE][4].

Session headers are cached in the gateway for the duration of the WSP session [WSP]. When the gateway generates HTTP request messages it combines the headers from the request message with the headers in the session's cache, according to the rules in section 8.2.3.2.

### 8.2.3.2 Combining Session and Request Headers

Upon reception of a WSP request, the WAP gateway combines the request headers with the cached WSP session headers. The result is a list of WSP headers that gets translated into one HTTP request (see section 8.2.3.3).

The following procedure is used to combine the request headers with the cached session headers into one list of headers:

[4] This is not possible in WSP version 1.0.

**[17.]** Headers from the WSP request MUST override cached WSP session headers according to the rules for client headers defined in [WAE].

**[18.]** If, after the previous operation, both request headers and session headers are present in the list of headers, request headers MUST follow after session.

Since request headers are appended to the list after the session headers, request headers will take precedence over session headers if the headers are applied to the profile in the same order as they are transported.

**Example 1:**

In this example, both request and session headers remain after the WSP request and the WSP session have been combined. In the result, the request header comes after the session headers in the list.

CC/PP cached session headers:

```
Profile: URIx
Profile: URIy
```

CC/PP request header:

```
Profile-Diff: 0x01 <?xml …>
```

Will result in:

```
Profile: URIx
Profile: URIy
Profile-Diff: 0x01 <?xml …>
```

**Example 2:**

In this example, the request headers override all session headers.

CC/PP cached session headers:

```
Profile: URIx
Profile: URIy
```

CC/PP request header:

```
Profile: URIz
Profile-Diff: 0x01 <?xml …>
```

Will result in:

```
Profile: URIz
Profile-Diff: 0x01 <?xml …>
```

### 8.2.3.3 Header Translation Between CC/PP-WSP and HTTP

Optionally, a WAP gateway MAY forward WSP requests as HTTP 1.1 requests [WAE]. In forwarding the request, a gateway MUST forward all CC/PP-WSP headers (defined in Section 8.2.2 and resolved at the gateway according to the rules of Section 8.2.3.2) as W-HTTP headers (defined in section 8.1) according to these rules:

**[19.]** Each CC/PP-WSP *Profile-Diff* header field is translated into exactly one W-HTTP *x-wap-profile-diff* header field. The HTTP *Profile-Diff* headers are generated dynamically as specified in [CCPPex].

**[20.]** A single W-HTTP *x-wap-profile* header field is generated as specified in [CCPPex] by listing the values of each CC/PP-WSP *Profile* header and the values of each dynamically generated W-HTTP *x-wap-profile-diff* header. The ordering of this list preserves the ordering of the corresponding WSP *Profile* and/or *Profile-Diff* headers in the WSP setup/Resume and WSP request messages, as appropriate.

**[21.]** One W-HTTP *x-wap-profile-warning* response header is translated into exactly one CC/PP-WSP *Profile-Warning* response header. The warning text from the W-HTTP header is not translated.

**[22.]** The client that wants to convey Accept header information MUST do so through standard WSP headers, such as Accept, Accept-Charset, and Accept-Language. Information contained in these headers MUST constitute part of the CPI.

In forwarding the HTTP request and generating the W-HTTP *x-wap-profile* and *x-wap-profile-diff* headers, a gateway MAY insert additional profile information into the request. If provided, this additional information MUST be presented by appending to the end of the *x-wap-profile* header either a URI or a dynamically generated *x-wap-profile-diff* header identifier. Accordingly, a compliant gateway MAY therefore introduce an *x-wap-profile* (and, if necessary, *x-wap-profile-diff* header) on behalf of a client whose WSP session has no cached *Profile* or *Profile-Diff* headers. This support enables a gateway to support User Agent Profiles on behalf of client devices that are otherwise unable to convey profile information.

**Example:**

The WSP headers:

```
Profile: URIx
Profile-Diff: 0x01 <?xml version="1.0"?><RDF>AA…
Profile: URIy
Profile-Diff: 0x01 <?xml version="1.0"?><RDF>BB…
```

Are translated into the following HTTP headers:[5]

```
x-wap-profile: "URIx", "1-uKdjJHuhjHUuj", "URIy", "2-jdsjhUHjsuHjU"
x-wap-profile-diff: 1;<?xml version="1.0"?><RDF>AA…
x-wap-profile-diff: 2;<?xml version="1.0"?><RDF>BB…
```

[5] The value of the *Profile-diff* digest is not real, see [CCPPex].

# 9. Origin Server Behaviour

*This section is informative.*

From the gateway to the origin server, the User Agent Profile is transported over the Internet. In the WAP architecture specification, and as specified in Section 8.2.3.3, HTTP is assumed to be the transport mechanism for Internet-related information, using the HTTP 1.1 [HTTP] mechanisms for header management and caching [WAE].

The transmission of profiles information is described in section 8.

A combination of the following components may be necessary to implement an Internet server capable of receiving User Agent Profile information:

- A HTTP 1.1 server [HTTP]
- The HTTP 1.1 Extension Framework [HTTPext]
- The CC/PP Exchange Protocol [CCPPex]

Upon receiving a User Agent Profile, an origin server may do the following (architecture is described in Figure 5.1):

- Retrieve profile from the profile repository
- Parse the profile
- Validate the syntax of the profile
- If error condition occurs inform gateway/proxy with the proper error code as specified in Section 8.2.2.3
- Resolve attribute values by applying overriding rules and default values, applying the algorithm described in Section 7.5
- Validate the attribute value types
- Customize content according to the information contained within the profile

# 10. Deployment Considerations

*This section is informative.*

End-to-end systems supporting User Agent Profiles will depend upon the support of many of the elements in the WAP architecture:

- Client devices may need to store, generate, and transmit profile information.
- Gateways need to forward profile information to proxies or servers.
- Proxies may need to modify some of the profile information according to services that they provide, or they may need to tailor their provided services according to the client's profile information. They may also provide profile persistence or caching services.
- Servers may need to utilize the profile information to help adapt content that is to be provided to the client device.

A variety of schemes may be used to provision and maintain the profile information. Designation of these methods and approaches is beyond the scope of this specification. However, this section outlines some concepts that may need to be taken into consideration in the preparation and deployment of a UAProf capable system.

The possible uses of CC/PP are described by the W3C in [CCPP].

## 10.1 Client Support

It is clear that this specification will not be applied to initial WAP products that conform to the WAP 1.1 specifications. For backward compatibility, therefore, future systems will need to support clients and gateways that are unaware of User Agent Profiles. Similarly, support for one-way and broadcast service paradigms will create service models different than that for web browsing. Generally, therefore, client device support for User Agent Profiles will take various forms and will need to be supported in a variety of ways.

### 10.1.1 Client Devices Not Supporting User Agent Profiles

For those devices that do not directly support or cannot transmit UAProf information, indirect support may be provided by the gateway. It may be possible to have a static profile provisioned at the gateway by a carrier or service provider. This profile would be presented by WAP gateways to proxies and servers on behalf of the device(s) involved. The client device in this case does not require any special provisioning or support for this service.

The WAP gateway may also support dynamic, customized profiles on behalf of these legacy devices and one-way devices. For example, the WAP gateway may apply a dynamically generated profile according to the particular device ID or subscriber invoking the service.

### 10.1.2 Client Devices Supporting User Agent Profiles

For those devices that provide UAProf capabilities, the support may take different forms and therefore demand various provisioning schemes. Some devices may only support a static profile header which is transmitted during WSP session invocation or on sending HTTP request. Other devices may also be able to collect and send user preference information during the WSP session invocation or on sending HTTP request.

#### 10.1.2.1 Static Header Support

For some client devices, a fixed header may be utilized to support the UAProf *Profile* header reported during WSP session creation or on sending HTTP request by using the HTTP extension framework. And for some other client devices, a fixed header may be utilized to support the UAProf *x-wap-profile* header reported on sending HTTP request. This fixed header may be common for all devices of the same model or may be somewhat individualized for each device. A profile could be loaded into non-volatile memory by the manufacturer, network operator, or service provider. Typically, this profile would simply be a URI reference to a shared set of preference data stored on some repository on behalf of all such devices. This shared set of profile information would likely be deployed by the manufacturer, network operator, or service provider to cover the service characteristics that they wish to enable.

Depending on the number of devices that share the static profile, the supplied URI may itself become a de-facto preference indicator for proxies or origin servers. For example, a reference to a profile at http://profile.anyuri.com/anypda that would be invoked by millions of client devices eventually could lead to content shaping operations based upon the URI itself rather than upon the particular profile attributes that the URI references. This could lead to a misuse of the UAProf capability because, strictly speaking, the profile referenced by the URI could change arbitrarily at its server, thereby leading to incorrect behavior at the proxies or origin servers. Consequently, once a URI emerges as a moniker to represent a default profile, care must be taken not to modify the profile that the URI references.

An individualized URI for the static profile could be stored on each device. This would provide the opportunity to have some particular device-specific information stored within the profile.

Alternatively, the entire static profile may be stored on the client. The definition of the values in such a schema may be performed in several ways: preset by the manufacturer during production; hardcoded in the user agent software; programmed by the network operator or service provider at time of subscriber lease/purchase; and/or, programmed over-the-air by the network operator or service provider. The methods that may be developed are beyond the scope of this specification.

#### 10.1.2.2 Session-Constant Support

The client device may have the capability to dynamically construct its profile and present that profile during the WSP session creation. The profile would be used throughout the WSP session lifetime.

The profile attributes delivered by the device may be user alterable or may be pre-set by the device manufacturer, network operator, or service provider. For example, a device may permit the user to select the default language (e.g. English, French, or German) but may prevent the user from altering certain other profile attributes, including the URI representing the basic characteristics of the device or network. The techniques for changing these attributes may differ: a user interface or form may be used for the user-modifiable values, and an over-the-air scheme may be used for the server-managed values. The methods that may be developed are beyond the scope of this specification.

#### 10.1.2.3 Request-Specific Support

For more dynamic profile updates during the WSP session, the client device needs to support the *Profile-Diff* header scheme. And when using UAProf over W-HTTP, each profile is Request-Specific.

For example, a user agent, like a browser, may allow the subscriber to indicate a preference for images. By changing the preference, the user agent would have to send an update over the WSP session indicating the change so that it could be included in the cached profile at the WAP gateway. Depending on the user interface model of the user agent, the change may be temporary (for the current WSP request) or more long term (service toggle). These would be reflected in whether the new attribute value is sent within a *Profile-Diff* included in a WSP request or a WSP **Resume** operation.

The device may update the cached session profile without direct user intervention. For example, a phone could be designed to block a user agent's access to the audio services while a call is in progress. When a telephone call is made, the device may send an updated *Profile-Diff* to the WAP gateway or the Origin Server to indicate that the user agent no longer has access to audio capabilities.

## 10.2 Repository Support

A profile repository typically would be an HTTP server that provides UAProf CPI elements upon request. UAProf profiles may reference data stored in repositories provided and operated by the subscriber; network operator; gateway operator, device manufacturer; or service provider. It is expected that there will be a variety of mechanisms put into place to create and manage the data in such repositories:

- Manufacturers and software vendors may provide static profile resources that describe the client devices that they produce. Such profiles will describe the hardware and standard software elements that exist on the client devices.
- Network operators may provide profile information that includes network characteristics, and gateway operators may provide profile information that defines permitted service levels through the communications infrastructure. Some of these operators may provide additional services enabling the storage of user preference information.
- Subscribers may look to other entities to store preference information.

- Service providers, including enterprise IT managers, may create profile repositories that reflect the needs of their hosted applications or services.

Independent of the content of the stored profile, policy controls may be imposed that limit what profile information is delivered to a particular requester. Any such limits, or methods employed to ensure compliance, are beyond the scope of this specification.

*The specification of an architectural element that can be used as a profile repository is not in the scope of this document.*

## 10.3 Interim Proxy Support

The CC/PP working group describes in [CCPP] how proxies can be used to do content adaptation and transformation.

# Appendix A. User Agent Profile Schema and Datatypes (Normative)

*This section is normative.*

A repository of schemas for UAProf's base vocabulary is available at the following URI:

```
http://www.openmobilealliance.org/tech/profiles/
```

The schemas in the repository are named **`ccppschema-YYYYMMDD`**. YYYY is the year, MM is the number of the month and DD is the day the schema was created (e.g. `20030226`). A new schema will be created when the base vocabulary changes (additions, modification, deletions, etc.). A user-friendly, description of the base vocabulary is provided in the above URI in HTML formt (e.g `ccppschema-20030226.html`). However, the schema is Normative. If an attribute has authoritative values, the authoritative values are listed in the HTML file.

As specified in [RDF] and [RDF-Schema], the datatypes used by the base vocabulary are defined in an XML Schema [XML-Schema]. The XML Schemas are also stored in the URI above and they are named **`xmlschema-YYYYMMDD`**. See the XML Schema for more information about the datatypes and their lexical constraints.

# Appendix B. Reserved Attributes (Normative)

*This section is normative.*

The following attributes are not currently included in the base vocabulary for User Agent Profiles, but are reserved for possible incorporation at a future date. Extended vocabularies SHOULD NOT use these attribute names.

| Name | Description | Type | Resolution Rule | Examples |
|---|---|---|---|---|
| LocationCapable | Indicates wheterh the device supports determination and transmission of its location. | Boolean | Override | “Yes”, “No” |
| LocationEncoding | Description of the encoding of the device’s location. | Literal | Locked | “lat-long”, “cell-id” |
| LocationValue | Description of the device’s location. | Literal | Override | |

# Appendix C. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [CREQ].

## C.1 Client

### C.1.1 Protocol

| Item | Function | Reference | Mandatory / Optional | Requirement |
|---|---|---|---|---|
| WAG-UA-proto-C-001 | Profile transport | Section 8 | M | WAG-UA-Proto-C-002 OR WAG-UA-Proto-C-003 |
| WAG-UA-proto-C-002 | Profile transport using WSP | Section 8.2 | O | WAG-UA-P-C-001 AND WAG-UA-P-C-002 AND WAG-UA-P-C-003 AND WAG-UA-P-C-004 AND WAG-UA-P-C-006 |
| WAG-UA-proto-C-003 | Profile transport using W-HTTP | Section 8.1 | O | WAG-UA-P-C-009 |

### C.1.2 Transport Variant : WSP

| Item | Function | Reference | Mandatory / Optional | Requirement |
|---|---|---|---|---|
| WAG-UA-P-C-001 | Profile header in WSP Connect | 8.2.1.2 | O | WAG-UA-P-C-006 AND WSP-CO-C-001 |
| WAG-UA-P-C-002 | Behaviour on missing Profile-warning in response | 8.2.3.1 | O | |
| WAG-UA-P-C-003 | Order of request and session headers | 8.2.3.2 | O | |
| WAG-UA-P-C-004 | Does the client comply with the rules in [14] and [16]? | 8.2.3.1 | O | |
| WAG-UA-P-C-005 | Additonal headers with requests | 8.2.3.3 | O | |
| WAG-UA-P-C-006 | Does the Profile header match the production rule [1]? | 8.2.2.1 | O | |
| WAG-UA-P-C-007 | Does the Profile-Diff header match the production rule [4]? | 8.2.2.2 | O | |
| WAG-UA-P-C-008 | Does the Profile-warning header match the production rule [8]? | 8.2.2.3 | O | |

### C.1.3 Transport Variant : W-http

| Item | Function | Reference | Mandatory / Optional | Requirement |
|---|---|---|---|---|
| WAG-UA-P-C-009 | Client generates x-wap-profile header | 8.1.1.1 | O | |
| WAG-UA-P-C –010 | Client generates x-wap-profile-diff header | 8.1.1.2 | O | WAG-UA-P-C-012 AND WAG-UA-P-C-013 |
| WAG-UA-P-C-011 | Client processes x-wap-profile-warning header | 8.1.1.3 | O | |
| WAG-UA-P-C-012 | x-wap-profile contains sequnence number anf digest of instances x-wap-profile-diff | 8.1.2.2 | O | |
| WAG-UA-P-C-013 | Corrupt x-wap-profile-diff header values will be ignored. | 8.1.2.2 | O | |

## C.2 Gateway

| Item | Function | Reference | Mandatory / Optional | Requirement |
|---|---|---|---|---|
| WAG-UA-proto-S-001 | Profile transport | Section 8 | M | WAG-UA-proto-S-002 OR WAG-UA-Proto-S-003 |
| WAG-UA-proto-S-002 | Profile transport using WSP | Section8.2 | O | WAG-UA-P-S-001 AND WAG-UA-P-S-002 AND WAG-UA-P-S-003 AND WAG-UA-P-S-004 AND WAG-UA-P-S-005 AND WAG-UA-P-S-006 AND WAG-UA-P-S-010 |
| WAG-UA-proto-S-003 | Profile transport using W-HTTP | Section8.1 | O | WAG-UA-P-S-011 AND WAG-UA-P-S-012 AND WAG-UA-P-S-013 |

### C.2.1 Transport Variant WSP

| Item | Function | Reference | Mandatory / Optional | Requirement |
|---|---|---|---|---|
| WAG-UA-P-S-001 | Does the gateway comply with the rules in [15]? | 8.2.3 | O | WSP:MSF |
| WAG-UA-P-S-002 | Does the gateway comply with the rules in [17]? | 8.2.3.2 | O | |
| WAG-UA-P-S-003 | Does the gateway comply with the rules in [18]? | 8.2.3.2 | O | |
| WAG-UA-P-S-004 | Does the gateway comply with the rules in [19]? | 8.2.3.3 | O | |
| WAG-UA-P-S-005 | Does the gateway comply with the rules in [20]? | 8.2.3.3 | O | |
| WAG-UA-P-S-006 | Does the gateway comply with the rules in [21]? | 8.2.3.3 | O | |

| | | | | |
|---|---|---|---|---|
| WAG-UA-P-S-007 | Caching of Profile and Profile-Diff headers | 8.1.2 | O | WSP:MSF |
| WAG-UA-P-S-009 | Transmission of multiple profiles in one header | 8.1.3 | O | |

### C.2.2 Transport Variant W-HTTP

| Item | Function | Reference | Mandatory / Optional | Requirement |
|---|---|---|---|---|
| WAG-UA-P-S-011 | Does the gateway pass x-wap-profile header through | 8.1.1 | O | |
| WAG-UA-P-S-012 | Doess the gatway pass x-wap-profile-diff header through | 8.1.1 | O | |
| WAG-UA-P-S-013 | Does the gateway pass x-wap-profile-warning header through | 8.1.1 | O | |

## C.3 Schemas

| Item | Function | Reference | Mandatory / Optional | Requirement |
|---|---|---|---|---|
| WAG-UA-Schema-001 | Is the schema a valid WAP User Agent Profile Schema? | | M | WAG-UA-S-002 AND<br>WAG-UA-S-003 AND<br>WAG-UA-S-004 AND<br>WAG-UA-S-005 AND<br>WAG-UA-S-006 AND<br>WAG-UA-S-008 AND<br>WAG-UA-S-009 AND<br>WAG-UA-S-010 AND<br>WAG-UA-S-011 AND<br>WAG-UA-S-012 AND<br>WAG-UA-S-013 AND<br>WAG-UA-SA-002 AND<br>WAG-UA-SA-003 AND<br>WAG-UA-SA-005 AND<br>WAG-UA-SA-006 AND<br>WAG-UA-SA-007 AND<br>WAG-UA-SA-008 AND<br>WAG-UA-SA-010 AND<br>WAG-UA-SP-002 AND<br>WAG-UA-SP-003 AND<br>WAG-UA-SP-004 AND<br>WAG-UA-SP-005 AND<br>WAG-UA-SP-006 AND<br><br>WAG-UA-SX-001 AND<br>WAG-UA-SX-002 AND<br>WAG-UA-SX-003 AND<br>WAG-UA-SX-004 AND |

| Item | Function | Reference | Mandatory / Optional | Requirement |
|---|---|---|---|---|
| | | | | WAG-UA-SX-005 |
| WAG-UA-S-001 | Does the schema correspond to a vocabulary specified in conformity with the RDF standard? | 7.1 | O | |
| WAG-UA-S-002 | Is the namespace of the schema identified using the prefixes “rdf” and “rdfs”? | 7.1 | M | |
| WAG-UA-S-003 | Are all namespace declarations in the schema declared as attributes of the “rdf:RDF” root element? | 7.1 | M | |
| WAG-UA-S-004 | Are all namespace aliases defined outside of nested XML elements of the schema? | 7.1 | M | |
| WAG-UA-S-005 | Does a unique URI serve as an unambiguous identifier for the vocabulary? | 7.1 | O | |
| WAG-UA-S-006 | Does the schema consist of at least one component? | 7.1 | O | |
| WAG-UA-S-007 | Does each component in said schema describe a set of attributes within one or more description blocks? | 7.1 | O | |
| WAG-UA-S-008 | Do all components in the schema have the same schema structure (layout)? | 7.1 | O | |
| WAG-UA-S-009 | Is every component in the schema an object of the type Class? | 7.1 | O | |
| WAG-UA-S-010 | Does each component’s default description block (if present) contain a subordinate description block for default attributes? | 7.1 | O | |
| WAG-UA-S-011 | Are default attributes described within the default description block? | 7.1 | O | |
| WAG-UA-S-0012 | Are descriptions to override default values included in the component description, but outside the default description block? | 7.1 | O | |
| WAG-UA-S-013 | Schema extensions adhere to the guidelines and recommendations. | 7.8 | O | |

### C.3.1 Attributes

| Item | Function | Reference | Mandatory / Optional | Requirement |
|---|---|---|---|---|
| WAG-UA-SA-001 | Does each profile attribute belong to only one component in the schema? | 7.3 | O | |
| WAG-UA-SA-002 | Is a domain constraint (rdfs:domain) used to describe the attribute-component relationship in the schema? | 7.3 | O | |

| | | | | |
|---|---|---|---|---|
| WAG-UA-SA-003 | Is each profile attribute defined using a name and value pair syntax? | 7.3 | O | |
| WAG-UA-SA-005 | Is every attribute description within an RDF description embedded inline to denote the value, or expressed as an RDF resource? | 7.3 | O | |
| WAG-UA-SA-006 | Are all multi value and composite attributes described as RDF resources? | 7.3 | O | |
| WAG-UA-SA-007 | For all cases when lists of values are associated with given attributes, are RDF containers (Bag or sequence) used? | 7.3 | O | |
| WAG-UA-SA-008 | Is the description of the attribute within the **ccpp:defaults** property resolved first?<br>Do other descriptions of the attribute override the attribute's **ccpp:defaults** description?<br>Is the ultimate value of the attribute determined by the resolution rules for that attribute?<br>(see section 7.3 (*Locked*, *Override*, *Append*)) | 7.3 | O | |
| WAG-UA-SA-010 | Does the attribute follow the naming conventions defined in RDF, and the RDF schema? | 7.3 | O | |

## C.3.2 Profile

| Item | Function | Reference | Mandatory / Optional | Requirement |
|---|---|---|---|---|
| WAG-UA-SP-001 | Is the root node's child element (rdf:Description element) identified with an invariant node ID? | 7.4 | O | |
| WAG-UA-SP-002 | Does the profile contain one or more attributes identified in the base vocabulary? | 7.4 | O | |
| WAG-UA-SP-003 | Are the component instances of the components identified in the schema? | 7.4 | O | |
| WAG-UA-SP-004 | Do all profile parsers parse the profile based on the component type? | 7.4 | O | |
| WAG-UA-SP-005 | Is the relative order of the profiles to be merged maintained? | 7.5 | M | |
| WAG-UA-SP-006 | Are ccpp:defaults property values loaded first? | 7.5 | M | |
| WAG-UA-SP-007 | A profile's attribute values must be of the appropriate type. | 7.2 | M | |

# Appendix D. Change History (Informative)

## D.1 Approved Version History

| Reference | Date | Description |
|---|---|---|
| OMA-TS-UAProf-V2_0 | 17 Jan 2006 | Approved by TP<br>Ref TP Doc# OMA-TP-2005-0351-INP_UAProf_V2_0_for_final_approval |
| OMA-TS-UAProf-V2_0 | 06 Feb 2006 | Implementation of Class 3 CRs:<br>OMA-UAPROF-2005-0024-CR-UAPROF-Appendix-A<br>OMA-UAPROF-2005-0022R01-CR_UAPROF20<br>Note. The service indicator part of the version number has not been incremented as the material had not yet been published at the time when the changes were incorporated, |