A test client API for CC/PP and UAProf

Charles Smith[1], Mark H. Butler
Information Infrastructure Laboratory
HP Laboratories Bristol
HPL-2002-269
October 11th , 2002*

E-mail: chasmi@hplh.hpl.hp.com, smithcr@tcd.ie, mark-h_butler@hp.com

CC/PP, UAProf, capability description, delivery context, device independence, testing, protocol

CC/PP is a standard proposed by the W3C to allow devices to communicate their capabilities to other devices. UAProf is a related standard proposed by the Open Mobile Alliance (formerly the WAP Forum). Here we describe a Java test client for CC/PP that can test CC/PP aware servers using HTTP. The client implements all the HTTP based protocols currently used for transmission of CC/PP and UAProf profiles and allows users to test the performance of a server under unusual conditions.

* Internal Accession Date Only
Approved for External Publication
[1] Trinity College Dublin, Dublin, Eire.

A test client API for CC/PP and UAProf

Charles Smith (chasmi@hplb.hpl.hp.com, smithcr@tcd.ie)*
Mark H. Butler (mark-h_butler@hp.com)
Hewlett Packard Laboratories, Bristol UK
12 September 2002

Abstract

CC/PP is a standard proposed by the W3C to allow devices to communicate their capabilities to other devices. UAProf is a related standard proposed by the Open Mobile Alliance (formerly the WAP Forum). Here we describe a Java test client for CC/PP that can test CC/PP aware servers using HTTP. The client implements all the HTTP based protocols currently used for transmission of CC/PP and UAProf profiles and allows users to test the performance of a server under unusual conditions.

Keywords

CC/PP, UAProf, Capability Description, Delivery Context, Device Independence, Testing, Protocol

1 Introduction

The CC/PP (Composite Capabilities/Preferences Profiles)[1] standard provides a uniform way for devices to transmit a profile of their characteristics to other devices. As the number of CC/PP aware devices and servers increases, it is useful to be able to test their interoperability. Specifically, this test client API was created to test DELI[2], a CC/PP Java servlet developed at HP Labs Bristol. There can be considerable variations between the format of different device profiles and the way in which they are transmitted so simulation of these variations is an important element of the testing process.

Currently there are four broadly equivalent protocols defined for CC/PP profiles, as the CC/PP specification does not impose constraints on the transmission protocol. This means that devices could theoretically use any protocol, therefore this test client is designed to be easily extensible should new protocols be defined. It is likely that any protocol defined to transmit CC/PP profiles in the future will be similar to the ones currently in use, but in the authors' opinion it would be advantageous to adopt a single standardised protocol for CC/PP, as this would make implementation of a CC/PP server easier. At present deployed devices and software already deviate from the defined protocols, so it is harder to guarantee that a server will be able to process any request containing CC/PP device profiles. The test environment described here can also replicate some of the better known deviations, in order to test that servers can still interoperate with these implementations.

* Charles Smith is a summer intern at HP Labs Bristol from Trinity College Dublin, Eire.

2 Protocols and requests

2.1 CC/PP Exchange Protocol [CCPPex]

The CC/PP Exchange Protocol (hereafter referred to as CCPPex) is detailed in a W3C note[3] and makes use of the HTTP Extension Framework (HTTPex)[4] to transmit profile information. HTTPex provides extensions using numerical namespaces which are pre-pended to any HTTP request header which belongs to the particular extension. The extensions can be declared to be mandatory (i.e. a server must implement them or else return an error) or optional (i.e. a server can ignore the extension if it does not support it). A mandatory extension is introduced using a *Man* request header in a HTTP request, and an optional extension uses an *Opt* header. The value of this header contains the extension identifier (for example http://www.w3.org/1999/06/24-CCPPexchange) and a numerical namespace declaration e.g.:

```
Man: "http://www.w3.org/1999/06/24-CCPPexchange" ; ns=25
```

In addition to *Man* and *Opt*, requests can also be declared to be hop-by-hop rather than end-to-end, by replacing *Man* with *C-Man* and *Opt* with *C-Opt*, and including a *Connection* header. If the *Man* or *Opt* headers are used the profile information should be transmitted intact to the origin server whereas if *C-Man* or *C-Opt* are used it should be analysed by intervening proxies.

A *Profile* header must be included if CC/PP information is being sent. This contains a list of all reference profile and profile diffs (profiles transmitted inline). A reference profile is sent as a URL whereas the profile diff is represented by a profile diff sequence number and a profile diff digest that contains an MD5[5] encoding of the content of the profile corresponding to the profile diff. The profile diff digest can be used to identify profile diffs for caching purposes, and to protect it from modification. An example profile diff name is:

```
1-Rb0sq/nuUFQU75vAjKyiHw==
```

Here, the leading 1 indicates the profile diff sequence number, and the characters after the hyphen are the diff digest.

The profile corresponding to the profile diff is transmitted as a *Profile-Diff* header which contains the profile diff sequence number (as used in the *Profile* header) and the content of the profile diff.

An example of the headers needed to send a profile ref and a profile diff are shown below:

```
Man: "http://www.w3.org/1999/06/24-CCPPexchange" ; ns=25
25-Profile: "http://www.ex.com/hw," "1-CWccARHXxtYJE+rKkoD8ng=="
25-Profile-Diff-1: <?xml version="1.0"?> <rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#">
       <rdf:Description rdf:ID="MyDeviceProfile">
       <prf:component>
       <rdf:Description rdf:ID="HardwarePlatform">
       <rdf:type
       rdf:resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-
       20010430#HardwarePlatform"/>
       <prf:SoundOutputCapable>No</prf:SoundOutputCapable>
       </rdf:Description>
       </prf:component>
```

```
      </rdf:Description>
      </rdf:RDF>
```

2.2 Wireless Profiled HTTP [W-HTTP]

The Wireless Profiled HTTP protocol (hereafter referred to as W-HTTP) is functionally equivalent to the CCPPex protocol but was defined in the UAProf[6] specification created by the OMA. It has the advantage over CCPPex that it does not use numerical namespaces or non-standard HTTP methods. The Profile header of CCPPex is replaced by *x-wap-profile*, and the Profile-Diff header is replaced by *x-wap-profile-diff*. The content of these headers remains equivalent to CCPPex. Sending a profile ref and diff with this protocol is done as follows:

```
x-wap-profile: "1-CWccARHXxtYJE+rKkoD8ng=="
x-wap-profile-diff: 1;<?xml version="1.0"?> <rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#">
      <rdf:Description rdf:ID="MyDeviceProfile">
      <prf:component>
      <rdf:Description rdf:ID="HardwarePlatform">
      <rdf:type
      rdf:resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-
      20010430#HardwarePlatform"/>
      <prf:SoundOutputCapable>No</prf:SoundOutputCapable>
      </rdf:Description>
      </prf:component>
      </rdf:Description>
      </rdf:RDF>
```

An important difference between W-HTTP and CCPPex is that W-HTTP includes profile diff sequence numbers in the profile diff header value:

```
x-wap-profile-diff:1;
```

while CCPPex stores this inside the profile diff header name:

```
25-Profile-Diff-1:
```

2.3 The Wireless Session Protocol [WSP]

The Wireless Session Protocol (WSP[7]), defined in the UAProf specification, is designed to allow transmission of profile information from WAP phones to WAP gateways. It encodes profiles in a binary form (WBXML[8]) to reduce the amount of data that needs to be sent. Gateways will translate requests using WSP into CCPPex, and then forward this to the origin server. This test client does not support WSP, as it is not designed to be used through a WAP gateway.

2.4 The Session Initiation Protocol [SIP]

A protocol for CC/PP has been defined based on the Session Initiation Protocol (SIP)[9,10], which is designed to support session based services rather than sessionless services based on the HTTP model. Since this test client is designed as a HTTP client it does not support this protocol at present.

3 Response Warning Headers

Servers adapting content for devices based on CC/PP profile information can provide information pertaining to the type of adaptation (if any) that has been performed on the data

provided in a response, as well as the way in which profiles have been used. W-HTTP servers return an *x-wap-profile-warning* header, containing a warning code.

```
HTTP/1.1 200 OK
x-wap-profile-warning: 202
…
```

CCPPex clients expect a response to contain a *Profile-Warning* header. This header must include a numerical namespace, which presumably[*] should be defined in the response as part of an extension declaration, similar to those used in requests (see section 2). If necessary, an "Ext:" header (with no header value, just a header name) must be included, to indicate that a mandatory extension has been recognised. The "Cache-Control" header must indicate that this header is not cacheable. An example of a CCPPex response is:

```
HTTP/1.1 200 OK
Ext:
Man: "http://www.w3.org/1999/06/24-CCPPexchange" ; ns=99
99-Profile-Warning: 202 www.something.com "Content Generation Applied"
Cache-Control: no-cache="Ext"
...
```

4 HTTP Extensions Server Compatibility Issues

The HTTP Extensions standard defines a set of new HTTP methods not present in the HTTP 1.1 specification, such as M-GET and M-POST. These are used to indicate that a request contains a mandatory extension. These methods cause problems for some servers that do not implement HTTP extensions, since these will refuse to handle a request if they do not recognise the method it uses. The Java servlet API currently does not support HTTP extensions, since they are not defined in RFC 2068[11]. Java servlets are often used in order to implement various types of web applications therefore represents a significant obstacle to the use of the CCPPex protocol. W-HTTP does not suffer from this problem, as it conforms to HTTP 1.1. Therefore W-HTTP is preferable. Unfortunately though W-HTTP is only part of the WAP 2.0 specification, so WAP gateways supporting WAP 1.2.1 support UAProf by converting WSP to CCPPex. Hence in order to support currently deployed devices, a server must support a subset of the CCPPex protocol. Luckily though the authors have not encountered a WAP gateway that uses non-standard methods. This simplifies the implementation of CCPPex.

Furthermore the usefulness of the mandatory methods to CC/PP is questionable, since their function is to cause a server to return an error when it does not support an extension. In the case of CC/PP this would mean that a client would prefer to receive an error response rather than untailored content from servers which are not CC/PP aware. Since the client could use warning headers (see section 6) to determine whether content has been tailored by a server processing a request with an optional extension, situations where mandatory extensions are really necessary would seem to be quite rare.

5 Non-standard use of protocols

During interoperability testing, we have found that several CC/PP devices and implementations use these protocols in non-standard ways. The most common of these are:

[*] "Presumably" is used here since the CCPPex Note does not make this clear

[1] Not enclosing the values of the profile header fields in quotes

The protocols state that profile ref and diff names must be quoted in the profile header, but some devices do not adhere to this. See Appendix B for a sample request.

[2] Not including a *Profile* or *x-wap-profile* header i.e. using only the profile-diff headers

Both protocols require that these headers are used; if they are omitted difficulty can arise when identifying profile diffs, and when caching profiles. See Appendix A for a sample request.

[3] Calculating a profile diff digest from the entire profile diff header (not just the header content)

Some devices calculate the profile diff digest incorrectly, by including the header names when running the MD5 algorithm. In the CCPPEx and WHTTP protocols, the server should ignore the diff if the digest does not match as an intermediary may have modified the diff. Therefore if a client and server calculate the diff digest differently, this will mean the server will always ignore diffs from that client. Another possible consequence of this is unpredictable caching behaviour, since different requests may have different digests for the same profile diff.

[4] Using *profile* and *profile-diff* headers instead of *x-wap-profile* and *x-wap-profile-diff* [W-HTTP]

Certain devices and emulators use different headers from those defined by W-HTTP, making the server's job of isolating CC/PP profile data more complex.

[5] Using profile diff sequence numbers that are not consecutive in the absence of a profile header.

Using consecutive numbers is not required by the protocols, but can cause difficulty for servers when no profile header is present. This means the CC/PP processor must search a HTTP request for CC/PP profile headers. If the profile diff numbers are not consecutive then it is more difficult to ensure that all the relevant profile information is used. In this case it would be necessary to use some form of wildcarding to extract the header data.

Since these conditions occur often, the test client is designed to be able to simulate them for the purposes of checking the robustness and stability of a CC/PP aware server.

6 API summary

The code for test client API is available as part of the DELI distribution[12] under the package *com.hp.hpl.deliTestClient*. Javadoc of the API is available as part of the distribution.

6.1 Class Structure-UML Class Diagram

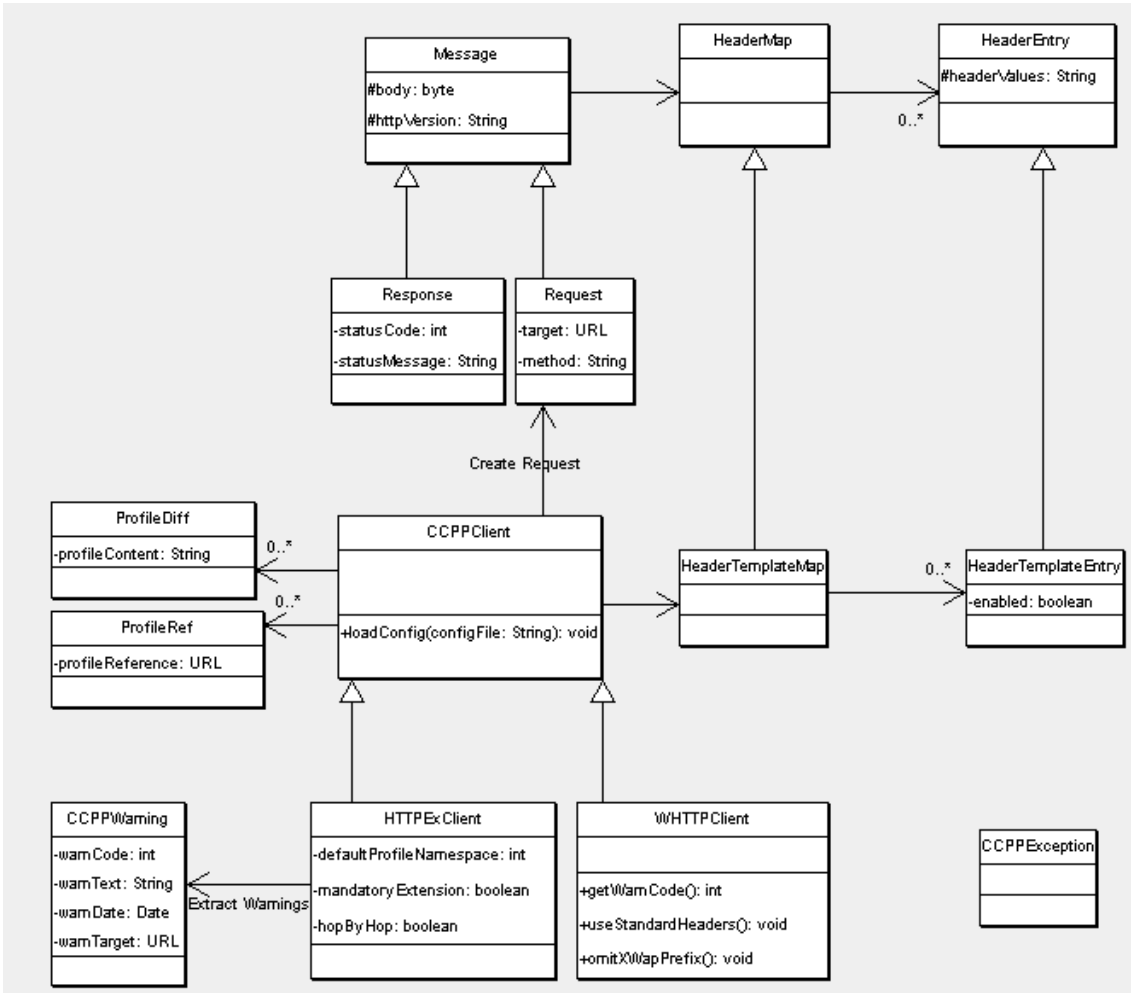

Figure 1 – Test Client Class Structure

6.2 Configuration File

The client can load configuration information from an XML configuration file. An example of such a file is shown below:

```
<?xml version="1.0"?>
<CCPPTestClientConfig>
  <headers>
    <header>
      <name>Accept</name>
      <value>text/html,text/plain</value>
    </header>
    <header>
      <name>User-Agent</name>
      <value>CCPP Test Client</value>
    </header>
    <header>
```

```
      <name>Connection</name>
      <value>close</value>
    </header>
  </headers>

  <quoteHeaderValues>true</quoteHeaderValues>
  <includeProfileHeader>true</includeProfileHeader>

  <CCPPex>
    <HTTPextNamespace>19</HTTPextNamespace>
    <mandatoryExtension>false</mandatoryExtension>
    <hopByHop>false</hopByHop>
  </CCPPex>

  <WHTTP>
    <omitXWapPrefix>false</omitXWapPrefix>
  </WHTTP>
</CCPPTestClientConfig>
```

If a CCPPex client configured using this information is then instructed to create a request for `http://www.example.com/index.html` using HTTP version 1.1 and the GET method, and including a profile ref to `http://www.profiles.com/device-profile`, then it will create a request as follows:

```
GET /index.html HTTP/1.1
Host: www.example.com
Content-Length: 0
Accept: text/html, text/plain
User-Agent: CCPP Test Client
Connection: close
Opt: "http://www.w3.org/1999/06/24-CCPPexchange" ; ns=19
19-Profile: "http://www.profiles.com/device-profile"
```

The `<headers>` element allows any number of `<header>` sub elements which contain header names and values. These will be included in any request the client sends. The `<quoteHeaderFields>` element specifies whether profile ref and diff names should be quoted. The `<includeProfileHeader>` element determines whether the client will send a *Profile* or *x-wap-profile* header with requests. The `<CCPPex>` element contains settings specific to the CCPPex protocol, which are the namespace number to use, and whether the extension is mandatory or optional and hop-by-hop or end-to-end. Similarly the `<WHTTP>` element only applies to that protocol, and determines whether the client should omit the *x-wap* prefix from the *profile* header.

6.3 Usage Example

Here is an example of using the API to request a web page using the GET method, and including a profile ref and profile diff:

```
HTTPExtClient client = new HTTPExtClient();
client.loadConfig(...);
URL target = ...
byte[] body = ...

client.addProfileRef(new ProfileRef(...));

client.addProfileDiff(new ProfileDiff(...));

Socket connection = new Socket(InetAddress.getByName(target.getHost()),80);

Request request = client.createRequest(HTTPExtClient.GET,target,body);
```

```
request.writeToSocket(connection.getOutputStream());
Response response = new Response(connection.getInputStream());
connection.close();
```

The first line creates a test client for the HTTPex protocol. It then loads configuration information from an XML document (see section 6.2). The target of the request is represented by a URL object, and the body of the request can be either a byte array or a String object. A profile ref and a profile diff are then added to the client. This means that any request generated by the client will include this profile information. A Socket object is created, connected to the server of the target URL (port 80 is the default HTTP port). The client is then used to create a Request object, containing all the appropriate CC/PP header information, and this request is sent by invoking a writeToSocket method. The server response can be retrieved by creating a new response object, which will parse the data from the socket's input stream. The transaction is now complete, and the socket is closed.

6.4 Command Line Interface

Two command line interfaces to the test client are provided, one for each existing CC/PP protocol. They send a request to a given target, including headers as specified by their parameters, and retrieve a response which is parsed and printed. Both take the same parameters, and both look for a configuration file (as described in section 6.2) located at config/CCPPClientConfig.xml inside the DELI distribution. Note to use the test client, first it is necesary to configure the CLASSPATH environment variable as described in the documentation accompanying DELI. The parameters are specified as follows:

```
HTTPExtTest [-r profileRefs] [-d profileDiffs] [-v HttpVersion] [-m HttpMethod] [-p
proxyServer] [-b requestBodyFile] [-w responseBodyFile] target

WHTTPTest [-r profileRefs] [-d profileDiffs] [-v HttpVersion] [-m HttpMethod] [-p
proxyServer] [-b requestBodyFile] [-w responseBodyFile] target
```

`[-r]` Indicates that the client should include the list of profile refs which follow it.

`[-d]` Indicates that the client should include the list of profile diffs which follow it.

`[-v]` Indicates which version of HTTP the client should use. 1.0 and 1.1 are supported, 1.0 being the default.

`[-m]` Indicates which HTTP method to use when sending the request. GET is the default.

`[-p]` Indicates that the client should use the given URL as the address of a HTTP proxy server to use.

`[-b]` Specifies a file from which to read the body of the request

`[-w]` Instructs the client to write the body of the response it receives to a given file.

6.4.1 CCPPex Example

```
java com.hp.hpl.deliTestClient.HTTPExtTest -r http://www.profiles.com/device-
profile -v 1.1 -m POST -b requestData.txt http://localhost
:8080/index.html
```

generates the following request

```
POST http://localhost:8080/index.html HTTP/1.1
Content-Length: 5901
Host: localhost
Accept: text/html,text/plain
19-Profile: "http://www.profiles.com/device-profile"
Opt: "http://www.w3.org/1999/06/24-CCPPexchange" ; ns=19
User-Agent: CCPP Test Client

[BINARY DATA]
```

6.4.2 W-HTTP Example

```
java com.hp.hpl.deliTestClient.WHTTPTest -d testProfile01.rdf -v 1.0 -m GET –p
http://http-proxy http://www.hp.com/
```

generates the following request

```
GET http://www.hp.com/ HTTP/1.0
x-wap-profile: "1-OZb/VXn3J+IHjBdh78HHvg=="
Content-Length: 0
Host: www.hp.com
Accept: text/html,text/plain
x-wap-profile-diff: 1;<?xml version="1.0"?> <rdf:RDF…
      </rdf:RDF>
User-Agent: CCPP Test Client
```

7 Summary

The test client described here is not designed to test the full functionality of a web server, but merely the degree to which it supports CC/PP. For this reason, many of the optional components of the HTTP 1.1 specification are not implemented, namely those which have no bearing on the operation of a CC/PP server. If it becomes necessary to test a particular aspect of a server requiring special request headers, the flexibility of the client allows this to be accomplished simply. As well as being able to test the compliance of a server with the HTTP based protocols for transmission of CC/PP, this client can be used to retrieve content from a server side CC/PP application and thus ensure its correct function.

Appendix A

Example CCPPex request sent from an Ericsson T39 phone through an Ericsson gateway.

```
GET /ccpp/html HTTP/1.0
Host: 127.0.0.1:8080
WAP-Connection: Stack-Type=CO
Opt: "http://www.w3.org/1999/06/24-CCPPexchange"; ns=56
Accept: application/vnd.wap.wmlc, application/vnd.wap.wbxml,
  application/vnd.wap.wmlscriptc, */*, text/vnd.wap.wml,
  application/xml, text/xml, text/vnd.wap.wmlscript
Accept-Language: en
```

```
User-Agent: EricssonT39/R201
Accept-Charset: *
56-Profile-Diff-1:<?xml version="1.0"?>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:prf="http://www.wapforum.org/UAPROF/ccppschema-19991014#">
  <!-- browser vendor site: Default description of properties -->
  <rdf:Description>
  <prf:CcppAccept>
  <rdf:Bag><rdf:li>application/vnd.wap.wmlc</rdf:li>
  <rdf:li>application/vnd.wap.wbxml</rdf:li>
  <rdf:li>application/vnd.wap.wmlscriptc</rdf:li>
  <rdf:li>*/*</rdf:li>
  <rdf:li>text/vnd.wap.wml</rdf:li>
  <rdf:li>application/xml</rdf:li>
  <rdf:li>text/xml</rdf:li>
  <rdf:li>text/vnd.wap.wmlscript</rdf:li>
  </rdf:Bag>
  </prf:CcppAccept></rdf:Description></rdf:RDF>
56-Profile-Diff-2:<?xml version="1.0"?>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:prf="http://www.wapforum.org/UAPROF/ccppschema-19991014#">
  <!-- browser vendor site: Default description of properties -->
  <rdf:Description><prf:CcppAccept-Charset><rdf:Bag>
  <rdf:li>*</rdf:li></rdf:Bag>
  </prf:CcppAccept-Charset></rdf:Description>
  </rdf:RDF>
Content-Type: text/vnd.wap.wml; charset=iso-8859-1
Date: Tue, 09 Apr 2002 11:31:16 GMT
Pragma: no-cache
Transfer-Encoding: chunked
Server: Apache Tomcat/4.0 (HTTP/1.1 Connector)
Cache-Control: no-cache, must-revalidate
Expires: -1
Set-Cookie: JSESSIONID=1783D571E4A7E5116339D189C238A923;Path=/
```

Appendix B

Example W-HTTP request sent by an Ericsson R520 phone through an Openwave gateway.

```
x-up-uplink: testgate.wdps.com
Profile: http://mobileinternet.ericsson.com/UAprof/R520.xml
Accept-Language: en
User-Agent: EricssonR520/R201 UP.Link/4.3.2
x-up-devcap-max-pdu: 3000
Accept-Application: 2
x-up-devcap-charset: us-ascii, iso-8859-1, utf-8, iso-10646-ucs-2
Connection: Keep-Alive
Encoding-Version: 1.2
Accept: application/vnd.wap.wmlc, application/vnd.wap.wbxml,
  application/vnd.wap.wmlscriptc, */*,
  text/x-wap.wml,text/vnd.wap.wml,text/x-hdml,
  text/html,text/vnd.wap.wmlscript
Bearer-Indication: 0
Accept-Charset: us-ascii, iso-8859-1, utf-8, iso-10646-ucs-2, UTF-8, *
```

[1] *W3C CC/PP Working Group*
http://www.w3.org/Mobile/CCPP/

[2] *DELI: A Delivery context Library for CC/PP and UAProf*

HP Labs Technical Report 2001-260
Mark H. Butler
http://www-uk.hpl.hp.com/people/marbut/DeliUserGuideWEB.htm

[3] *CC/PP exchange protocol based on HTTP Extension Framework*
W3C Note 24 June 1999
Hidetaka Ohto, Johan Hjelm
http://www.w3.org/TR/NOTE-CCPPexchange

[4] *Internet Engineering Task Force RFC 2774 - An HTTP Extension Framework*
February 2000
H. Nielsen, P. Leach, S. Lawrence
http://www.ietf.org/rfc/rfc2774.txt

[5] *Internet Engineering Task Force RFC 1321 – The MD5 Message-Digest Algorithm*
April 1992
R. Rivest
http://www.ietf.org/rfc/rfc1321.txt

[6] *OMA / WAP Forum UAProf Specification*
http://www1.wapforum.org/tech/documents/WAP-248-UAProf-20010530-p.pdf

[7] *Wireless Session Protocol Specification*
OMA / WAP Forum WAP-230-WSP
http://www.wapforum.org/

[8] *WAP Binary XML Content Format*
OMA / WAP Forum WAP-240-WBXML
http://www.wapforum.org/

[9] *Internet Engineering Task Force RFC 2543 - SIP: Session Initiation Protocol*
March 1999
M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg
http://www.ietf.org/rfc/rfc2543.txt

[10] *Internet Engineering Task Force Internet Draft - CC/PP Exchange Protocol based on SIP*
July 14, 2000
Takashi Nishigaya, Masanobu Yuhara
http://community.roxen.com/developers/idocs/drafts/draft-nishigaya-sip-ccpp-00.html

[11] *Internet Engineering Task Force RFC 2068 - Hypertext Transfer Protocol -- HTTP/1.1*
January 1997
R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee
http://www.ietf.org/rfc/rfc2068.txt

[12] DELI Java Servlet API,
http://delicon.sourceforge.net/