HTTP State Management

Approved Version 1.1 – 31 Mar 2008

Open Mobile Alliance

OMA-TS-HTTPSM-V1_1-20080331-A

Contents

1. SCOPE 4
2. REFERENCES 5
2.1 NORMATIVE REFERENCES 5
2.2 INFORMATIVE REFERENCES 5
3. TERMINOLOGY AND CONVENTIONS 6
3.1 CONVENTIONS 6
3.2 DEFINITIONS 6
3.3 ABBREVIATIONS 6
4. INTRODUCTION 7
5. HTTP STATE MANAGEMENT HEADERS 8
5.1 COOKIE 8
5.2 SET-COOKIE 8
6. WAP SPECIFIC HTTP STATE MANAGEMENT HEADERS 9
6.1 X-WAP-PROXY-COOKIE 9
6.2 X-WAP-PROXY-SET-COOKIE 9
7. WAP GATEWAY RESPONSIBILITIES 11
8. COOKIE PROXY RESPONSIBILITIES 12
8.1 PASS THROUGH COOKIE PROXY 12
8.2 COOKIE MANAGEMENT AND STORAGE 12
8.3 ASSOCIATING COOKIE STORAGE WITH CLIENTS 13
8.4 MANAGING PROXY COOKIES 13
9. USER AGENT RESPONSIBILITIES 14
9.1 HTTP STATE MANAGEMENT 14
9.2 COOKIE PROXY MANAGEMENT 14
APPENDIX A. CHANGE HISTORY (INFORMATIVE) 15
A.1 APPROVED VERSION 1.1 HISTORY 15
APPENDIX B. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) 16
B.1 SCR FOR USER AGENT FEATURES 16
B.2 SCR FOR COOKIE PROXY FEATURES 16

1. Scope

Wireless Application Protocol (WAP) is a result of work by the WAP Forum, now continued by the Open Mobile Alliance (OMA), to define an industry-wide specification for developing applications that operate over wireless communication networks. The scope of OMA is to define a set of specifications to be used by service applications for wireless communication devices. The wireless market is growing very quickly and reaching new customers and services. To enable operators and manufacturers to meet the challenges in advanced services, differentiation and fast/flexible service creation, WAP defines a set of protocols in transport, session and application layers. For additional information on the WAP architecture, refer to "*Wireless Application Protocol Architecture Specification"* [WAP].

This specification defines the HTTP state management model for the WAP architecture. The WAP HTTP state management model is an implementation of the HTTP State Management Mechanism, also known as "cookie management", as defined in [RFC2109]. On the World Wide Web, the HTTP State Management mechanism stores state information in a file ("cookie") on the client, as defined in [RFC2109]. The same mechanism can also be used over the WAP protocols, as HTTP headers are used to convey all state and state manipulation information.

Some WAP user agents may have motivation to store and manage cookies locally, as defined in [RFC2965]. This functionality follows precisely the current World Wide Web model, where cookies are typically stored and managed by regular web browsers.

This specification defines an additional mechanism to let an intermediate proxy store and manage cookies on behalf of the WAP client, as an alternative to client-local storage and management. Storing cookies in the network has many advantages. WAP user agents may have a limited storing capacity. When cookies are stored in the proxy, they do not have to be transmitted across the air, for every request/response transaction. In case the user changes device, and cannot move the cookies from the old device to the new one, the user can still access the cookies in the proxy via the new device. On the other hand, storing and managing cookies in the client allows the user to gain the benefit of the same cookies independent of the access point used. This aspect becomes more important in the future in conjunction with WAP gateway roaming architecture. Some users may prefer storing private information in the client, instead of depending on the security of the network. Because both models are complementary, this specification defines a dual approach to WAP HTTP state management, while still maintaining full interoperability between the implementations and RFC2965.

2. References

2.1 Normative References

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC2234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. November 1997, URL:http://www.ietf.org/rfc/rfc2234.txt

[RFC2616] "Hypertext Transfer Protocol - HTTP/1.1", R. Fielding, et al., June 1999. URL: http://www.ieft.org/rfc/rfc2616.txt

[RFC2109] "HTTP State Management Mechanism", D. Kristol, et al, February 1997. URL: http://www.ietf.org/rfc/rfc2109.txt

[RFC2965] "HTTP State Management Mechanism", D. Kristol, et al, October 2000. URL: http://www.ietf.org/rfc/rfc2965.txt

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

[WAE] "Wireless Application Environment Specification", WAP Forum, 04-November-1999. URL: http://www.wapforum.org/

2.2 Informative References

[WAP] "Wireless Application Protocol Architecture Specification", WAP Forum, 30-April-1998. URL: http://www.wapforum.org/

[WSP] "Wireless Session Protocol", WAP Forum, 30-April-1998. URL: http://www.wapforum.org/

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

Client	a device (or application) that initiates a request for a connection with a server.
Cookie Proxy	an intermediate program that acts as a user agent for the purpose of managing cookies and cookie storage on behalf of other user agents.
Origin Server	the server, on which a given resource resides or is to be created, often referred to as a web server or an HTTP server. (also referred to as a "server" in this specification.)
Proxy	an intermediate program that acts as both a server and a client for the purpose of making requests on behalf of other clients ([RFC2616]).
Server	see "origin server".
User	a person, who interacts with a user agent to view, hear or otherwise use a resource.
User Agent	a user agent is any software or device that interprets WML, WMLScript or other content. This may include textual browsers, voice browsers and search engines.
User Agent Session	a session which begins when user agent is activated and ends when it exits.

3.3 Abbreviations

OMA	Open Mobile Alliance
HTTP	Hypertext Transfer Protocol [RFC2616]
RFC	Request For Comments
URI	Universal Resource Identifier
URL	Universal Resource Locator
W3C	World Wide Web Consortium
WAE	Wireless Application Environment [WAE]
WAP	Wireless Application Protocol [WAP]
WSP	Wireless Session Protocol [WSP]

4. Introduction

The HTTP State Management Mechanism is defined in [RFC2965]. In short, RFC2965 defines a means whereby an origin server can request that a small unit of state (a "cookie") is stored in the user agent, and included in subsequent requests to the origin server. A variety of controls are available to the origin server, allowing it to control when the "cookie" is included in subsequent requests, when the "cookie" expires as well as other state management and transport controls.

As defined in [RFC2965], the user agent is responsible for cookie management. In this model, the WAP gateway conveys state information between the user agents and the origin servers. It is then the responsibility of the user agent to manage and store the cookies and to offer the user means for control over these functions.

Although RFC2965 puts cookie management in the user agent, it may, in some cases be convenient to take advantage of an architecture, which enables network elements to manage and store cookies. The WAP HTTP State Management Architecture defines the concept of a **Cookie Proxy**. The cookie proxy is an HTTP proxy or proxy equivalent (e.g., WAP Gateway) that manages cookies on behalf of WAP user agents that do not implement the HTTP state function directly. The cookie proxy is responsible for managing and storing cookies on behalf of the user agents, and modifies HTTP requests and responses to and from the user agent to implement this function.

This architecture supports clients with and without local cookie storage, and enables the user agent to control whether proxy cookie storage is enabled. In addition to this, WAP specific HTTP state management headers allow a simple synchronization scheme for user agent and proxy-based cookies. User agents can indicate if they rely on having cookies stored in the Cookie Proxy for a specific user agent session, and Cookie Proxy can notify the user agent if it has problems with their management.

The cookie proxy operation has three stages:

- Enabling or disabling the storage of cookies on the proxy. The user agent controls this function with an HTTP header.
- Origin server request for a cookie to be stored for the duration of the user agent session or for a certain predefined period of time. This is performed via the HTTP `Set-Cookie` header.
- Delivery of the cookie to the origin server in subsequent requests. This is performed via the HTTP `Cookie` header.

5. HTTP State Management Headers

5.1 Cookie

`Cookie` header is defined in [RFC2109]

NOTE: RFC2965 obsoletes RFC2109. However, for the definition of the Cookie header reference should be made to RFC 2109.

5.2 Set-Cookie

`Set-Cookie` header is defined in [RFC2109].

NOTE: RFC2965 obsoletes RFC2109. However, for the definition of the Set-Cookie header reference should be made to RFC 2109.

6. WAP Specific HTTP State Management Headers

6.1 X-Wap-Proxy-Cookie

This header is sent in the request from the user agent to indicate whether the Cookie Proxy should store cookies from origin servers or not. `X-Wap-Proxy-Cookie` header is also used to send status information from user agent to the Cookie Proxy.

```
x-wap-proxy-cookie     =  "X-Wap-Proxy-Cookie:" choice
Choice                 =  "cache" | "cache-has-state" | "delete" |
                          "none" | "session" | "session-has-state"
```

The choices are introduced briefly as follows:

- When the value is "`cache`" or "`cache-has-state`", the Cookie Proxy caches cookies and sends them to the origin server on behalf of the user agent. Requests and responses between the Cookie Proxy and the origin server include `Cookie` and `Set-Cookie` headers (see section 5). User agent appends `cache-has-state` value instead of `cache` in case it has received at least one `X-Wap-Proxy-Set-Cookie` header during the ongoing user agent session. This mechanism enables simple method for synchronization between user agents and Cookie Proxy. On account of this information Cookie Proxy can e.g. detect if the user agent session based cookies from the previous usage time should be discarded.

- When the value is "`delete`", the Cookie Proxy does not send any cookies to the origin server or store any received cookies. That is, the proxy acts as a filter ("cookie monster") and deletes all cookies before they are sent to the user agent.

- If the header is not present, or has a value of "`none`", the proxy passes all HTTP cookie headers through between the user agent and the origin server without interception. In this document, a Cookie Proxy executing this function is known as a Pass-Through Cookie Proxy. This is the default condition.

- When the value is "`session`", or "`session-has-state`", Cookie Proxy and user agent functionalities are combined. If Cookie Proxy receives a response containing a `Set-Cookie` header from the origin server, it decides the place for cookie storage according to the presence of the `Max-Age` attribute in the `Set-Cookie` header. This method can be used to separate session-based cookies from long-lived ones. The difference between `session` and `session-has-state` values is similar to the difference between `cache` and `cache-has-state` values, which is described above.

Note that status of the session is bound to the user agent session, which begins when the user agent starts and ends when it exits. Status is not related to a certain cookie-derived session, but it simply tells if the user agent has cookies managed by the Cookie Proxy during a particular user agent session. The user agent session is not related to the concept of session defined in [WSP].

6.2 X-Wap-Proxy-Set-Cookie

This header is sent in the response to the user agent from the Cookie Proxy to indicate that one or more cookies were received in a response from an origin server and stored in the cookie proxy and/or at least one cookie was sent in the corresponding request. In addition to this, Cookie Proxy uses `X-Wap-Proxy-Set-Cookie` header to report an erroneous status to the user agents.

```
x-wap-proxy-set-cookie      =    "X-Wap-Proxy-Set-Cookie: choice"
choice                      =    "state" | "error"
```

The choices are introduced briefly as follows:

- When value is "`state`", the user-agent is able to detect that a stateful session is in progress. Cookie Proxy sends this value in the response to the user agent when it receives a `Set-Cookie` header from the origin server and chooses to manage the cookie. This header is also sent when the Cookie Proxy has added a `Cookie` header in the related HTTP request.

- When the value is "`error`", Cookie Proxy has detected a mismatch between the status of the user agent and the Cookie Proxy (i.e. Cookie Proxy has lost the cookies during a particular user agent session).

7. WAP Gateway Responsibilities

The WAP gateway is responsible for delivering state management information between the user agent and the origin server. Header encoding for HTTP state management headers and WAP specific state management headers are defined in [WSP].

8. Cookie Proxy Responsibilities

8.1 Pass Through Cookie Proxy

The Cookie Proxy MUST implement Pass Through Cookie Proxy functionality, i.e. passing the HTTP headers between the user agent and the origin server without interference. If HTTP state management is not implemented in the client user agent, then the actions taken by the Cookie Proxy are undefined in this specification.

8.2 Cookie Management And Storage

The Cookie Proxy MAY be responsible for managing and storing cookies on behalf of user agents. If this functionality is implemented, the `X-Wap-Proxy-Cookie` and `X-Wap-Proxy-Set-Cookie` headers MUST be used for communication between the client and the proxy. The proxy emulates user agent functionality when communicating with origin servers. User agent role in HTTP state management mechanism is defined in RFC2965.

The user agent MAY control the cookie management in the Cookie Proxy with `X-Wap-Proxy-Cookie` header. The Cookie Proxy MUST enforce the following rules when receiving WAP specific HTTP headers from the client (precondition: Cookie Proxy has identified and authenticated the client and chosen to manage cookies on behalf of the user agents)

1. If the choice in `X-Wap-Proxy-Cookie` header equals `cache` or `cache-has-state,` Cookie Proxy MUST cache cookies and send them to the origin server on behalf of the user agent. In addition to this, when Cookie Proxy receives `X-Wap-Proxy-Cookie: cache` header, it MUST discard all the current user agent session -based cookies (i.e. cookies which were sent from the origin server without `Max-Age` –attribute).
2. If the choice in `X-Wap-Proxy-Cookie` header equals `delete,` Cookie Proxy MUST NOT send cookies to the origin server or store any received cookies. In addition to this, Cookie Proxy MUST NOT send any received cookies to the client. Cookie Proxy MUST NOT delete any cookies stored prior to receiving the `delete` header.
3. If the choice in `X-Wap-Proxy-Cookie` header equals `none` or the header is missing from the request, Cookie Proxy MUST act as a Pass Through Cookie Proxy.
4. If the choice in `X-Wap-Proxy-Cookie` header equals `session or session-has-state,` Cookie Proxy MUST include cookies in the requests to the origin servers. If the Cookie Proxy receives a response containing the `Set-Cookie` header from the origin server, it MUST decide the place for cookie storage according to the presence of the `Max-Age` attribute in the `Set-Cookie` header. If `Max-Age` attribute is present, cookie MUST be transmitted to the user agent without interception. Otherwise it MUST be stored by the Cookie Proxy until it receives a subsequent `X-Wap-Proxy-Cookie: session` (or `X-Wap-Proxy-Cookie: cache`) header from the user agent. Similarly to `X-Wap-Proxy-Cookie: cache` header, `X-Wap-Proxy-Cookie: session` effectively indicates that user agent does not have any cookies bound to the current user agent session and thus all stored cookies without `Max-Age` attribute MUST be discarded.

The Cookie Proxy MUST NOT perform any cookie management, including storage or filtering, without the receipt of an `X-Wap-Proxy-Cookie: cache, X-Wap-Proxy-Cookie: cache-has-state, X-Wap-Proxy-Cookie: session or X-Wap-Proxy-Cookie: session-has-state` header from the user agent, indicating that cookie management is desired.

The Cookie Proxy MUST be prepared to receive `Cookie` headers from the user agent, regardless of the presence of an `X-Wap-Proxy-Cookie` header. If this situation occurs, the Cookie Proxy MUST transmit the state present in the `Cookie` header to the origin server, with the following criteria:

1. If a cookie proxy receives both `Cookie` and `X-Wap-Proxy-Cookie: cache/cache-has-state` or `X-Wap-Proxy-Cookie: session/session-has-state` headers, the Cookie Proxy may append other cookies to the `Cookie` header prior to performing the subsequent HTTP request. In the case where a user agent and a Cookie Proxy have an identical cookie to send, i.e. both cookies have identical values for path, domain and NAME attributes, the cookie MUST be delivered to the origin server as it is specified by the user agent. Cookies MUST be ordered in the `Cookie` header as specified in [RFC2965].

2. If cookie proxy receives both `Cookie` and `X-Wap-Proxy-Cookie: delete` or `X-Wap-Proxy-Cookie: none` headers, it MUST deliver the `cookie` header to the origin server without interception.

Cookie Proxy MUST include `X-Wap-Proxy-Set-Cookie: state` header in the response to the client, if it has received a cookie in the response from the origin server and chosen to manage it or it has sent a `Cookie` header in the associated HTTP request. . This header MUST NOT be sent if neither of the `Cookie` and `Set-Cookie` headers was present in the HTTP request and response, or if the cookie proxy has not cached any cookie information.

Cookie Proxy MUST include `X-Wap-Proxy-Set-Cookie: error` header in the response if user agent sends status information which is conflicting with the status recorded by the Cookie Proxy. This will happen when a user agent sends a request with `X-Wap-Proxy-Cookie: cache-has-state` or `X-Wap-Proxy-Cookie: session-has-state` header, but the Cookie Proxy does not have any cookies in storage for this particular user agent.

Cookie Proxy MUST NOT store the received cookie, if `Set-Cookie` header includes `secure` attribute. If `secure` attribute is present, Cookie Proxy MUST deliver the cookie to the client without interception. This attribute MAY be used by content authors to indicate that a specific cookie contains private or confidential information, and that the preferred storage is in the client.

If a cookie proxy receives an `X-Wap-Proxy-Cookie` header from a client and chooses to manage and store cookies on its behalf, it MUST remove the `X-Wap-Proxy-Cookie` header from the request and thus prevent it from going further to the network. If Cookie Proxy chooses not to manage cookies on behalf of the client, it MUST let the headers pass without interception.

8.3 Associating Cookie Storage With Clients

The Cookie Proxy MUST associate cookies with a single client and prevent another client from gaining access to the cookies. This may be achieved by associating the cookies with an authenticated client identifier. Content authors should be advised that different user agents located in the same client may use the same Cookie Proxy facilities and the same cookie storage.

The Cookie Proxy MUST NOT provide cookie proxy facilities to anonymous clients.

8.4 Managing Proxy Cookies

The Cookie Proxy SHOULD provide a Web application to let the user browse and control the stored cookies.

9. User Agent Responsibilities

9.1 HTTP State Management

The user agent MUST implement HTTP State Management, as specified in [RFC2965]. User agents with non-conforming implementations (i.e. no support) have undefined semantics.

In order to ensure support for an interoperable set of directives for cookie expiration, user agents MUST support at minimum both the Max-Age and Expires attributes of the Set-Cookie response header. Note that [RFC2965] does not include Expires as a Set-Cookie attribute, but mentions that the Expires attribute may be supported for compatibility with Netscape's cookie implementation. It is included here as a requirement since its use is common practice on the Web.

User agents MUST use the HTTP Date header, if available, when cookie freshness lifetime is calculated based on the Expires attribute.

When available, user agents SHOULD support use of a network-based time source, to ensure correct cookie/cache expiration in the event that the HTTP Date header is not provided in a response.

User agents MUST support a minimum cookie size of 2048 octets each, including the fully-qualified host name, cookie name, and cookie data.

User agents MUST support a minimum of 4 cookies.

User agents MUST support a cookie storage that is capable to store the minimal required number of cookies at the required minimal size.

The behaviour of user agents MUST be consistent each time maximum cookie limitations are exceeded, e.g. discarding the oldest cookie or applying a max cookies per domain limit.

9.2 Cookie Proxy Management

Support for use of Cookie Proxy functionality in the user agent is optional. User agent MAY include WAP Specific HTTP State Management Headers in requests to utilize Cookie Proxy facilities.

If Cookie Proxy functionality is supported, end-user MUST have an opportunity to elect to use either cookie proxy facilities or their own local cookie management or both.

User agent MUST send `X-Wap-Proxy-Cookie: cache-has-state` header instead of `X-Wap-Proxy-Cookie: cache` and `X-Wap-Proxy-Cookie: session-has-state` header instead of `X-Wap-Proxy-Cookie: session` in case it has received at least one `X-Wap-Proxy-Set-Cookie` header during the ongoing user agent session. When user agent receives `X-Wap-Proxy-Set-Cookie: error` header, it MAY notify the user that inconsistent service behavior might occur. WAP user agents MUST be prepared to receive `Set-Cookie` HTTP headers even when they have requested Cookie Proxy functionality alone, and must act in accordance with [RFC2109] in this situation (e.g., the user agent should make a best effort attempt to manage the cookie (See section 10.2)).

Appendix A. Change History (Informative)

A.1 Approved Version 1.1 History

Reference	Date	Description
WAP-223-HTTPSM-20001213-a	13 Dec 2000	Latest WAP Forum released version
OMA-TS-HTTPSM-V1_1	31 Mar 2008	Status changed to Approved by TP OMA-TP-2008-0090-INP_Browsing_V2_3_ERP_for_Final_Approval

Appendix B. Static Conformance Requirements (Normative)

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

B.1 SCR for User Agent Features

Item	Function	Reference	Status	Requirement
HSM_C001	User agent support for HTTP State Management Mechanism	9.1	M	
HSM_C002	User agent support at minimum both the Max-Age and Expires attributes of the Set-Cookie response header	9.1	M	
HSM_C003	User agent support for the HTTP Date header, if available, when cookie freshness lifetime is calculated based on the Expires attribute.	9.1	M	
HSM_C004	User agent support for a network-based time source to ensure correct cookie/cache expiration	9.1	O	
HSM_C005	User agent support for a minimum cookie size of 2048 octets each	9.1	M	
HSM_C006	User agent support for a minimum of 4 cookies	9.1	M	
HSM_C007	User agent support for storage of the minimal required number of cookies at of the required minimal size.	9.1	M	
HSM_C008	User agent support for consistent behaviour when the maximum cookie limitation is exceeded	9.1	M	
HSM_C009	User agent support for use of Cookie Proxy functionality	9.2	O	
HSM_C010	User agent support for WAP specific HTTP State management headers	9.2	C:HSM_C004	

B.2 SCR for Cookie Proxy Features

Item	Function	Reference	Status	Requirement
HSM_S001	Cookie Proxy support for passing of HTTP headers between the user agent and the origin server without interference.	8.1	M	
HSM_S002	Cookie Proxy support for Cookie Management and Storage	8.2	O	

Item	Function	Reference	Status	Requirement
	functionality.			
HSM_S003	Cookie Proxy support for user agent role in HTTP State Management Mechanism.	8.2	C:HSM_S002	
HSM_S004	Cookie Proxy support for WAP specific HTTP State Management headers and mechanisms.	8.2	C:HSM_S002	
HSM_S005	Cookie Proxy does not store the cookie if origin server includes `secure` attribute in `Set-Cookie` header.	8.2	C:HSM_S002	
HSM_S006	Cookie Proxy associates HTTP state with a particular client, and does not provide cookie management or storage for anonymous clients.	8.3	C:HSM_S002	
HSM_S007	Cookie Proxy support for WAP HTTP State Management user interface	8.4	O	