Wireless Application Environment Specification

Approved Version 2.2 – 20 Oct 2006

Open Mobile Alliance

OMA-WAP-WAESpec-V2_2-20061020-A

Continues the Technical Activities
Originated in the WAP Forum

Contents

Figures

Tables

1. Scope

Wireless Application Protocol (WAP) is a result of continuous work to define an industry wide specification for developing applications that operate over wireless communication networks. The scope for the WAP Forum is to define a set of specifications to be used by service applications. 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.

Wireless Application Environment (WAE) is part of the WAP Forum's effort to specify an application framework for wireless terminals such as mobile phones, pagers, and PDAs. The framework extends and leverages other WAP technologies, including session, transport and security protocols, as well as other Internet technologies such as XML, URLs, scripting, and various media types. The effort enables operators, manufacturers, and content developers to meet the challenges in building advanced and differentiated services and implementations in a fast and flexible manner.

This document provides a general overview of the overall WAE version 2.2 architecture and builds on the convergence with the Internet achieved in the initial WAE convergence releases, WAE versions 2.0 and 2.1. This specification also represents the root document of the Wireless Application Environment (WAE) version 2.2 specification hierarchy, i.e., the normative document hierarchy. The document represents the core of the WAE specifications, from which additional WAE specification documents have evolved. For this reason, chapters or sections in this document reference all other WAE specifications and certain sections may be moved to other specifications in the future. For additional information on the WAP architecture, refer to "*Wireless Application Protocol Architecture Specification"* [WAPArch].

2. References

2.1. Normative References

[CacheMod] "WAP Caching Model", Open Mobile Alliance™. WAP-120-UACach. URL:http://www.openmobilealliance.org/

[CacheOp] "WAP Cache Operation", Open Mobile Alliance™. WAP-175-CacheOp. URL:http://www.openmobilealliance.org/

[CryptoLib] "WMLScript Crypto Library Specification", Open Mobile Alliance™. WAP-161-WMLScriptCrypto. URL:http://www.openmobilealliance.org/

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

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

[EFI] "External Functionality Interface Framework Version 1.1", Open Mobile Alliance™. OMA-WAP-EFI-V1_1. URL:http://www.openmobilealliance.org/

[ESMP] "ECMAScript Mobile Profile", Open Mobile Alliance™. OMA-WAP-ESMP-V1_0. URL:http://www.openmobilealliance.org/

[ESMPCrypto] "ECMAScript Crypto", Open Mobile Alliance™. WAP-283-ECMACR. URL:http://www.openmobilealliance.org/

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

[HTTPBasicAuth] "HTTP Authentication: Basic and Digest Access Authentication", J. Franks et al., June 1999. URL:http://www.ietf.org/rfc/rfc2617.txt

[HTTPSM] "HTTP State Management Specification", Open Mobile Alliance™. WAP-223-HTTPSM. URL:http://www. openmobilealliance.org/

[IOPProc] "OMA Interoperability Policy and Process". Open Mobile Alliance™. OMA-IOP-Process-v1_1. URL:http//www.openmobilealliance.org/

[ISO10646] "Information Technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane", ISO/IEC 10646-1:2000.

[MMSCTR] "MMS Client Transactions". Open Mobile Alliance™. OMA-MMS—CTR-V1_2. URL:http://www.openmobilealliance.org/

[PICT] "WAP Pictogram, V1.1", Open Mobile Alliance™. OMA-WAP-TS-Pictogram-V1_1. URL:http://www.openmobilealliance.org/

[PSTOR] "WAP WAG Persistent Storage Interface", Open Mobile Alliance™. WAP-301-PSTOR. URL:http://www.openmobilealliance.org/

[PushMessage] "WAP Push Message", Open Mobile Alliance™. WAP-251-PushMessage. URL:http://www.openmobilealliance.org/

[PushOTA] "WAP Push OTA Protocol", Open Mobile Alliance™. WAP-235-PushOTA. URL:http://www.openmobilealliance.org/

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

[RFC2387] "The MIME Multipart/Related Content-type", E. Levinson, August 1998. URL:http://www.ietf.org/rfc/rfc2387.txt

[RFC2388] "Returning Values from Forms: multipart/form-data", L. Masinter, August 1998. URL:http://www.ietf.org/rfc/rfc2388.txt

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, et al., August 1998. URL:http://www.ietf.org/rfc/rfc2396.txt

[RFC2817] "Upgrading to TLS Within HTTP/1.1", RFC2817, R. Khare et al., May 2000. URL:http://www.ietf.org/rfc/rfc2817.txt

[RFC2818] "HTTP Over TLS", RFC2818, E. Rescorla, May 2000. URL:http://www.ietf.org/rfc/rfc2818.txt

[RFC2828] "Internet Security Glossary", RFC2828, R. Shirey, May 2000. URL:http://www.etf.org/rfc/rfc2828.txt

[RFC3023] "XML Media Types", M. Murata et al., January 2001. URL:http://www.ietf.org/rfc/rfc3023.txt

[RFC3236] "The 'application/xhtml+xml' Media Type. M. Baker, P. Stark." RFC3236. January 2002. URL:http://www.ietf.org/rfc/rfc3236.txt

[ServiceInd] "Service Indication", Open Mobile Alliance™. WAP-167-ServiceInd. URL:http://www.openmobilealliance.org/

[UNICODE] "The Unicode Standard: Version 2.0", The Unicode Consortium, Addison-Wesley Developers Press, 1996. URL:http://www.unicode.org/

[VCAL] "vCalendar - the Electronic Calendaring and Scheduling Format", version 1.0, The Internet Mail Consortium (IMC), September 18, 1996, URL:http://www.imc.org/pdi/vcal-10.doc

[VCARD] "vCard - The Electronic Business Card", version 2.1,The Internet Mail Consortium (IMC), September 18, 1996, URL:http://www.imc.org/pdi/vcard-21.doc

[WAEMedia] "WAE Defined Media Type", Open Mobile Alliance™. WAP-237-WAEMT. URL:http://www.openmobilealliance.org/

[WAPTLS] "WAP TLS Profile and Tunneling", Open Mobile Alliance™. WAP-219-TLS. URL:http://www.openmobilealliance.org/

[WBXML] "WAP Binary XML Content Format", Open Mobile Alliance™. WAP-192-WBXML. URL:http://www.openmobilealliance.org/

[WCSS] "Wireless CSS version 1.1", Open Mobile Alliance™. OMA-WAP-WCSS-V1_1. URL:http://www.openmobilealliance.org/

[WDP] "Wireless Datagram Protocol", Open Mobile Alliance™. WAP-259-WDP. URL:http://www.openmobilealliance.org/

[W-HTTP] "WAP Wireless Profiled HTTP", Open Mobile Alliance™. WAP-229-HTTP. URL:http://www.openmobilealliance.org/

[WML1] "Wireless Markup Language Version 1.3", Open Mobile Alliance™. WAP-191-WML. URL:http://www.openmobilealliance.org/

[WML2] "Wireless Markup Language", Open Mobile Alliance™. WAP-238-WML. URL:http://www.openmobilealliance.org/

[WMLScript] "WMLScript Language Specification", Open Mobile Alliance™. WAP-193-WMLS. URL:http://www.openmobilealliance.org/

[WMLStdLib] "WMLScript Standard Libraries Specification", Open Mobile Alliance™. WAP-194-WMLSL. URL:http://www.openmobilealliance.org/

[WTAI] "Wireless Telephony Application Interface Specification", Open Mobile Alliance™. WAP-268-WTAI. URL:http://www. openmobilealliance.org/

[XHTMLMP] "XHTML Mobile Profile 1.1", Open Mobile Alliance™. OMA-WAP—XHTMLMP-V1_1. URL:http://www.openmobilealliance.org/

[WSP] "Wireless Session Protocol", Open Mobile Alliance™. WAP-230-WSP. URL:http://www.openmobilealliance.org/

[WTLS] "Wireless Transport Layer Security", Open Mobile Alliance™. WAP-261-WTLS. URL:http://www.openmobilealliance.org

[WP-TCP] "WAP Wireless Profiled TCP", Open Mobile Alliance™. WAP-225-TCP. URL:http://www.openmobilealliance.org/

[XHTMLBasic] "XHTML™ Basic", W3C Recommendation, Mark Baker et al., 19 December 2000. URL: http://www.w3.org/TR/2000/REC-xhtml-basic-20001219.

[XML] "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, Tim Bray et al., 6 October 2000. URL: http://www.w3.org/TR/2000/REC-xml-20001006

2.2. Informative References

[CC/PP] "Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation", W3C Note, Franklin Reynolds, Johan Hjelm, Spencer Dawkins, and Sandeep Singhal, 27 July 1999. URL;http://www.w3.org/1999/07/NOTE-CCPP-19990727/

[CCPPex] "CC/PP exchange protocol based on HTTP Extension Framework", W3C Note, Hidetaka Ohto and Johan Hjelm, 24 June 1999. URL:http://www.w3.org/1999/06/NOTE-CCPPexchange-19990624.

[CSS2] "Cascading Style Sheets, level 2 (CSS2) Specification", W3C Recommendation, Bert Bos et al., 12 May 1998.
URL:http://www.w3.org/TR/1998/REC-CSS2-19980512.

[CSSMP] "CSS Mobile Profile 1.0", W3C Candidate Recommendation, Ted Wugofski. Doug Dominiak, Peter Stark, 24 October 2001.
URL:http://www.w3.org/TR/2001/CR-css-mobile-20011024.

[DLARCH] "Download Architecture". Open Mobile Alliance™. OMA-Download-ARCH-v1_0. URL:http://www.openmobilealliance.org

[DRMCF] "DRM Content Format", Open Mobile Alliance™. OMA-Download-DRMCF-v1_0.

URL:http://www.openmobilealliance.org/

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

[E2ESec] "WAP Transport Layer End-to-end Security Specification", Open Mobile Alliance™. WAP-187-TLE2E. URL:http://www.openmobilealliance.org/

[ECMA327] Standard ECMA-327, "ECMAScript 3 rd Edition Compact Profile", ECMA, June 2001, URL:

ftp://ftp.ecma.ch/ecma-st/Ecma-327.pdf

[ECMAScript] Standard ECMA-262: "ECMAScript Language Specification – Edition 3", ECMA, December 1999. URL: ftp://ftp.ecma.ch/ecma-st/Ecma-262.pdf

[ESMPCrypto] "ECMAScript Crypto", Open Mobile Alliance™. WAP-283-ECMACR. URL:http://www.openmobilealliance.org/

[HTML4] "HTML 4.01 Specification", W3C Recommendation, Dave Raggett, Arnaud Le Hors, and Ian Jacobs, 24 December 1999. URL:http://www.w3.org/TR/1999/REC-html401-19991224

[JavaScript] "JavaScript: The Definitive Guide", David Flanagan, O'Reilly & Associates, Inc., 1997

[MMSArch] "MMS Architecture Overview". Open Mobile Alliance™. OMA-MMS-Arch-V1_2. URL://http://www.openmobilealliance.org/

[MMSEncaps] "MMS Encapsulation Protocol". Open Mobile Alliance™. OMA-MMS-Encapsulation-V1_2. URL:http://www.openmobilealliance.org/

[ProvArch] "WAP Provisioning Architecture Overview", Open Mobile Alliance™. WAP-182-ProvArch. URL:http://www.openmobilealliance.org/

[ProvCont] "WAP Provisioning Content Specification", Open Mobile Alliance™. WAP-183-ProvCont. URL:http://www.openmobilealliance.org/

[ProvUAB] "WAP Provisioning User Agent Behaviour Specification", Open Mobile Alliance™. WAP-185-ProvUAB. URL:http://www.openmobilealliance.org/

[PAP] "Push Access Protocol", Open Mobile Alliance™. WAP-247-PAP. URL:http://www.openmobilealliance.org/

[PPGService] "Push Proxy Gateway Service", Open Mobile Alliance™. WAP-249-PPGService. URL:http://www.openmobilealliance.org/

[RDF] "Resource Description Framework (RDF) Model and Syntax Specification," W3C Recommendation, O. Lassila, R Swick, February 1999. URL:http://www.w3.org/TR/1999/REC-rdf-syntax-19990222

[RFC2045] "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", N. Freed, et al., November 1996. URL:http://www.ietf.org/rfc/rfc2045.txt

[RFC2046] "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", N. Freed et al., November 1996. URL:http://www.ietf.org/rfc/rfc2046.txt

[RFC2047] "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", K. Moore, November 1996. URL:http://www.ietf.org/rfc/rfc2047.txt

[RFC2318] "The text/css Media Type", H.Lie et al., March 1998. URL://www.ietf.org/rfc/rfc2318.txt

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

[ServiceLoad] "Service Loading", Open Mobile Alliance™. WAP-168-ServiceLoad. URL:http://www.openmobilealliance.org/

[SyncML] "SyncML Data Sync Protocol", Open Mobile Alliance™. OMA-SyncML-DataSyncProtocol-V1_1_2. URL:http://www.openmobilealliance.org

[UAPROF] "WAG UAProf", Open Mobile Alliance™. WAP-248-UAPROF. URL:http://www.openmobilealliance.org/

[WAE20] "Wireless Application Environment Specification – version 2.0", Open Mobile Alliance™. WAP-236-WAESpec. URL:http://www.openmobilealliance.org/

[WAE21] "Wireless Application Environment Specification – version 2.1", Open Mobile Alliance™. OMA-WAP-WAESpec-V2_1. URL:http://www.openmobilealliance.org/

[WAEOverview1] "Wireless Application Environment Overview", Open Mobile Alliance™. WAP-195-WAEOverview. URL:http://www.openmobilealliance.org/

[WAESpec1] "Wireless Application Environment Specification", Open Mobile Alliance™. WAP-190-WAESpec. URL:http://www.openmobilealliance.org/

[WAP1] "WAP June 2000 (WAP 1.2.1) Conformance Release", Open Mobile Alliance™. URL:http://www.openmobilealliance.org/

[WAPArch] "WAP Architecture Specification", Open Mobile Alliance™. WAP-210-WAPArch. URL:http://www.openmobilealliance.org/

[WMLTR] "WML Transformations", Open Mobile Alliance™. WAP-244-WMLTR. URL:http://www.openmobilealliance.org/

[WTA] "Wireless Telephony Application Specification", Open Mobile Alliance™. WAP-266-WTA. URL:http://www.openmobilealliance.org/

[WTP] "Wireless Transaction Protocol Specification", Open Mobile Alliance™. WAP-224-WTP. URL:http://www.openmobilealliance.org/

[XHTML] "XHTML™ 1.1 - Module-based XHTML", W3C Proposed Recommendation, 6 April 2001. URL:http://www.w3.org/TR/2001/PR-xhtml11-20010406

[XHTMLMP10] "XHTML Mobile Profile", Open Mobile Alliance™. WAP-277-XHTMLMP. URL:http://www.openmobilealliance.org/

[XHTMLMod] "Modularization of XHTML™", W3C Recommendation, Murray Altheim et al., 10 April 2001. URL:http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410

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, except “Scope”, “WAE Documentation” and “WAE Version 2 Introduction” are normative, unless they are explicitly indicated to be informative.

3.2. Definitions

Author	an author is a person or program that writes or generates WML, WMLScript or other content.
Bytecode	content encoding where the content is typically a set of low-level opcodes (i.e., instructions) and operands for a targeted hardware (or virtual) machine.
Client	a device (or application) that initiates a request for connection with a server.
Client Server Communication	communication between a client and a server. Typically the server performs a task (such as generating content) on behalf of the client. Results of the task are usually sent back to the client (e.g., generated content).
Content	synonym for data objects.
Content Encoding	when used as a verb, content encoding indicates the act of converting a data object from one format to another. Typically the resulting format requires less physical space than the original, is easier to process or store, and/or is encrypted. When used as a noun, content encoding specifies a particular format or encoding standard or process.
Content Format (or Format)	actual representation of content.
Deprecated	A deprecated feature (e.g. specification, element or attribute) is one that has been outdated by a newer feature. Deprecated features are defined in the specification and are clearly marked as deprecated. Deprecated features may become obsolete in a future version.
Device	a network entity that is capable of sending and receiving packets of information and has a unique device address. A device can act as both a client and a server within a given context or across multiple contexts. For example, a device can service a number of clients (as a server) while being a client to another server.
ECMAScript	a scripting language produced and managed by the European Computer Manufacturers Association (ECMA) which provides a common scripting language for the computer industry.
Gateway (or WAP Gateway)	a server which acts as an intermediary for some other server. A gateway performs protocol transformation as well as encoding/decoding content.
Host Object	ECMAScript objects provided by the user agent for the purpose of interaction with the loaded document.
Hybrid User Agent	a single user agent that supports multiple markup languages, e.g., XHTML Mobile Profile and WML1.
Hypermedia Transfer	The hypermedia transfer services provides for the transfer of self-describing hypermedia resources. The combination of WSP (Wireless Session Protocol) [WSP] and WTP (Wireless Transaction Protocol) [WTP] provide the hypermedia transfer service over secure and non-secure datagram transports over datagram-based protocol stack. The HTTP (Hypertext Transfer Protocol) [HTTP/1.1] provides the hypermedia transfer service over secure and non-secure connection-oriented transports over connection-oriented protocol stack.
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.
Media type	a MIME media type or an identifier for a given data type.

MIDlet Download	refers to the Java 2 Micro Edition (J2ME) Mobile Internet Device Profile Over-The-Air User Initiated Provisioning Recommended Practices. This defines the method of downloading and managing the lifecycle of J2ME applications, referred to as MIDlets.
WAP Proxy	an intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, with possible translation, to other servers. It may provide functions of protocol enhancement, transcoding or any number of other optimisation or transformation functions and may be associated with any gateways, proxies or servers being used in the deployment architecture. WAP gateway is one of the optional functionalities of WAP proxy.
PC Browser	an existing Web browser that already supports text/html.
Push Initiator	the entity that originates push content and submits it to the push framework for delivery to a user agent on a client.
Push Proxy Gateway (PPG)	a proxy gateway that provides push proxy services.
Resource	a network data object or service that can be identified by a URL. Resources may be available in multiple representations (e.g., multiple languages, data formats, size, and resolutions) or vary in other ways.
Server	a device (or application) that passively waits for connection requests from one or more clients. A server may accept or reject a connection request from a client.
Terminal	a device typically used by a user to request and receive information. Also called a mobile terminal or mobile station.
User	a user is a person who interacts with a User Agent to access a resource.
WAE User Agent (or User Agent)	a User Agent is any software or device that interprets markup and scripting languages or other content. This may include textual browsers, voice browsers, search engines, etc.
WAE version	the version of the WAE User Agent. The version of the WAE User Agent may be uniquely identifiable by the WAP version, e.g. WAP version 1.1 contains WAE version 1.1, or it may be a feature of the WAP version in which case the WAE versioning mechanisms are used to determine the WAE User Agent version.
WAP1	WAP Version 1, nominally the latest point release, e.g. WAP V1.2.1, unless otherwise noted.
WAP2	WAP Version 2. When used as a prefix, it indicates that something is compliant to the WAP Version 2 conformance requirements, e.g., a WAP2 client is a client that fulfils all the requirements for a user agent of WAP Version 2. WAP2 content is content with a media type specified in WAP Version 2.
WML	The Wireless Markup Language is a hypertext markup language used to represent information for delivery to a narrowband device, e.g., a phone.
WMLScript	A scripting language used to program the mobile device. WMLScript is an extended subset of the ECMAScript scripting language.
WSP Server	An origin server which uses WSP instead of HTTP.
WTA User Agent	The WTA user agent extends a WAE User Agent with capabilities for interfacing with mobile network services, that interacts with the defined WTA framework components.
WTA-WML	The WTA-WML has the same capabilities as WML but extends WML with WTA specific features for handling event parameters.
XML	the Extensible Markup Language is a World Wide Web Consortium (W3C) standard for Internet markup languages, of which WML is one such language. XML is a restricted subset of SGML.
vCalendar	Internet Mail Consortium (IMC) electronic calendar record.
vCard	Internet Mail Consortium (IMC) electronic business card

3.3. Abbreviations

API	Application Programming Interface
BNF	Backus-Naur Form

CGI	Common Gateway Interface
CPI	Capability and Preference Information
CSS	Cascading Style Sheets
DRM	Digital Rights Management
ECMA	European Computer Manufacturer Association
EFI	External Functionality Interface
ESMP	ECMAScript Mobile Profile
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol [HTTP/1.1]
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IMC	Internet Mail Consortium
ISO	International Standards Organisation
LSB	Least Significant Bit
MSB	Most Significant Bit
OTA	Over The Air
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
PPG	Push Proxy Gateway
RFC	Request For Comments
SIM	Subscriber Identification Module
TPS	Trusted Provisioning Server
UAProf	User Agent Profile
URI	Uniform Resource Identifier [RFC2396]
URL	Uniform Resource Locator [RFC2396]
UCS-4	Universal Character Set coded in 4 octets
W3C	World Wide Web Consortium
W-HTTP	Wireless Profiled HTTP
WML	Wireless Markup Language (WML1 or WML2)
WML1	Wireless Markup Language Version 1.3
WML2	Wireless Markup Language Version 2.0
WWW	World Wide Web
WSP	Wireless Session Protocol
WTP	Wireless Transaction Protocol
WDP	Wireless Datagram Protocol
WAP	Wireless Application Protocol
WAE	Wireless Application Environment. Unless otherwise stated it refers to this version.

WAE2	Wireless Application Environment version 2 (includes any point releases, e.g. 2.0)
WAE20	Wireless Application Environment version 2.0 [WAE20]
WAE21	Wireless Application Environment version 2.1 [WAE21]
WCSS	Wireless Cascading Style Sheets [WCSS] (formerly known as WAP CSS)
WTA	Wireless Telephony Application
WTAI	Wireless Telephony Application Interface
WBMP	Wireless BitMaP
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

3.4. Acknowledgements

Sun, Sun Microsystems, the Sun Logo, Java, J2ME are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

4. WAE Documentation

This section is informative.

4.1. The WAE Document Suite

The following lists the document suite of the Wireless Application Environment.

- Wireless Application Environment Specification (This document)
- XHTML Mobile Profile [XHTMLMP]
- ECMAScript Mobile Profile [ESMP]
- Wireless Markup Language Specification Version 1.3 [WML1]
- Wireless Markup Language Specification Version 2.0 [WML2]
- Wireless CSS [WCSS]
- WAE Defined Media Types [WAEMedia]
- WAP Binary XML Content Format [WBXML]
- WMLScript Language Specification [WMLScript]
- WMLScript Standard Libraries Specification [WMLStdLib]
- WAP Caching Model [CacheMod]
- HTTP State Management Specification [HTTPSM]
- WML Transformations [WMLTR]

The following picture illustrates the WAE Specification structure.

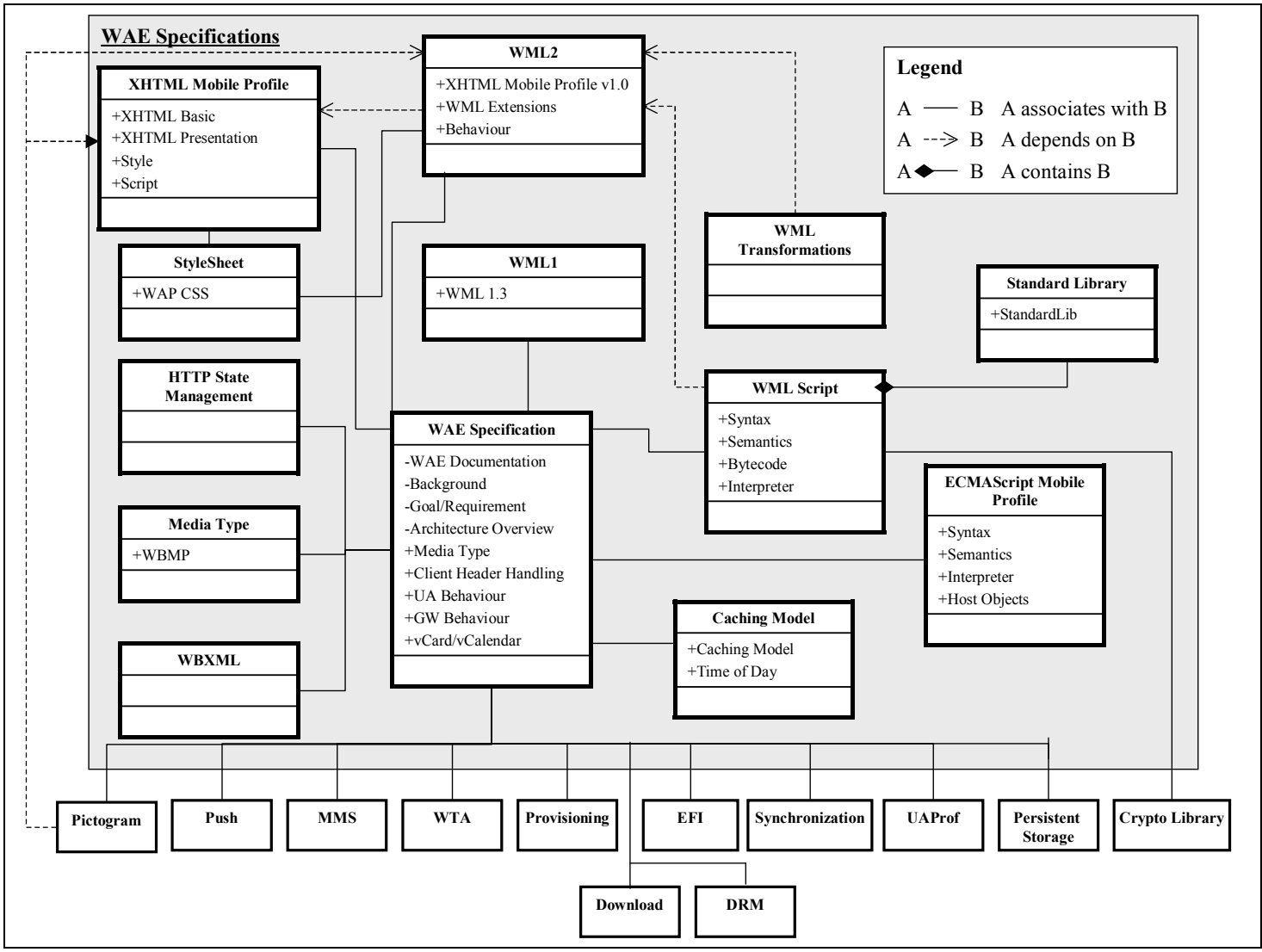


Figure 4-1: WAE Specification Structure

5. WAE Version 2 Introduction

5.1. Background

This section is informative.

WAE version 2 is a general-purpose application environment where operators and service providers can build their applications and services for a wide variety of wireless platforms.

WAE, since WAP version 1[WAP1], has been built fundamentally on the World Wide Web (WWW) technologies with additional optimisation for wireless devices and networks, taking account of their constraints such as small displays, limited user-input facilities, narrow bandwidth, high latency, limited memory, limited CPU, etc.

As the range of devices accessing, and being members of, the Internet has increased, with their spread of capabilities in terms of user interface, presentational capabilities etc., core Internet/WWW technologies have evolved and continue to evolve to satisfy the requirements of these devices and the new use-cases their development enables.

The developments in base Internet technology allows WAE version 2 to converge with, or adopt, more standard Internet technologies than was possible in WAP version 1. This tight convergence will maximise the ease with which WAP-enabled systems can use existing Internet/WWW applications and content. It will also simplify WAP's adoption of new evolving services.

Thus, WAE version 2 has evolved WAE version 1[WAEOverview1, WAESpec1] by adopting the most appropriate Internet technologies available and to supplement these only where necessary with additional WAP specific technology to enable a general-purpose application centric environment suitable wireless centric applications and services.

WAE version 2.0 [WAE20] was the initial release of WAE version 2 and this version, WAE version 2.1 [WAE21], addresses further convergence through adoption of additional Internet and related technology, and WAE version 2.2 introduces a significant minimum level of support for presentational control when the Wireless Cascading Style Sheets specification V1.1 [WCSS] is supported. The specific changes in WAE version 2.2 are indicated.

5.2. Goals and Requirements for WAE version 2

This section is informative.

The goals and requirements for WAE version 2 are stated in this section. The specifications associated with the goals and requirements will be delivered as part of WAE version 2.

5.2.1. Goals

WAE version 2 is an evolution of version 1. The goals are:

- To enable optimised access to applications and services from the current and emerging generations of WAP-enabled wireless devices. These devices are generally assumed to be small, battery operated, have relatively limited memory and CPU power. Their small form factor results in limited display space and restricted input facilities.
  For these reasons, WAP is optimised for efficient use of device resources. Furthermore, WAP architecture and protocols maximise the user experience given the limited display space and restricted input facilities.

- To create system and application services that are sensitive to the limited bandwidth requirements of wireless devices, including but not limited to response time, round trip service cost, and cost of bandwidth resources.

- To enhance and extend the web-centric application model to support wider varieties of wireless devices and networks than is possible with standard web technology, and to provide a richer document rendering environment that is possible with WAP version 1.

- To take advantage of the work of other standards bodies and the legacy of WAP version 1 technologies and to converge with those technologies.
  Furthermore, work done in the WAP Forum has been fed back into other standards bodies to achieve successful convergence.
  Specific standards bodies with which WAE version 2 has converged with, are:

  - W3C
  - ECMA (introduced in WAE version 2.1)
  - IETF
  - ISO

- To allow the creation of Man Machine Interfaces (MMIs) with maximum flexibility and ability for the vendor to enhance the user experience. In this way vendors can provide distinct user interfaces
- To have security that is consistent and interoperable with Internet security models
- To enable a secure interface to external functionality, i.e., functions outside the basic browser such as additional hardware (GPS, smartcards etc.) or software (additional software etc.)
- To ensure the privacy of the user's personal information such as user location, credit card information, address and phone number.
  Many types of Internet content, including WAP content, are more exposed to user privacy issues during exchanges between origin servers and clients. WAP protects the transfer of personal information and ensures an adequate degree of service authentication.
- To target common character codes for international use. This includes international symbols and pictogram sets for end users, and local-use character encoding for content developers.
- To leverage mobility functions of devices and networks. Mobile devices and networks can offer services that cannot be as easily offered on fixed-line networks. WAP has been leveraged for such services. In particular, one class of services that would enrich WAP is the class of services that are fertilised by the location information of the users.
- To enable the personalisation and customisation of the device, the content delivered to it, and the presentation of the content. It is possible for such personalisation and customisation to be performed by a variety of parties, including end-users, operators, service providers, infrastructure vendors and device vendors.

5.2.2. Requirements

- Backward compatibility – Special attention has been paid to ensure a smooth transition of the standard. Potential discontinuities have been identified, and handled by creating a feasible migration path, allowing to preserve the investment in services built on WAE version 1, i.e. WML1[WML1] and WMLScript[WMLScript]. See Section 6.2.3
- Convergence with W3C Specifications
  - XHTML[XHTML] defined by W3C is a reformulation of HTML in XML[XML]. By including XHTML in the WAP specifications, the WAP Forum will easily incorporate new technologies from the W3C.
  - CSS[CSS2] is a style sheet language to control the presentation of HTML/XML documents. By specifying a suitable subset of CSS for WAP terminals, authors can control the presentation of WAP content.
- Convergence with ECMA's ECMAScript [ECMAScript] Edition 3; the established scripting technology used in today's Internet applications.

  WAE version 2.1 achieves convergence through the adoption of a profile of ECMAScript [ECMAScript] Edition 3. By adopting an appropriate profile of ECMAScript content written for WAE version 2.1 user agents will be usable by established Internet browsers and WAE version 2.1 user agents will be able to process significant amounts of Internet content utilising ECMAScript.
- Prescription of User Agent Behaviour – Processing model will be specified to ensure interoperability. Requirements from other specifications such as WTA and EFI have been taken into consideration. The caching model for User Agents has also been defined.
- Definition of WAP-specific Media types, e.g., WBMP.

- WINA Name and Number Database has been properly maintained, e.g., WXBML Public Document ID, WMLScript Library ID.
- Semantic Level Backward Compatibility – WAP version 1 semantics have been included in WAP version 2, just for ensuring backward compatibility. The following are examples of those semantics of WAE:
  - Deck/Card architecture (See Section 6.2)
  - Variables (See Section 6.2)
  - Script (See Section 6.4)

5.3. WAE Version 2 Components

This section is informative.

Among those components of WAE are XHTML Mobile Profile, WML, WCSS, ECMAScript Mobile Profile [ESMP] (introduced in WAE version 2.1), WMLScript, WBXML, vCard and vCalendar.

- XHTML Mobile Profile[XHTMLMP]
  W3C's HTML [HTML4] is evolving into XHTML and has been modularised [XHTMLMod], thus allowing operators or service providers to build their applications based on the modules necessary for the target platforms or users. A basic set of modules has been defined as XHTML Basic [XHTMLBasic], and can be used as the foundation for an extension to the XHTML family of markup languages. Using this framework, XHTML Mobile Profile is built on top of XHTML Basic with additional extensions.
  - A pure XHTML Basic document is also a valid XHTML Mobile Profile document. The XHTML Mobile Profile browser indicates it accepts both XHTML Basic and XHTML Mobile Profile documents.
- WML
  WML is a markup language based on XML[XML] and is intended for use in specifying content and user interface for narrowband devices, including cellular phones and pagers. WML is designed with the constraints of small narrowband devices in mind. These constraints include:
  - Small display and limited user input facilities
  - Narrowband network connection
  - Limited memory and computational resources

  WML provides the following specific features:
  - WML supports the deck and card architecture: A deck is a transmission unit, consisting of multiple cards; a card is a user interaction unit. This architecture allows the content optimised for small displays to be transmitted without repetition and redundancy over narrow bandwidth links.
  - WML includes navigation and event handling models. By explicitly managing the navigation between cards and decks, WML can offer the user experience optimised for wireless environment. It also includes provisions for event handling in the device, which may be used for navigational purposes or to execute scripts.
  - WML includes support for managing User Agent state, through the use of:
    - ✧ Variables – parameters used to change the characteristics and content of a WML card or deck
    - ✧ History – navigational history, which may be used to facilitate efficient backward navigation
  - WML specifies layout and presentation in an abstract way, thus allowing terminal and device vendors to control the MMI design of their products, enabling them to differentiate their products.

  WML1[WML1] was introduced in the WAP Version 1 WAE platform, focusing on providing wireless specific features as described above. WML2[WML2] is built on top of XHTML Basic with additional wireless extension modules that

provide the WML specific features, thus WML2 provides on the convergence with the existing Internet standard as well as backward compatibility to WML1.

  - A pure XHTML Basic document and a XHTML Mobile Profile document is also a valid WML2 document. The WML2 browser indicates it accepts XHTML Basic, XHTML Mobile Profile and WML2 documents.

- WCSS[WCSS]
  W3C is promoting the use of stylesheets such as CSS [CSS2] to control the presentation of content without sacrificing the device-independence. W3C is also defining CSS Mobile Profile [CSSMP], a minimum set of features which are appropriate for mobile devices. WCSS is defined using a proper subset of CSS based on the W3C's Mobile Profile. WCSS can be used to optimise WML2 presentation as well as XHTMLMP.

- ECMAScript Mobile Profile[ESMP] is a profile of the ECMAScript Language Specification, edition 3 [ECMAScript] based on the ECMAScript 3rd Edition Compact Profile [ECMA327] with the inclusion of both native and commonly used host objects to provide an interoperable and consistent programming environment for applications.

- WMLScript[WMLScript] is a lightweight scripting language, loosely based on ECMAScript[ECMAScript], which provides programmable functionality that can be used in clients with limited capabilities over narrow-band communication links. Scripting enhances the standard browsing and presentation facilities of WML with behavioural capabilities. It can be used to support more advanced UI functions, add intelligence to the client, provide access to the device and its peripheral functionality and reduces the amount of bandwidth needed to send data between the server and the client. WAE defines the library interfaces for the standard set of libraries[WMLStdLib] supported by WMLScript to provide access to the core functionality of a WAP client.

- WBXML[WBXML] is a compact binary representation of the XML. Its format is designed to reduce the transmission size of XML documents with no loss of functionality or semantic information, allowing more effective use of XML data on narrow-band communication channels. The format is designed to preserve the element structure of XML. The binary format encodes the parsed physical form of an XML document, i.e., the structure and content of document entities. Meta-information, including the document type definition and conditional sections, is removed when the document is converted to the binary format. In the WAP version 1 environment, most of XML based document types, including WML1, are encoded into WBXML and support for WBXML is required. However, WBXML encoding for some formats are removed or deprecated in WAP version 2; e.g., WML2 document can not be encoded into WBXML.

- vCard[VCARD] and vCalender[VCAL] are supported for exchanging phone book and calendar data objects. vCard and vCalendar are industry-standards defined by the IMC. vCard and vCalendar data are exchanged using WDP, WSP, or W-HTTP including WAP Push.

Figure 5-1 illustrates components of the WAE User Agent and WAP Application Framework in relationship to the WAP version 2 architecture [WAPArch]. In Figure 5-1, those components which are beyond the WAE scope and have no dependencies on WAE in dark grey and those with dependencies on WAE in the hatched grey. Note: This is not an exhaustive figure.

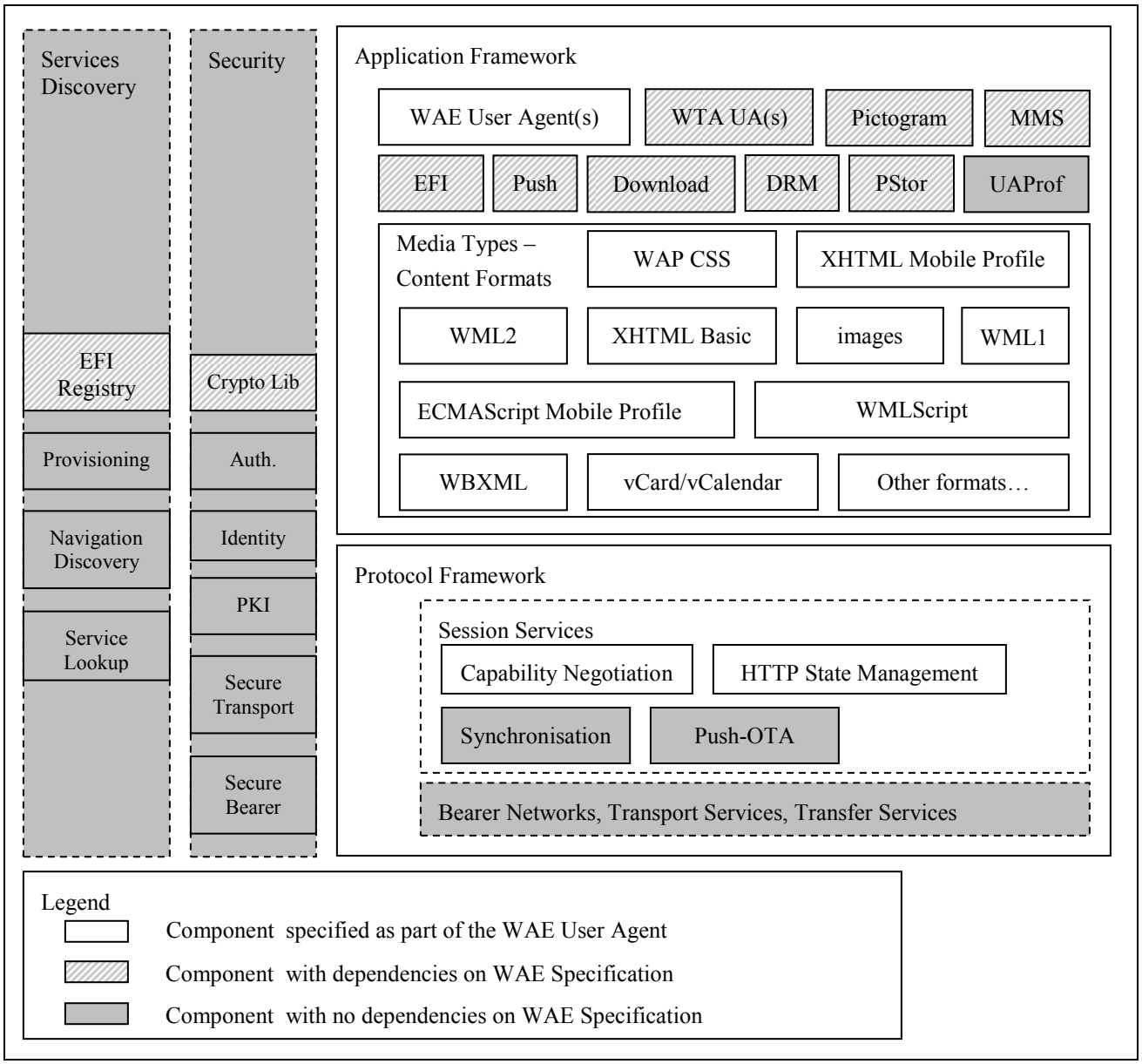


Figure 5-1: WAE Components in relation to the WAP version 2 architecture

WAE assumes the client exists on the WAP/wireless network and the server exists on the Internet. In some configurations, a WAP proxy may exist between them that is responsible for providing gateway and proxy functions for data transferred from and to the mobile client. Additionally a WAP proxy may provide additional services on behalf of the client, e.g. transcoding. The client and the server may communicate directly using the Internet protocol, or via a WAP proxy which takes the responsibility of translating the transfer protocol when the protocol used in the WAP/wireless network is different from the Internet. See [WAPArch] for more detail.

The changes between WAE version 2.1 and WAE version 2.0 are as follows:

- ECMAScript Mobile Profile [ESMP]is added, see section 6.4.2 for more details;
- XHTMLMP [XHTMLMP] and WAE are updated to define the use of ESMP within WAE;

- WMLScript [WMLScript] and WMLScript libraries [WMLStdLib] are deprecated, see section 6.4.1 for more details;
- The OMA Download [DLARCH] feature is introduced, see section 7.1.5 for more details;
- The OMA Digital Rights Management [DRM] feature is introduced, see section 7.3 for more details;
- The Multi-Media Messaging [MMSArch] feature is added; see section 7.4 for more details.

6. WAE Media Types

6.1. Media Types Supported by WAE

This section is normative.

Table 6-1 shows the list of WAE Media Types supported by WAE. A WAE User Agent MUST support the MIME Media Types defined in table 6-1 for the supported device features. A WAE User Agent may support any other media types. A WAE User Agent SHOULD advertise a list of data types that it supports by sending MIME Media Types with the HTTP Accept header. A server may send back content with a MIME media type different from the one the User Agent advertised. Note: File Extension is only a hint as to the application media type.

Data Type	MIME Media Type	File Extension
WML1 textual form	text/vnd.wap.wml	.wml
WML1 binary form	application/vnd.wap.wmlc	.wmlc
WML2	application/vnd.wap.wml+xml (The registration of application/wml+xml is on-going)	.wml
XHTML Basic	application/xhtml+xml [RFC3236] text/html[1]	.xhtml, .xht
XHTML Mobile Profile	application/vnd.wap.xhtml+xml application/xhtml+xml; profile="http://www.wapforum.org/xhtml" [RFC3236] text/html	.xhtml, .xht
Wireless CSS (formerly WAP CSS)	text/css	.css
WMLScript textual form	text/vnd.wap.wmlscript	.wmls
WMLScript binary form	application/vnd.wap.wmlscriptc	.wmlsc
ECMAScript Mobile Profile	text/ecmascript text/javascript	.js
WBXML	application/vnd.wap.wbxml	
WBMP	image/vnd.wap.wbmp	.wbmp
VCard	text/x-vCard	.vcf
vCalendar	text/x-vCalendar	.vcs
WTA-WML textual form	text/x-wap-wta-wml	
WTA-WML binary from	application/x-wap-wta-wmlc	
Textual form of Multipart Messages that are used when the body parts are independent and need to be bundled in a particular order	multipart/mixed	
Binary form of Multipart Messages that are used when the body parts are independent and need to be bundled in a particular order	application/vnd.wap.multipart.mixed	
Textual form of Multipart Message representing objects that are aggregates of related MIME body	multipart/related	

[1] "text/html" MIME media type can be used for the XHTML Mobile Profile content or XHTML Basic content to be delivered to a PC browser.

parts		
Binary form of Multipart Message representing objects that are aggregates of related MIME body parts	application/vnd.wap.multipart.related	
Textual form of Multipart Message that is used when each of the body parts is alternative version of the same information	multipart/alternative	
Binary form of Multipart Message that is used when each of the body parts is alternative version of the same information	application/vnd.wap.multipart.alternative	
Textual form of Multipart Message for returning values from form	multipart/form-data	
Binary form of Multipart Message for returning values from form	application/vnd.wap.multipart.form-data	
Channels in textual form	text/vnd.wap.channel	
Channels in binary form	application/vnd.wap.channelc	
Service Indication in textual form	text/vnd.wap.si	
Service Indication in binary form	application/vnd.wap.sic	
Service Loading in textual format	text/vnd.wap.sl	
Service Loading in binary format	application/vnd.wap.slc	
Cache Operation in textual form	text/vnd.wap.co	
Cache Operation in binary form	application/vnd.wap.coc	
Provisioning Document in textual form	text/vnd.wap.connectivity-xml	
Provisioning Document in binary form	application/vnd.wap.connectivity-wbxml	
MMS Message PDU	application/vnd.wap.mms-message	
Download Descriptor	application/vnd.oma.dd+xml	.dd
DRM Message	application/vnd.oma.drm.message	.dm
DRM Content	application/vnd.oma.drm.content	.dcf
DRM Rights in textual form	application/vnd.oma.drm.rights+xml	.dr
DRM Rights in binary form	application/vnd.oma.drm.rights+wbxml	.drc

Table 6-1: Mime Media Types Supported by WAE

6.1.1. Conformance Rules

Conformance: A User Agent MUST use the MIME media type as one component when determining the content data type of a document. (WAESpec-MT-C-001)

Conformance: A User Agent MUST support the WAE MIME Media types defined in Table 6-1 for all supported features. (WAESpec-MT-C-002)

Conformance: A Proxy or Server MUST support the WAE MIME Media types defined in Table 6-1 for all supported features. (WAESpec-MT-S-001)

Conformance: A Proxy MUST appear transparent to all requests and responses from user agents for which no specific action is required based on the MIME Media types defined in Table 6-1 or the user agent preferences, other than the fetching on behalf of, and delivering the resulting content to, the user agent. (WAPSpec-MT-S-002)

6.2. Markup Languages in WAE2

6.2.1. XHTML Mobile Profile

This section is informative.

XHTML Mobile Profile is a language which extends the syntax and semantics of XHTML Basic[XHTMLBasic] and CSS Mobile Profile[CSSMP] with support for internal CSS [WCSS] and some presentation elements from XHTML1.1.

XHTML is the reformulation of HTML 4.0 as an application of XML. XHTML Basic is a subset of XHTML 1.1 that includes the minimal set of modules required to be an XHTML Family document type, and in addition it includes images, forms, and basic tables. It is designed for Web clients that do not support the full set of XHTML features, for example, Web clients such as mobile phones, PDAs, pagers, and set-top boxes. The document type definition is implemented using XHTML modules as defined in *"Modularization of XHTML*" [XHTMLMod].

XHTMLMP should be used for new services/applications based on WAE2.

XHTML Mobile Profile version 1.1 [XHTMLMP] enhances XHTML Mobile Profile version 1.0 [XHTMLMP10] by adding support for the script and intrinsic event modules from XHTML Modularisation [XHTMLMod]. WAE Version 2.1 user agents must support XHTML Mobile Profile version 1.1. [XHTMLMP]. See section 6.4 for details of related scripting language support.

6.2.2. WML2

This section is informative.

WML2 is a strict superset of the XHTML Mobile Profile version 1.0 [XHTMLMP10]. It adds extensions to the XHTML Mobile Profile for WML1 compatibility, creating a document type that is used for achieving backward compatibility for with services/applications written using WML1. WML2 is deprecated in WAE2; XHTMLMP should be used for new services/applications. WML2 should only be sent to a WAE User Agent as the result of WML1 transformation. WML2 should not be used to create new services/applications.

WML2 is a language which extends the syntax and semantics of XHTMLMP version 1.0 [XHTMLMP10] and Wireless CSS [WCSS] with the unique semantics of WML1, optimised for specifying presentation and user interaction on limited capability devices such as mobile phones and other wireless mobile terminals.

In addition to XHTMLMP10, WML2 includes "*WML Extension Module(s)*", which supports WML1 features described in Section 6.2.3. WML2 also includes those XHTML features, beyond XHTML Basic that represent similar features to those expressed previously in WML1. Where there is overlap between WML1 and XHTML, preference has been given to the XHTML expression of a feature.

WAE 2.1 user agents may support WML2. See section 6.4 for details of related scripting language support

6.2.3. WML 1

This section is informative.

WML1 is a markup language based on XML and is intended for use in specifying content and user interface for narrowband devices, including cellular phones and pagers.

WML1 is included in WAE2 to ensure backwards compatibility for services/applications written using WML1. WML1 is deprecated in WAE2; XHTMLMP should be used for new services/applications.

WML1 is designed with the constraints of small narrowband devices in mind. These constraints include:

- Small display and limited user input facilities
- Narrowband network connection
- Limited memory and computational resources

WML1 provides the following specific features, which are also included in WML2.

- *User Input*
  Along with XHTML forms support, WML supports several elements to solicit user input. WML includes a *text entry*

control that supports text and password entry. Text entry fields can be masked to prevent the end user from entering incorrect character types. WML also supports client-side validation by allowing the author to invoke scripts at appropriate times to check the user's input. WML includes an *option selection* control that allows the author to present the user with a list of options that can set data, navigate among cards, or invoke scripts. WML supports both single and multiple option selections.

- *Card and Deck Structure*
  In addition to the standard HTML model of *head and body*, information in WML may be organised into a collection of *cards* and *decks*. A deck may include one or more cards. Cards specify one or more units of user interaction (e.g., a choice menu, a screen of text or a text entry field). Logically, a user navigates through a series of WML cards, reviews the contents of each, enters requested information, makes choices and moves on to another card. The cards and decks model allows content authors to provide dialog-like interactive application.

- *Navigation and History Stack*
  WML has an HTML-style hyperlink mechanism for inter-card navigation. WML also defines an explicit history stack model that allows users to navigate between URIs. Several types of navigating operations in WML are called *tasks*; e.g., the user may jump to a particular URI, back to the previous card, or refresh display with the current variable values.

- *Variables*
  WML exposes a global flat context (i.e., a linear non-nested context).Variables hold user inputs or initialise their value explicitly. Variables may be used in the attribute values, and be substituted during run-time with their current value. The state of the variables can be used to modify the contents of a parameterised card without having to communicate with the server. Furthermore, the lifetime of a variable state can last longer than a single document and can be shared across multiple documents without having to use a server to save intermediate state between document invocations.

- *Events*
  WML has a simple event-handling feature. Some events are intrinsic to WML features (e.g., navigation, timer expiration).

See [WML1] for a full description.

WAE version 2.1 user agents should support WML1. See section 6.4 for details of related scripting language support

6.2.3.1. Managing Access to WML1 Content

Content based on WML1 is considered legacy WAP version 1 [WAP1] content since WAE version 2 deprecates WML1. Access to content based on WML1 can be achieved by

i. direct processing of WML1 content on the client, or

ii. transformation into WML2 on a WAP proxy, or

iii. transformation into WML2 on the client itself, or

iv. transformation into XHTMLMP.

WML2 is fully backward compatible with WML1 at a semantic level. A mapping rule for transforming WML1 into WML2 is defined using XSLT [WMLTR] but the use of the XSLT directly to transform the content is not required.

Whatever method of choices (i), (ii) or (iii) above is used to achieve WML1 compatibility, the result must be transparent to the user. Choice (iv) above is accepted as being not transparent and will be "best efforts" translations in situations where WAE version 2.1 user agents supporting only XHTMLMP access applications based on WML.

There are several cases to define for accessing WAP content through WAP2 proxies or other infrastructure implementing the necessary access to content, including content transformation services:

a) When a WAP2 client is accessing WAP2 content the optimisations specified in WAP2 are applied.

b) When a WAP2 client implementing support for WML2 only is accessing WAP1 content the transformation of content will be applied by the transformation service of the WAP proxy or other infrastructure.

c) When a WAP2 client also claims support for WML1, through the use of client implementation of the transformation from WML1 to WML2 or through the use of a hybrid user agent, and accesses WAP1 content the WAP proxy or other infrastructure will not apply any transformation functions to the content.

d) When a WAP2 client only supporting XHTMLMP, i.e. a WAE version 2.1 user agent supporting the minimum markup language features, accesses WAP1 content, transformation from WML1 to XHTMLMP on a best efforts basis may be provided by a WAP proxy or other infrastructure to support the WAP2 User agent

When transforming WML1 into WML2 and/or WCSS, optimisations and enhancements might be possible.

The specification of the hybrid user agent is considered out of scope and device implementers need to be aware of the need to manage all aspects of the user agent behaviour including but not limited to caching, events, history, variables etc. as though the device did implement a single User Agent.

6.2.4. Conformance Rules

This section is normative.

Conformance: A WAE User Agent MUST support the textual form of XHTML Mobile Profile [XHTMLMP]. (WAESpec-ML-C-005)

Conformance: A WAE User Agent SHOULD support the textual and/or binary forms of WML1[WML1]. (WAESpec-ML-C-006, WAESpec-ML-C-007)

Conformance: A WAE User Agent MAY support the textual form of WML2[WML2]. (WAESpec-ML-C-003)

Conformance: When a User Agent supporting XHTML Mobile Profile [XHTMLMP] and WML1[WML1] and/or WML2[WML2] navigates between XHTML Mobile Profile[XHTMLMP] and WML1 [WML1] or WML2[WML2] or between WML1[WML1] and WML2[WML2], the WML context (history and variables) MUST be maintained across languages, as if a single user agent was supported. (WAESpec-ML-C-004)

6.3. Wireless CSS

6.3.1. Description

This section is informative.

Wireless CSS [WCSS] is an extended subset of the *Cascading Style Sheet* (CSS)[CSS2], which is used to specify document presentation in conjunction with WML2 and XHTML Mobile Profile. This subset is based on CSS Mobile Profile[CSSMP], but further optimised for small devices that have small displays and limited computational resources.

The `text/css` MIME media type[RFC2318] can be used to represent Wireless CSS style sheets, thus no specific MIME media type for Wireless CSS has been defined.

The use of style, and in particular [WCSS], to achieve presentational control is strongly encouraged for all devices capable of supporting it. The intent is that devices will support style unless it is impractical for that class of device, i.e. the device has insufficient user interface capabilities which prevents its meaningful use such as a fixed format, non UI programmable low-end device.

Some features in WAP version 1 can be achieved by both the WML proprietary elements/attributes and the XHTML elements/attributes with CSS properties. There should be preference given to the XHTML expression and thus content authors are encouraged to use the latter if possible, to leverage the convergence of WAP and the standard Internet.

See [WCSS] for more information.

6.3.2. Conformance Rules

This section is normative.

Conformance: A User Agent SHOULD support the use of style to allow authors to control the appearance of a document, i.e. the User Agent is expected to support it unless it is impractical for a class of device, i.e. insufficient user interface capabilities which preventing its meaningful use. Where the User Agent supports style to control appearance of the document, the User Agent MUST support Wireless CSS as defined in [WCSS]. (WAESpec-STY-C-001, WAESpec-STY-C-002)

6.4. Scripting languages in WAE2.1

This section is informative

WAE version 2.1 contains two scripting environments but only one is required.

WAE version 2.1 introduced the ECMAScript Mobile Profile scripting environment which is the preferred scripting environment, this preference being indicated by its "required" status for all conformant WAE version 2.1 user agents. The WMLScript and its associated WMLScript Standard Libraries scripting environment remains part of WAE version 2.1 though its status is deprecated to be consistent line with WML. The inclusion of WMLScript and WMLScript Standard Libraries provides a recognised backwards compatibility path.

ECMAScript Mobile Profile should be use for all content and services aimed at WAE version 2.1 user agents.

6.4.1. WMLScript

6.4.1.1. Description

This section is informative.

WMLScript is a lightweight procedural scripting language. It enhances the standard browsing and presentation facilities of WML with behavioural capabilities, support of more advanced UI behaviour, client intelligence, a convenient mechanism to access the device and its peripherals, and reduces the need for round-trips to the origin server.

WMLScript is loosely based on a subset of the ECMAScript WWW scripting language. It is an extended subset of ECMAScript and forms a standard means of adding procedural logic to WML documents. WMLScript refines ECMAScript for the narrow-band device, integrates it with WML, and provides hooks for integrating future services and in-device applications. The high reliance of WMLScript on WML implies that WMLScript's capabilities can only be fully utilised used in conjunction with WML.

WMLScript provides the application programmers with a variety of interesting capabilities:

- The ability to perform mathematical functions and complex procedural logic.
- The ability to check the validity of the user's input before it is sent to the content server
- The ability to access device facilities and peripherals
- The ability to interact with the user without introducing round-trips to the origin server (e.g., display an error message)

Key WMLScript features include:

- *ECMAScript-based scripting language:*
  WMLScript starts with an industry standard solution and adapts it to the narrow-band environment. This makes it very easy for a developer to learn and use WMLScript.
- *Procedural Logic:*
  WMLScript adds the power of procedural logic to WAE.
- *Event-based:*
  WMLScript may be invoked in response to certain user or environmental events.
- *Compiled implementation:*
  WMLScript can be compiled down to a more compressed bytecode that is transported to the client.

- *Integrated into WAE:*
  WMLScript is fully integrated with the WML browser. This allows authors to construct their services using both technologies, using the most appropriate solution for the task at hand. WMLScript has access to the WML state model and can set and obtain WML variables. This enables a variety of functionality (e.g., validation of user input collected by a WML card).

- *International Support*
  The character set for WMLScript source text is the Universal Character Set, defined jointly by the Unicode Standard and ISO/IEC 10646. The character encodings UTF-8 and UTF-16 must be supported at a minimum. Additional support may be available for WMLScript source text using compatible character sets and encodings from which the characters may be transcoded into the Universal Character Set.

- *Efficient extensible library support:*
  WMLScript can be used to expose and extend device functionality without changes to the device software.

One design objective of the WMLScript language was to make it as close as possible to core JavaScript™[JavaScript]. In particular, WMLScript was based on the ECMA-262 Standard "ECMAScript Language Specification". The originating technologies for the ECMA Standard include many technologies, most notably JavaScript™ and JScript™. WMLScript is not fully compliant with ECMAScript. The standard has been used only as the basis for defining the WMLScript language. WMLScript supports several categories of functions including:

- *Local script functions* (i.e., script functions defined inside the same script that the calling expression is in)

- *External script functions* (i.e., script functions defined in another script not containing the calling expression)

- *Standard library functions* (i.e., functions defined in a library that is part of the WAE specification)

WMLScript defines several standard libraries including a language library, a string library, a browser library, a floating point library and a dialog library. See [WMLStdLib] for more information.

A WAE User Agent executes WMLScript in textual and/or bytecode format. Where a User Agent requires WMLScript bytecode, WAP proxies should provide the compilation function.

WAE version 2.1 user agents may support WMLScript and WMLScript Standard Libraries to provide the scripting capability for WML. WAE version 2.0 and all WAE version 1 user agents must support WMLScript and WMLScript Standard Libraries.

WMLScript and the associated WMLScript Standard Libraries are deprecated in WAE version 2.1.

Conformance criteria are stated in section 6.4.4.

6.4.2. ECMAScript Mobile Profile

6.4.2.1. Description

This section is informative.

ECMAScript Mobile Profile [ESMP] provides the general scripting capability for the WAE version 2.1 user agent and is consistent with the embracing of standard Internet technologies in the WAP version 2 architecture [WAPArch].

ESMP is based on a profile of ECMAScript [ECMAScript] targeted at mobile devices. ESMP inherits much from the ECMA327 [ECMA327] profile of ECMAScript Edition 3 [ECMAScript] but there are small differences in functionality between ECMA327 and the core functionality of ESMP. ESMP also adopts well-known and appropriate host objects to provide a common, interoperable programming environment to application writers.

ESMP complements the XHTML markup language [XHTMLMP] and style capabilities of WCSS in WAP. However, the XHTMLMP and WCSS content is *static* and there is no way to extend the language without modifying the markup itself. The following list contains some capabilities that are not supported by XHTML:

- The ability to check the validity of user input (forms validation).
- The ability to apply mathematic and procedural logic locally to document data.

- Providing access to facilities of the device. For example, on a phone, allow the programmer to make phone calls, send messages, add phone numbers to the address book, access the SIM card etc.
- The ability to generate messages and dialogs locally, reducing the need for expensive round-trip for alerts, error messages, confirmations etc.
- The ability to handle events.
- The ability to allow the dynamic creation and/or modification of documents on the client.

ESMP is designed to overcome these limitations and to provide programmable functionality that can be used over narrowband communication links and in clients with limited capabilities.

ESMP is intended to provide interoperability between WAP capable devices implementing ESMP and applications or content which exploit ESMP and with existing Internet content/applications which use the feature combination from ECMAScript and additional host objects that represents ESMP.

Mechanisms for extensibility and version control are provided.

WAE version 2.1 user agents must support ESMP to provide the scripting capability for XHTML Mobile Profile version 1.1.

Conformance criteria are stated in section 6.4.4.

6.4.3. Managing access to WMLScript content in WAE2.1

This section is informative.

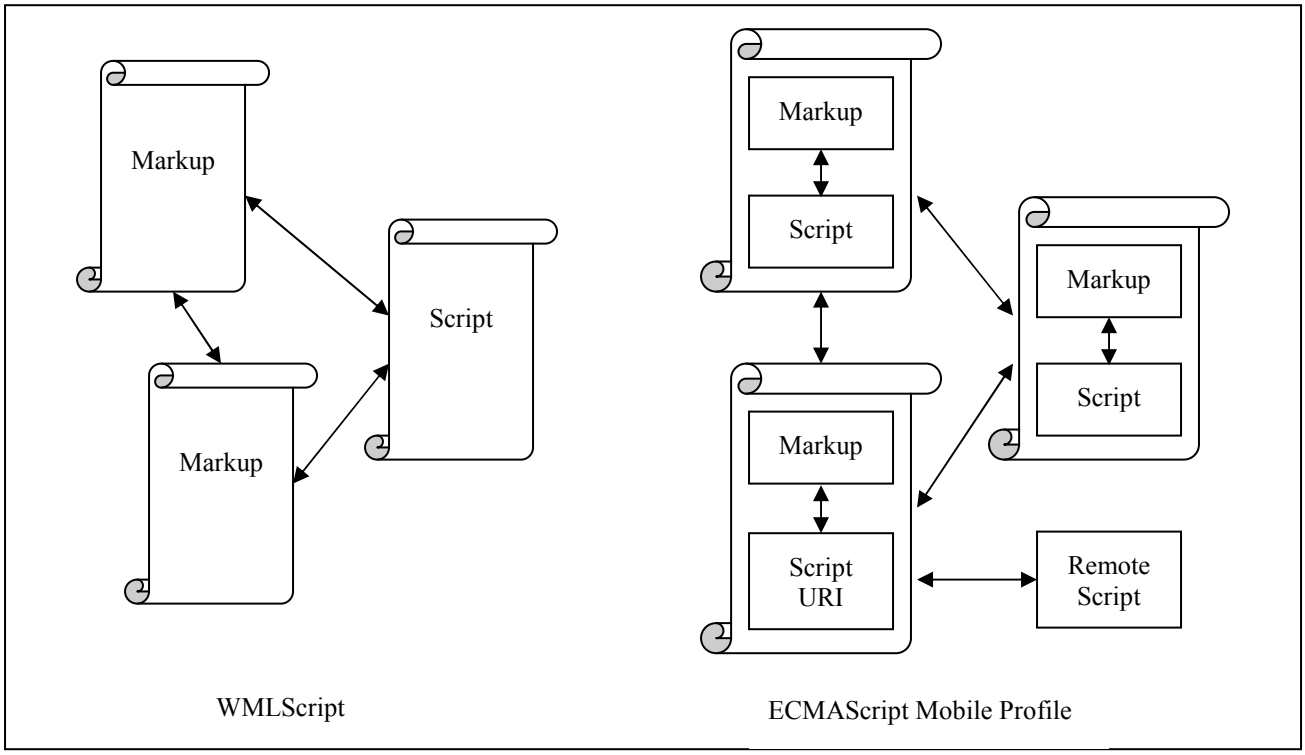
ECMAScript Mobile Profile [ESMP] and WMLScript are not directly compatible due to syntactical and in part semantic differences, the use of WMLScript binary data where User Agents do not support the textual version, and the relationship between script and markup contents in an application.

WMLScript is primarily intended to be used with WML while ECMAScript Mobile Profile is only to be used with XHTMLMP content. WAE version 1 [WAESpec1] defined the relationship between WML and WMLScript. WAE version 2.0 [WAE20] maintained the relationship between WML and WMLScript but did not clearly define the relationship between XHTMLMP and WMLScript; use of WMLScript with XHTMLMP10 is possible but with constraints, i.e. the scripting may need to subsequently call WML content when passing back any parameters. WAE version 2.1 makes the definition of relationship between the markup and the script languages to ensure a smooth transition from the combination of WML and WMLScript based applications to the combination of XHTMLMP and ESMP based applications.

The WML documents and associated WMLScript documents in a WAE are essentially peer to peer and share global variables. An application can therefore be constructed purely from WMLScript. The shared variables are used as a means to pass information between markup and script.

ECMAScript Mobile Profile on the other hand only shares a context with its host document, i.e. the document that includes either the script or the reference to the associated script. The script does not exist on its own. Sharing of information between the markup and script uses either the form variables or other well established means such as cookies.

The following picture illustrates the invocation and flows for various documents, i.e. documents, primarily WML, can call WMLScript documents, cards or decks, or XHTMLMP documents can contain ESMP which in turn can call other documents containing markup and script etc..



For these reasons the transition from WMLScript to ESMP is not a simple evolution with inherent backwards compatibility. Further the ECMAScript and WMLScript languages have different semantics, syntax and constructs for extensibility (objects vs libraries) adding to the problems of evolving with inherent backwards compatibility.

The ECMAScript Mobile Profile specification [ESMP] compares the semantics of WMLScript and ESMP. While ESMP does not contain a complete overlap with WMLScript it is asserted the omissions are not relevant to ESMP and this does not impact backwards compatibility. Migration of the script component of applications can therefore be considered possible through the means of re-authoring, translation or through native support for both WMLScript and ESMP on user agent.

Any such translation process or re-authoring needs to consider the relationship between script and markup and map variables to cookies, provide appropriate minimalist host documents, etc. as appropriate.

This affords significant savings opportunities for user agent developers and application providers as translation processes can be provided to authors as tools or to service providers in terms of a value added run-time process for the WAP Proxy.

The requirement that WAE version 2.1 User Agents must support XHTMLMP and ESMP (see sections 6.2.1 & 6.4.2) and may support WML and WMLScript (see sections 6.2.2, 6.2.3 & 6.4.1) provides the convergent application environment for WAE while enabling backwards compatibility. However support for WML and WMLScript is optional and both are deprecated. Legacy applications based on WML and WMLScript can be migrated through the means of tooling, re-authoring and translation to use the combination of XHTMLMP and ESMP..

6.4.4. Conformance Rules

This section is normative.

Conformance: A User Agent MUST support ECMAScript Mobile Profile [ESMP] for any content using an XHTMLMP MIME type. (WAESpec-ESMP-C-001)

Conformance: A WAP proxy MUST support the fetching of, and subsequent delivery in response to the fetch of, ECMAScript Mobile Profile [ESMP] content in response to a request from a user agent (WAESpec-ESMP-S-001)

Conformance: A User Agent MAY support WMLScript[WMLScript] and WMLScript Standard Libraries[WMLStdLib] for any content using a WML1 or WML2 MIME type, by supporting the textual and/or bytecode format(s). (WAESpec-WMLS-C-001, WAESpec-WMLS-C-002)

Conformance: A WAP proxy MUST transform WMLScript into WMLScript bytecode as defined in [WMLScript] when the User Agent prefers the binary format. (WAESpec-WMLS-S-001)

Conformance: Upon detection of a compile error, a WAP proxy SHOULD inform the WAE User Agent of the error by using the HTTP status code 502 “Bad Gateway”[HTTP/1.1]. (WAESpec-WMLS-S-002)

6.5. WBXML

6.5.1. Description

This section is informative.

WAP Binary XML (WBXML) is a compact binary representation of XML. The binary XML content format is designed for several purposes:

- *Reduction of the content size*
  Use of WBXML reduces the transmission size of XML documents, allowing more effective use of XML data on narrow-band communication channels.

- *Separation of XML parser from the application*
  The WBXML decoder may be more compact and suitable for devices that have limited memory and computational resources than full XML parsers. WBXML allows WAP proxies to encode textual XML documents into WBXML form on the behalf of such devices.

The MIME media type for generic WBXML is application/vnd.wap.wbxml. However, each content format that may be encoded into the binary format may have a dedicated MIME media type for that binary format. See [WBXML] for more information.

There are no conformance rules solely for WBXML. Support of the WBXML based binary format contents is defined in each section or document that describes its textual form.

6.6. Graphical Images

6.6.1. Description

This section is informative.

WAE provides a visual environment that is designed to address several competing requirements, including support for multiple pixels depths, support for colour space tables, small encoding, very low CPU and RAM decoding and presentation demands and allowance for commonly available tools and support.

The Wireless BitMaP (WBMP) format enables graphical information to be sent to a variety of handsets. The WBMP format is terminal independent and describes only graphical information. The MIME Media type of WBMP is image/vnd.wap.wbmp. The WBMP format is specified in WAE Media Types Specification [WAEMedia].

A WAP proxy may transform a graphical image into another format when the User Agent does not support the format of the original image, and the User Agent informs a WAP proxy, with which it is communicating, of the media types of the supported image formats (e.g., using HTTP/WSP Accept header or UAProf).

A WAE User Agent may support any graphical image formats beyond the mandatory image format.

6.6.2. Conformance Rules

This section is normative.

Conformance: A WAE User Agent MAY support graphical images. (WAESpec-IMG-C-001)

Conformance: A WAE User Agent that supports graphical images MUST support Wireless BitMaP (WBMP) as defined in [WAEMedia]. (WAESpec-IMG-C-002)

6.7. The Electronic Business Card Format (vCard)

6.7.1. Description

This section is informative.

WAE User Agents may exchange electronic business cards during both client-server communication and peer-to-peer communication. WAE defines the use of the industry-standard vCard as a format for exchanging electronic business cards.

Exchange of vCard information may be achieved using either of the following methods :

- WDP datagram exchange, thereby enabling clients to communicate using vCard without the use of a WAP proxy.

  To accurately identify data exchanged in a WDP datagram packet, a well-known port number is used, thereby allowing the user agent to presume that data received on the port is in a particular format. All data must be sent to the right port number and data received on that port should be assumed to be of the associated type. See [WDP] for details of the port numbers.

- WSP- or W-HTTP-based requests to a network server or receiving data via a WAP PUSH. This enables clients and servers to communicate with each other using vCards.

  The WSP or W-HTTP Content-Type header identifies the data being exchanged in the request. The header contains a MIME media type indicating the data type in the associated WSP or W-HTTP payload.

  For vCard phone book and business card data exchange the MIME media type is text/x-vCard.

  In the situation where the Content-Type header is missing or unavailable, the user agent may use other methods of determining the type of the data. The file extension in the data name may provide an indication of the content type. For vCard phone book and business card data exchange the file extension name is .vcf.

Any terminal supporting phone book exchange must at least be able to:

- display the ‘Name’ and ‘Telephone Number’ properties to the user upon receipt of the vCard,
- include the 'Name' and 'Telephone Number' properties in any transmitted vCards.

6.7.2. Conformance Rules

This section is normative.

Conformance: WAE User Agents MAY exchange electronic business cards. (WAESpec-VCARD-C-001)

Conformance: WAE User Agents supporting vCard[VCARD] exchange through WSP and/or W-HTTP MUST support the text/x-vCard MIME type. WAE User Agents supporting vCard[VCARD] exchange MAY use the .vcf file extension name in addition to the MIME type, or if the correct MIME type is not provided, when determining the data type.(WAESpec-VCARD-C-002)

Conformance: WAE User Agents supporting vCard WDP exchange MUST support vCard on the port assigned in [WDP] (WAESpec-VCARD-C-003)

Conformance: User Agents that support electronic business card exchange MUST be able to display the ‘Name’ and ‘Telephone Number’ properties (WAESpec-VCARD-C-004)

Conformance: User Agents that support transmission of electronic business cards MUST include the ‘Name’ and ‘Telephone Number’ properties (WAESpec-VCARD-C-005)

6.8. Internet Calendaring and Scheduling Core Object (vCalendar)

6.8.1. Description

This section is informative.

WAE User Agents may exchange calendar and scheduling information in both client-server communication and peer-to-peer communication. WAE defines the use of the industry-standard vCalendar for the use of exchanging calendar and scheduling information.

Exchange of vCalendar information may be achieved using either of the following methods:

- WDP datagram exchange, thereby enabling clients to communicate using vCalendar without the use of a WAP proxy or other network proxy.

  To accurately identify data exchanged in a WDP datagram packet, a well-known port number is used, thereby allowing the user agent to presume that data received on the port is in a particular format. All data must be sent to the right port number and data received on that port should be assumed to be of the associated type. See [WDP] for details of the port numbers.

- WSP- or W-HTTP-based requests to a network server or receiving data via a WAP PUSH. This enables clients and servers to communicate with each other using vCalendar objects.

  The WSP or W-HTTP Content-Type header identified the data being exchanged in the request. The header contains a MIME media type indicating the data type in the associated WSP or W-HTTP payload.

  For vCalendar calendar and scheduling information exchange the MIME media type is text/x-vCalendar.

  In the situation where the Content-Type header is missing or unavailable, the user agent may use other methods of determining the type of the data. The file extension in the data name may provide and indication of the content type. For vCalendar calendar and scheduling information data exchange the file extension name is .vcs.

Any terminal supporting calendar and scheduling information must at least be able to display the vEvent object to the user.

6.8.2. Conformance Rules

This section is normative.

Conformance: WAE User Agents MAY exchange calendar and scheduling information. (WAESpec-VCAL-C-001)

Conformance: WAE User Agents supporting vCalendar[VCAL] exchange through WSP and/or HTTP MUST support the text/x-vCalendar MIME type. WAE User Agents supporting vCalendar[VCAL] exchange MAY use the .vcs file extension name in addition to the MIME type, or if the correct MIME type is not provided, when determining the data type.(WAESpec-VCAL-C-002)

Conformance: WAE User Agents supporting vCalendar WDP exchange MUST support vCalendar on the port assigned in [WDP] (WAESpec-VCAL-C-003)

Conformance: User Agents that support calendar and scheduling information MUST be able, at least, to display the vEvent object to the user (WAESpec-VCAL-C-004)

6.9. Multipart Messages

6.9.1. Expected usage and process of multipart messages

This section is informative

WAE includes a multipart encoding specification, suitable for exchanging multiple typed entities over Hypermedia Transfer Service as well as Push. WSP[WSP] defines a compact binary form of the MIME multipart entity (see [RFC2045], [RFC2046], [RFC2047]), which is optimised for narrow-band environments.

The multipart/mixed messages can be used to transfer multiple related and/or unrelated resources at once. The multipart/related can be used to transfer related resources together, e.g., a document with its related images using hyperlinks. The multipart/alternative can be used to transfer the alternative resources of the same information, e.g., the same information in different languages.

6.9.2. multipart/mixed

This section is normative.

Upon receipt of the multipart/mixed (or equivalent application/vnd.wap.multipart.mixed) content, the User Agent SHOULD decompose the message into its elements. The elements SHOULD be processed in the order in which they are in the multipart message. The processing for each element is the same as it would be when the element is received individually.

Conformance: A User Agent that supports WSP SHOULD support application/vnd.wap.multipart.mixed. (WAESpec-MUL-C-001)

Conformance: A User Agent that supports W-HTTP SHOULD support multipart/mixed. (WAESpec-MUL-C-002)

Conformance: A WAP proxy that supports WSP SHOULD be able to convert multipart/mixed into application/vnd.wap.multipart.mixed. (WAESpec-MUL-S-002)

6.9.3. multipart/related

This section is normative.

Upon receipt of the multipart/related (or equivalent application/vnd.wap.multipart.related) content, a User Agent SHOULD process the content according to [RFC2387]. The multipart is retained as a single compound entity. The root entity is processed as if only that entity was received. The processing of the root entity might imply that other entities in the multipart are processed.

Conformance: A User Agent that supports WSP SHOULD support application/vnd.wap.multipart.related. (WAESpec-MUL-C-003)

Conformance: A User Agent that supports W-HTTP SHOULD support multipart/related. (WAESpec-MUL-C-004)

Conformance: A WAP proxy that supports WSP SHOULD be able to convert multipart/related into application/vnd.wap.multipart.related. (WAESpec-MUL-S-003)

6.9.4. multipart/alternative

This section is normative.

Upon receipt of the multipart/alternative (or equivalent application/vnd.wap.multipart.alternative) content, a User Agent should choose the "best" content type based on the order of the entities within the multipart message and the local environment, e.g. device constraints and user preferences. The entities should be ordered from least preferable to most preferable representation.

Conformance: A User Agent that supports WSP MAY support application/vnd.wap.multipart.alternative. (WAESpec-MUL-C-005)

Conformance: A User Agent that supports W-HTTP MAY support multipart/alternative. (WAESpec-MUL-C-006)

Conformance: A WAP proxy that supports WSP SHOULD be able to convert multipart/alternative into application/vnd.wap.multipart.alternative. (WAESpec-MUL-S-004)

6.9.5. multipart/form-data

This section is normative.

The multipart/form-data[RFC2388] returns a set of values from a form filled out by the user. Upon receipt of the application/vnd.wap.multipart.form-data, a WAP proxy MUST decode it into multipart/form-data before tranferring it to the origin server.

Conformance: A User Agent MUST support multipart messages returning a set of values from a form (WAESpec-MUL-C-007)

Conformance: A User Agent that supports WSP MUST be able to submit form data in application/vnd.wap.multipart.form-data. (WAESpec-MUL-C-008)

Conformance: A User Agent that supports W-HTTP MUST be able to submit form data in multipart/form-data. (WAESpec-MUL-C-009)

Conformance: A WAP proxy that supports WSP MUST be able to convert application/vnd.wap.multipart.form-data into multipart/form-data. (WAESpec-MUL-S-001)

6.10. Channels

This section is informative.

WTA defines a separate content format and encoding for delivering channels from the WTA server/WAP proxy to the WTA User Agent. A channel is the format which is delivered to and stored in the client for executing WTA services. The MIME media type of the channel in textual format is text/vnd.wap.channel. The channel also may be encoded into its binary form using WBXML; the MIME media type of the binary form of channel is application/vnd.wap.channelc.

See [WTA] for details on the channel format.

6.11. Service Indication

This section is informative.

The Service Indication (SI) content format is a notification, sent to end-users in an asynchronous manner. Such notifications may, for example, indicate new e-mails, changes in stock price, news headlines, advertising, or reminders.

The MIME media type of the Service Indication is text/vnd.wap.si. The Service Indication may also be encoded into its binary form using WBXML; the MIME media type of the binary form of the Service Indication is application/vnd.wap.sic.

See [ServiceInd] for more information.

6.12. Service Loading

This section is informative.

The Service Loading (SL) content format directs a User Agent on a mobile client to load and execute a service. The SL contains a URI indicating the content to be loaded by the User Agent without user intervention when appropriate. The Service Loading is “pushed” by a Push Initiator to User Agents.

The MIME media type of the Service Loading is text/vnd.wap.sl. The Service Loading may also be encoded into its binary form using WBXML; the MIME media type of the binary form of the Service Loading is application/vnd.wap.slc.

See [ServiceLoad] for more information.

6.13. Cache Operation

This section is informative.

The Cache Operation content format is an XML document that is used to indicate the validity of cached objects with the given URIs or cached objects with the same URI prefix.

The MIME media type of Cache Operation content is text/vnd.wap.co. The Cache Operation media type may also be encoded using WBXML; the MIME media type of the binary form of the Cache Operation is application/vnd.wap.coc.

The Cache Operation is defined in [CacheOp].

6.14. Provisioning Document

This section is informative.

The provisioning document type is an XML document that is used to convey provisioning information. See [ProvCont] for the specification of the document type for the provisioning document.

The MIME media type of the textual document is text/vnd.wap.connectivity-xml. . A provisioning document may also be encoded into a binary format based on WBXML. The MIME-type of the binary form of the provisioning document is application/vnd.wap.connectivity-wbxml.

7. WAE Features

7.1. Hypermedia Transfer Service Interface

7.1.1. Transport Protocol

7.1.1.1. Description

This section is informative.

WAE User Agents do not depend on any particular transport protocol, although WAE only defines a browser model of User Agent. User Agents are required to provide a hypermedia transfer service[WAPArch]. The combination of WSP (Wireless Session Protocol) [WSP] and WTP (Wireless Transaction Protocol) [WTP] provides the hypermedia transfer service over secure and non-secure datagram transports. The HTTP (Hypertext Transfer Protocol) [HTTP/1.1] provides the hypermedia transfer service over secure and non-secure connection-oriented transports. WAE User Agents must, at a minimum, implement either WSP or Wireless Profiled HTTP. The network communication takes place in the form of WSP/HTTP 1.1 headers and content.

7.1.1.2. Conformance Rules

This section is normative.

Conformance: WAE User Agents and WAP proxies MUST support the Hypermedia Transfer Service. (WAESpec-HTS-C-001, WAESpec-HTS-S-001)

Conformance: A WAE User Agent and a WAP proxy MUST support WSP[WSP] or Wireless Profiled HTTP[W-HTTP]. (WAESpec-HTS-C-002, WAESpec-HTS-C-003, WAESpec-HTS-S-002, WAESpec-HTS-S-003)

7.1.2. Cache Model

7.1.2.1. Description

This section is informative.

WAE defines the caching model for User Agents. The User Agent caching model is an implementation of the HTTP/1.1 caching model used on the World Wide Web. The design is intended to allow the adoption of the HTTP/1.1 caching model with no loss of semantics or function while providing support for low-end WAP terminals.

WAP Caching Model Specification [CacheMod] addresses the following issues:

- User agent caching of resources fetched from network services. This may include XTHML Mobile Profile documents, WML documents, WMLScript compilation units, or any other resources named with a URL and fetched over the hypermedia transfer service.
- WAP proxy behaviour – the role and responsibility of WAP proxies in the implementation of reliable caching.
- Support infrastructure needed to implement HTTP/1.1 caching on a User Agent.

7.1.2.2. Conformance Rules

This section is normative.

Conformance: A User Agent MUST support caching model as specified in [CacheMod]. (WAESpec-HTS-C-004)

7.1.3. HTTP State Management (Cookie)

This section is informative.

WAE provides the HTTP state management model, also known as "cookie management", defined in [RFC2965]. Some User Agents may be capable of storing and managing cookies locally. This functionality follows precisely the current World Wide

Web model, where cookies are typically stored and managed by regular web browsers. In addition, WAE defines an additional alternative mechanism to enable WAP proxies to store and manage cookies on behalf of the User Agent.

See [HTTPSM] for more information.

7.1.4. Client Header Handling

This section is normative.

Client headers (as described in section 6.3.3.1 of [WSP]), which User Agents send as part of the WSP layer connect service primitive, MUST be used as cached request headers. This means that the request headers in each method invocation during the lifetime of the established WSP session MUST be combined with the cached request headers before the request header information is used. Upon WSP request, if a request header with a particular name is not present in the request but is present in the cached request headers, the cached request header MUST be regarded as included in the request. If a header is present in the request, any cached request headers with the same name MUST NOT be taken into account during the processing of the method invocation. This means that if a cached request header needs to be overridden for a request then the client MUST send the entire value of the original request header In the case of request headers that are lists (e.g., the Accept header) then the entire modified list MUST be sent.

Client headers that User Agents send as part of the resume service primitive of the WSP layer MUST be used to update the cached request headers. The new cached request headers MUST be created by combining the client headers with the existing cached request headers in the resume primitive. This would be the same combination mechanism as is used with request headers in method invocations.

Example 1, Combining empty request headers with cached request headers:

Request headers from User Agent:	**Cached request headers on WAP proxy:**	**WAP proxy generated headers:**
empty	Accept: image/vnd.wap.wbmp	Accept: image/vnd.wap.wbmp
(No additional request headers are needed)	Accept: text/vnd.wap.wml	Accept: text/vnd.wap.wml
	Accept-Language: dk	Accept-Language: dk
	Accept-Charset: iso-8859-1	Accept-Charset: iso-8859-1

Example 2, Changing the Accept-Language request header:

New Request headers from User Agent:	**Cached request headers on WAP proxy:**	**WAP proxy generated headers:**
Accept-Language: en	Accept: image/vnd.wap.wbmp	Accept: image/vnd.wap.wbmp
(User changed the default language)	Accept: text/vnd.wap.wml	Accept: text/vnd.wap.wml
	Accept-Language: dk	**Accept-Language: en**
	Accept-Charset: iso-8859-1	Accept-Charset: iso-8859-1

Example 3, Changing the Accept request header:

New Request headers from User Agent:	**Cached request headers on WAP proxy:**	**WAP proxy generated headers:**
Accept: text/vnd.wap.wml	**Accept: image/vnd.wap.wbmp**	Accept: text/vnd.wap.wml
(User disabled downloading images)	Accept: text/vnd.wap.wml	Accept-Language: dk
	Accept-Language: dk	Accept-Charset: iso-8859-1
	Accept-Charset: iso-8859-1	

Conformance: A WAP proxy that supports WSP Connect primitive MUST use client headers in the connect service primitive of WSP as cached request headers. (WAESpec-HTS-S-004)

Conformance: A WAP proxy that supports WSP Resume primitive MUST use client headers in the resume service primitive of WSP as cached request headers. (WAESpec-HTS-S-005)

7.1.5. OMA Download

This section is normative.

OMA Download [DLARCH, DLOTA] is based on two already existing and successful download services: Basic HTTP Download and MIDlet Download. The purpose of OMA Download is to provide a service similar to MIDlet Download. The difference between MIDlet Download and OMA Download is that OMA Download is not designed specifically for the downloading of Java MIDlets, or any other specific media type - OMA Download is a general download framework.

OMA Download extends Basic HTTP Download with two additional steps:

Before download, a description of the media object is downloaded. The description is a small file, the download descriptor, containing metadata about the media object and instructions to the download agent in the device, how to download the media object.

After download, a status report is posted to a Web site, indicating the outcome of the download. Depending on the outcome - success or failure - the user can be directed to a Web location provided in the download descriptor.

Conformance: A WAE User Agent MAY support OMA Download (WAESpec-DL-C-001).

7.2. URI Schemes

This section is normative.

WAE defines a number of URI schemes to identify resources. The general URI syntax is defined in [RFC2396].

Conformance: WAE User Agents and WAP proxies MUST be able to handle a URI that is at least 1024 octets. (WAESpec-URI-C-001, WAESpec-URI-S-001)

The following standard URI schemes are defined for WAE User Agents.

7.2.1. The HTTP URL Scheme

This section is normative.

The HTTP URL scheme identifies a particular URI syntax suitable for naming resources stored on hypermedia origin servers (see [RFC2396]). The specification of an http scheme does not imply the use of a particular communication protocol between a User Agent and an origin server or a WAP proxy. A User Agent provides access to the origin server specified by the URI either directly using W-HTTP/HTTP, or via a WAP proxy that supports protocol conversion between HTTP and WSP. Alternatively, the URI may specify a network server, which combines the function of WAP version 1 protocol gateway and origin server into one entity. In this case, the resource is accessed directly across the WSP protocol.

Conformance: A WAE User Agent retrieving a resource specified by the HTTP URI scheme MUST communicate with a WAP proxy or the origin server using Wireless Profiled HTTP and/or WSP. (WAESpec-URI-C-002, WAESpec-URI-S-002)

7.2.2. The HTTPS URI Scheme

This section is normative.

The HTTPS URI scheme indicates that the named resource is on a secure origin server. This conventionally implies the use of the HTTP[HTTP/1.1] over a transport layer security protocol such as TLS between the User Agent and the origin server. See [RFC2828] for Internet Security Glossary; [RFC2818] for HTTP over TLS; [RFC2817] for Upgrading to TLS in HTTP/1.1.

A User Agent retrieving the resource specified by the HTTPS URI using the connection-oriented protocol stack, consisting of Wireless Profiled HTTP [W-HTTP] and Wireless profiled TCP/IP [WP-TCP], MUST use Wireless Profiled HTTP over the transport layer security protocol in accordance with “WAP TLS Profile and Tunnelling Specification”[WAPTLS] to

communicate with the origin server; otherwise, an indication of lack of security MUST be given to the user and the retrieval attempt MUST be aborted.

If a User Agent uses the connection-oriented protocol through a WAP proxy to communicate with the origin server, the WAP proxy MUST behave in accordance with [RFC2817] and tunnel the transport layer security protocol (e.g., WAPTLS) through the proxy for the duration of the secure session. The User Agent MUST use the HTTP CONNECT method[WAPTLS] to establish the secure transport protocol tunnel with the origin server. An indication MUST be given to the user and the retrieval attempt MUST be aborted if a tunnel cannot be established.

The establishment of a secure session using the connection-oriented protocol whether directly or through one or more proxies SHOULD be indicated to the user.

A User Agent retrieving the resource specified by the HTTPS URI using the datagram-oriented protocol stack, consisting of WDP, WTP + WSP with WTLS providing the transport layer security protocol, MUST request the resource via WSP[WSP] using a WTLS[WTLS] secured session. An indication MUST be given to the user and the retrieval attempt MUST be aborted if the WTLS session cannot be established.

If the User Agent retrieving the resource specified by the HTTPS URI uses the datagram-oriented protocol through a WAP proxy to communicate to the origin server, the WAP proxy MUST use [HTTP/1.1] over TLS or SSL to communicate with the origin server; otherwise, the WAP proxy SHOULD deny the retrieval attempt and SHOULD respond with an indication to the User Agent by a well known warning or error code. In order to indicate to the origin server that the transport layer security consists of WTLS between User Agent and WAP proxy the HTTP VIA header [[HTTP/1.1] section 14.45] SHOULD be inserted by the WAP proxy with the received-protocol for the secure WSP connection being indicated as WSPS/[WSP version number] to indicate secure WSP.

In the event a User Agent accesses content directly from an origin server or application hosting domain, either using a secure tunnelled request through a WAP proxy or a direct W-HTTP connection, and the content is only available in a format that the User Agent does not support, the origin server or application hosting domain is strongly recommended either provide a transcoded version that can be supported by the User Agent, e.g. using an available and trusted transcoding program over the network, or abort the request with an appropriate error message to the User Agent.

Conformance: A User Agent and a WAP proxy MUST support HTTPS URI scheme. (WAESpec-URI-C-003, WAESpec-URI-S-003)

Conformance: A User Agent retrieving a resource specified by the HTTPS URI scheme using a connection-oriented protocol MUST conform with the behaviour specified in Section 7.2.2. (WAESpec-URI-C-004)

Conformance: A User Agent retrieving a resource specified by the HTTPS URI scheme using a connection-oriented protocol through a WAP proxy MUST conform with the behaviour specified in Section 7.2.2. A WAP proxy mediating a resource specified by the HTTPS URI scheme using a connection-oriented protocol MUST conform with the behaviour specified in section 7.2.2. (WAESpec-URI-C-004, WAESpec-URI-S-004)

Conformance: A User Agent retrieving a resource specified by the HTTPS URI scheme using a datagram-oriented protocol MUST conform with the behaviour specified in section 7.2.2. (WAESpec-URI-C-005)

Conformance: A User Agent retrieving a resource specified by the HTTPS URI scheme using a datagram-oriented protocol through a WAP proxy MUST conform with the behaviour specified in section 7.2.2. A WAP proxy mediating a resource specified by the HTTPS URI scheme using a datagram-oriented protocol MUST conform with the behaviour specified in section 7.2.2. (WAESpec-URI-C-005, WAESpec-URI-S-005)

Conformance: A User Agent retrieving a resource specified by the HTTPS URI scheme MUST report an error to the user, when a secure connection cannot be established. A WAP proxy mediating a resource specified by the HTTPS URI scheme MUST report an error by using the HTTP status code 502 “Bad Gateway”[HTTP/1.1], when a secure connection cannot be established. (WAESpec-URI-C-006, WAESpec-URI-S-006)

7.2.3. Other URI Schemes

This section is informative.

There are several URI schemes that are used in the WAE. For example:

- `wtai` – Access to some of the Wireless Telephony Application Interface (WTAI) function libraries can be handled through URI "calls" using the dedicated WTAI URI encoding scheme. See [WTAI] for more information.
- `pict` – A pictogram representation can be specified using PICT URL encoding scheme. See [PICT] for more information.
- `wps` – The base level persistent storage system's naming scheme. See [PSTOR] for more information
- `efi` – This scheme identifies the name as belonging to the EFI namespace. See [EFI] for more information.

7.3. Digital Rights Management

7.3.1. Description

This section is informative.

Digital Rights Management (DRM) [DRM, DRMCF]is the means to control the usage of media object once it has been downloaded. DRM enables content providers to define rights for media objects. It is possible to associate different rights with a single media object. Different rights may have different prices. A content provider can grant a user the rights to preview media objects for free and charge the user only for the full usage rights. DRM makes it possible to sell the rights to use the media object, rather than selling the media objects itself. The rights are expressed in an XML rights language [DRMREL]. When a media object is distributed in encrypted form, the key is with the rights. Thus the rights are required in order to use the media object.

7.3.2. Conformance Rules

This section is normative.

Conformance: A WAE User Agent MAY support OMA Digital Rights Management. (WAESpec-DRM-C-001)

7.4. Multimedia Messaging Service

7.4.1. Description

This section is informative.

The Multimedia Messaging Service (MMS) is an application by which a WAP client is able to provide a messaging operation with a variety of media types. The MMS defines the interaction between the WAP MMS Client and its service partner, the MMS Proxy -Relay, which operates as a WAP Origin Server for this specialised service.

This MMS specification suite consists of the MMS Architecture overview [MMSArch], MMS Client Transactions [MMSCTR] and MMS Encapsulation Protocol [MMSEncaps].

7.4.2. Conformance Rules

This section is normative.
Conformance: A WAE user agent MAY support MMS (WAESpec-MMS-C-001)

Conformance: A WAP Proxy or Server MAY support MMS (WAESpec-MMS-S-001)

7.5. Push

7.5.1. Description

This section is informative.

The WAP Push framework introduces a means within the WAP effort to transmit information to a client without a previous user action. A push operation in WAP occurs when Push Initiator transmits content to a User Agent.

The Push Initiator is on the Internet, and the WAP client is in the wireless domain. If the User Agent and the Push Initiator share no protocol, the Push Proxy Gateway (PPG)[PPGService] acts as an intermediate translating server. Thus, the Push Initiator contacts the PPG from the Internet side, delivering content for the client using Internet protocols. The PPG forwards the pushed content to the wireless domain, and the content is then transmitted over-the-air in the mobile network to the client.

In addition to providing simple proxy gateway services, the PPG has the capability of notifying the Push Initiator about the final outcome of the push operation. The PPG may also provide the Push Initiator with client capability lookup services, enabling a Push Initiator to select the optimal flavour of content for sending to the client.

The content encoded as defined in [PushMessage] is pushed using the Push Access Protocol[PAP] from the Push Initiator to the PPG and is pushed using the Push Over-The-Air(Push OTA) Protocol[PushOTA] from the PPG to the client. Push OTA Protocol over WSP is called OTA-WSP. Push OTA Protocol over W-HTTP is called OTA-HTTP. The Push Access Protocol, or PAP, uses XML messages that may be tunnelled through various well-known Internet protocols, for example HTTP. The OTA protocol is based on WSP and W-HTTP.

A User Agent in WAE handles pushed content based on its media type. Each media type specifies both data structure and its semantics and the User Agent is responsible for interpreting the content according to the rules specified for each media type. A User Agent should not take any other actions than discarding the content or placing it in cache, if neither the definition of the media type itself nor the User Agent specifies any particular push behaviour for that media type. For example, this applies to WML and WML Script pushed to the WML User Agent.

7.5.2. Conformance Rules

This section is normative.

Conformance: A User Agent MAY support Push. (WAESpec-PUSH-C-001)

Conformance: A User Agent that supports Push MUST support Push Message format as defined in [PushMessage]. (WAESpec-PUSH-C-009)

Conformance: A User Agent that supports Push MUST support Push OTA Protocol as defined in [PushOTA]. (WAESpec-PUSH-C-010)

Conformance: A User Agent that supports Push MUST support the Service Indication as defined in [ServiceInd]. (WAESpec-PUSH-C-011)

Conformance: A User Agent that supports OTA-WSP MUST support application/vnd.wap.multipart.mixed. (WAESpec-PUSH-C-002)

Conformance: A User Agent that supports OTA-HTTP MUST support multipart/mixed. (WAESpec-PUSH-C-003)

Conformance: A User Agent that supports OTA-WSP MUST support application/vnd.wap.multipart.related. (WAESpec-PUSH-C-004)

Conformance: A User Agent that supports OTA-HTTP MUST support multipart/related. (WAESpec-PUSH-C-005)

Conformance: A User Agent that supports OTA-WSP MUST support application/vnd.wap.multipart.alternative. (WAESpec-PUSH-C-006)

Conformance: A User Agent that supports OTA-HTTP MUST support multipart/alternative. (WAESpec-PUSH-C-007)

Conformance: A User Agent SHOULD NOT take any other actions than discarding the content or placing it in cache, if neither the definition of the media type itself nor the User Agent specifies any particular push behaviour for that media type. For example, this applies to WML and WML Script pushed to the WML User Agent (WAESpec-PUSH-C-008)

7.6. Internationalisation

7.6.1. Character Set and Character Encoding

This section is normative.

The document character set for XHTML Mobile Profile, WML and WMLScript is the Universal Character Set, defined jointly by the Unicode Standard [UNICODE] and ISO/IEC 10646 [ISO10646]. A conformant WAE User Agent or WAP proxy MUST support character encodings UTF-8 and UTF-16 as required by XML [XML]. Additional support MAY be available in User Agents using compatible character sets and encodings from which the characters can be transcoded into the Universal Character Set (e.g., US-ASCII, ISO-8859-1).

Although listed as an optional parameter, the use of the charset parameter in the Content-Type field is strongly recommended when the content (e.g., XHTML Mobile Profile document, WML document or WMLScript) is conveyed via the hypermedia transfer service, unless otherwise this information is reasonably embedded in the content (e.g., by a charset field in WBXML). A User Agent or a WAP proxy uses this information to determine the character encoding of the content. The charset parameter can also be used to provide protocol-specific operations, such as charset-based content negotiation in the hypermedia transfer service.

A User Agent should inform a WAP proxy or the origin server, with which it is communicating, which language and character encoding it supports using Accept-Language and Accept-Charset HTTP/WSP header.

Conformance: WAP proxies SHOULD transform character encoding of the content when the User Agent does not support the original character encoding. See section 7.7 for how the User Agent informs its characteristics. (WAESpec-I18N-S-001)

Conformance: WAP proxies and User Agents MUST support both UTF-8 and UTF-16; i.e., they MUST be able to process text encoded in UTF-8 and UTF-16. (WAESpec-I18N-C-001, WAESpec-I18N-C-002, WAESpec-I18N-S-002, WAESpec-I18N-S-003)

Conformance: When processing XML documents, it is REQUIRED to treat character encoding of the document as specified in [RFC3023]. WAP proxies or User Agents MUST inform the user of the error if the XML document includes unknown characters. (WAESpec-I18N-C-003, WAESpec-I18N-S-004)

7.6.2. Pictogram

This section is normative.

"WAP Pictogram Specification" [PICT] defines the common pictogram set and the format of its use within content. The common pictogram set is a set of pictograms that the User Agents recognise. Content authors may use these images to promote efficiency of communication, data transfer and network traffic. Manufactures may install images of pictograms that are proprietary to a device (e.g., size, colour, image format, etc.).

To meet the requirements of the worldwide market, the common pictogram set is divided into several classes. For example, some pictograms have operational, cultural and time independent semantics and are intended for general use. Other pictograms are included in a glossary of symbols representing certain embodiments.

Conformance: A WAE User Agent MAY support pictograms [PICT]. (WAESpec-PICT-C-001)

7.7. Advertising of User Agent Characteristics

This section is informative.

In order to optimise the WAE client-server model, a number of characteristics are sent from the user agent to the WAP origin server. These characteristics allow the origin server to avoid sending inappropriate content to the user agent. They also provide the server and WAP proxies with a means of customising the response for a particular user agent.

The general mechanisms designed to provide this functionality are described in detail in [UAPROF]. A WAE User Agent that wants to convey Accept header information must do so through standard hypermedia transfer service (i.e., HTTP or

WSP) headers such as Accept, Accept-Charset and Accept-Language. Information contained in these Accept headers must be consistent with the information provided by UAProf.

7.7.1. HTTP/WSP Accept headers

This section is normative.

Wireless Profiled HTTP and WSP includes the following request-header fields for describing user agent capabilities and user preferences to the server:

- *Accept* – The Accept request-header field can be used to specify certain media types which are acceptable for the response (e.g., text/vnd.wap.wml).
- *Accept-Charset* – The Accept-Charset request-header field can be used to indicate what character sets are acceptable for the response (e.g., iso-8859-1).
- *Accept-Encoding* – The Accept-Encoding request-header field restricts the content-codings that are acceptable in the response (e.g., compress, gzip).
- *Accept-Language* – The Accept-Language request-header field restricts the set of natural languages that are preferred as a response to the request (e.g., en, dk).

However, an origin server is not limited to these accept-headers and may vary the response based on any aspect of the request, including information outside the request -header fields.

See [WSP], [W-HTTP], and sections 14.1 and 10.4.7 of [HTTP/1.1] for guidance.

Conformance: A WAE User Agent that wants to advertise its characteristics MUST at least use HTTP/WSP Accept, Accept-Charset, Accept-Encoding and Accept-Language request headers regardless of the UAProf support. (WAESpec-UAC-C-001, WAESpec-UAC-C-002, WAESpec-UAC-C-003, WAESpec-UAC-C-004)

Conformance: WAP proxies MUST assume the value of Accept, Accept-Charset, Accept-Encoding and Accept-Language headers as defined in [HTTP/1.1]. If the WAP proxy cannot send a response which is acceptable to the WAE User Agent, the WAP proxy SHOULD respond as defined in [HTTP/1.1]. (WAESpec-UAC-S-001, WAESpec-UAC-S-002, WAESpec-UAC-S-003, WAESpec-UAC-S-004, WAESpec-UAC-S-005)

Conformance: When a User Agent expresses a preference through a mechanism such as an HTTP quality parameter, a WAP proxy SHOULD honour those preferences. For example, a WAP proxy SHOULD transform a binary form of WML1[WML1] document into the textual form when the User Agent prefers the textual format, while a WAP proxy SHOULD NOT transform the binary form of WML1[WML1] into the textual form when the User Agent prefers the binary format. A WAP proxy SHOULD transform WML1 document into WML2 document when the User Agent prefers WML2. (WAESpec-UAC-S-006)

Conformance: A WAP proxy SHOULD NOT send content that the User Agent does not accept. For example, a WAP proxy should transform WML1 into WML2 when the User Agent only accepts WML2. (WAESpec-UAC-S-007)

Conformance: A WAP proxy MUST transmit content without changing the media type when the WAE User Agent prefers the original media type. For example, while a WAP proxy SHOULD NOT transform the binary form of WML1[WML1] into the textual form when the User Agent prefers the binary format. (WAESpec-UAC-S-001, WAESpec-UAC-S-002, WAESpec-UAC-S-003, WAESpec-UAC-S-004, WAESpec-UAC-S-008)

7.7.2. UAProf

This section is informative.

UAProf captures classes of device information for content formatting by origin servers, WAP proxies and other interim servers. This device information is referred to as Capability and Preference Information (CPI). It includes information about:

- HardwarePlatform – screen size, colour capabilities, image capabilities, manufacturer, etc.

- SoftwarePlatform – operating system vendor and version, list of audio and video encoders, etc.
- BrowserUA – browser manufacturer and version, mark-up language and versions supported, scripting languages supported, etc.
- NetworkCharacteristics – bearer characteristics such as latency and reliability, etc.
- Supported WapCharacteristics – WMLScript libraries, WAP version, WML document size, etc.

The user agent profile is defined to comply with the emerging W3C standards for Composite Capabilities/Preferences Profile (CC/PP) [CC/PP, CCPPex] distribution over the Internet. The user agent profile uses an RDF schema and vocabulary [RDF], as defined in the CC/PP model, to define a robust, extensible framework for describing and transmitting CPI.

The CPI is transmitted and maintained using designated hypermedia transfer service (i.e., HTTP or WSP) headers.

The hypermedia transfer service uses Profile and Profile-Diff headers to convey the CPI. A Profile header contains URL(s), where each URL is referencing an externally accessible CPI document. The Profile-Diff header contains a CPI document. Multiple Profile and Profile-Diff headers may be cached by a WAP proxy and/or included with a request.

When transmitted using WSP, this information is initially conveyed when a WSP session is established with a compliant WAP proxy. The WAP proxy caches the CPI and applies it on all requests during the lifetime of the WSP session. The WAP proxy is also responsible for translating HTTP responses into appropriate WSP responses for delivery over the wireless network to the requesting client device. In forwarding these responses, the WAP proxy must also forward any CPI usage headers provided by the origin server and/or any intermediate HTTP proxies.

It is expected that a class of proxy services will be deployed to support the conversion of content based on the information available in the UAProf. Those adapting proxies will extend the request profile (e.g., by adding a new Profile-Diff) to advertise their capabilities. If WAP proxies or other interim servers wish to add to or modify the Accept* header information, they must also add an equivalent Profile-diff segment in order to reflect the same information in the CPI.

See [UAPROF] for more information.

7.8. Calendar and Phone Book

This section is informative.

WAE includes support for the exchange of calendar and phone book data objects. There are currently two available methods for exchanging vCard and vCalendar data:

- Using WDP datagrams enables clients to transfer vCard and vCalendar data without the use of a WAP proxy or other network proxy.
- Issuing WSP/HTTP-based requests to a network server or receiving data via WAP Push enables clients and servers transfer vCard and vCalendar data.

7.8.1. WDP Datagram Data Exchange

This section is informative.

To accurately identify data exchanged in a WDP datagram packet, a well-known port number is used. This allows the user agent to presume that any data received on the port is in a particular format. All data must be sent to the correct port number, and data received on that port should be assumed to be of the associated type.

When datagrams are received at the given WDP port, the information can be provided to a standard vCard or vCard reader, or it can be accessed by another application, such as the WML User Agent. The method of display is implementation dependent.

7.8.2. WSP/HTTP Data Exchange

This section is informative.

The WSP/HTTP Content-Type header identifies data exchanged in a WSP request. This header contains a MIME media type, indicating the data type in the associated WSP payload.

Data Type	MIME Media Type	File Extension Name
vCard (phone book and business card data)	text/x-vCard	.vcf
vCalendar (calendar data)	text/x-vCalendar	.vcs

Table 7-1: MIME Media Types and Extension Names

A User Agent should use the MIME Media Type to determine data type. In the situation where the Content-Type header is missing or unavailable, the user agent may use other methods of determining the type of the data. The file extension name may provide an indication of the type.

7.9. Provisioning

This section is informative.

Provisioning is the process by which a WAP client is configured with a minimum of user interaction. The term covers both OTA provisioning and provisioning by other means, e.g., SIM cards. The WAP provisioning framework specifies connectivity and application information to provide for devices.

An initial set of connectivity information must be loaded into the device in order to initialise a configuration context and establish a basic relationship between the device and a WAP infrastructure in this context. This information may contain a trusted point of configuration, including references to generic WAP proxies. The bootstrap process is bearer specific.

See [ProvArch], [ProvCont] and [ProvUAB] for more information.

7.10. External Functionality Interface (EFI)

This section is informative.

EFI allows for new kinds of functionality into a wireless terminal; either through the integration of new features into the terminal or by allowing external devices to be connected to the terminal.

EFI is a framework where applications are to access External Functionality in a uniform way through the EFI Application Interface (EFI AI). The EFI Application Interface (EFI AI) is a high level interface that shall suit a number of different applications. The EFI framework provides an extensible set of interfaces that can support new services, including the ability to query for the particular service as well as the ability to capture external device's functionality. External Functionality (EF) is a general term for components or entities with embedded applications that execute outside of WAE but conforms to the EFI requirement. The External Functionality can be built-in, or connected to a WAP terminal. This connection can be permanent or temporary.

EFI defines EF Classes, each of which is a group of the functions, devices or extensions that share the same functionality. Each EF Class has mandatory and optional functions. Examples of classes are payment, position, etc. An EF Class consists of one or more EF Unit(s), which provide services for that class.

EFI services are accessible through Script and Markup APIs.

- The Script API provides a well-defined, programmatic interface for accessing server attributes and class properties, and for discovering and instantiating services.
- The Markup API provides a simple means for invoking EFI services. EFI services that are involved using the Markup API return a document that is rendered by the WAE User Agent.

Both the Script and Markup APIs use an extensible namespace and naming scheme defined in the EFI framework.

See [EFI] for more information.

7.10.1. Conformance Rules

This section is normative

Conformance: A WAE User Agent MAY support EFI [EFI]. (WAESpec-EFI-C-001)

Conformance: A WAP proxy that supports WMLScript compilation function MUST be able to compile a script compilation unit that refers to the EFI library. See [EFI]. (WAESpec-EFI-S-001)

7.11. Synchronisation

This section is informative.

Data synchronisation is a process of exchanging information between multiple physical or virtual location for purpose of ensuring that each location's copy of that information reflects the same information content. Data Synchronisation can be achieved based on SyncML[SyncML].

7.12. Security and Access Control

7.12.1. Basic Authentication Scheme

This section is normative.

HTTP provides a simple challenge-response authentication mechanism. HTTP may be used by a server to challenge a client request and used by a client to provide authentication information. The "basic" authentication scheme is based on the model that the client must authenticate itself with a user-ID and a password. This basic authentication scheme is specified in [HTTP/1.1] and [HTTPBasicAuth].

This Basic authentication scheme is not a secure method of user authentication, nor does it in any way protect the cleartext entity transmitted across the physical network.

Conformance: User Agents MUST implement HTTP/1.1 Basic Authentication as specified in [HTTP/1.1] and [HTTPBasicAuth]. (WAESpec-SEC-C-001)

7.12.2. Access Control Pragma in WML and WMLScript

This section is informative.

Both WML and WMLScript include access control constructs that communicate to the client URL-based access restrictions. In particular, the constructs allow the authors of WML documents and WMLScript to grant public access to the content (i.e., the document or script can be referenced from other content) or restrict access to the content to set of particular documents or scripts.

The access control mechanism in WML or WMLScript is not a secure method, as the access control constructs in the content is always validated by the User Agent after the content was transmitted over the network. The purpose of the mechanism is to provide content authors the means to define the scope of the application context.

7.12.3. Persistent Storage

This section is normative

Persistent Storage [PSTOR] defines a means to store and retrieve data objects using storage, e.g. non-volatile memory, available to the WAE User Agent through a WMLScript interface.

WAE User Agents supporting PSTOR are RECOMMENDED to provide a means of managing access to stored objects to the applications or domain that stored them. (WAESpec-PSTOR-C-002)

Conformance: A WAE User Agent MAY support Persistent Storage. (WAESpec-PSTOR-C-001)

7.12.4. WMLScript Crypto Library

7.12.4.1. Description

This section is informative.

[CryptoLib] specifies the application layer security library interface for WMLScript [WMLScript] that provides cryptographic functionality to User Agents. [CryptoLib] also specifies a signed content format to be used to convey signed data to/from User Agents.

7.12.4.2. Conformance Rules

This section is normative

Conformance: A WAP proxy that supports WMLScript compilation function MUST be able to compile a script compilation unit that refers to the Crypto library. See [CryptoLib]. (WAESpec-SEC-S-001)

Conformance: A WAE user agent MAY support the WMLScript Crypto Library [Crypto]. (WAESpec-SEC-C-002)

7.12.5. ECMAScript Mobile Profile Crypto Library

7.12.5.1. Description

This section is informative.

[ESMPCrypto] specifies the application layer security object interface for ECMAScript Mobile Profile [ESMP] that provides cryptographic functionality to User Agents and complements the use of [CryptoLib] with [WMLScript]. [ESMPCrypto] also specifies a signed content format to be used to convey signed data to/from User Agents.

7.12.5.2. Conformance

This section is normative

Conformance: A WAE user agent MAY support ECMAScript Mobile Profile Crypto Library [ESMPCrypto] (WAESpec-SEC-C-003)

7.12.6. Secure Transport

This section is informative.

At the Transport Service layer protocols are defined for secure transport over datagrams and connections. WTLS is defined for secure transport over datagrams. WTLS and TLS are defined for secure transport over connections.

TLS is the preferred method, when the User Agent directly communicates with the origin server over TCP.

When the User Agent communicates with the origin server through a WAP proxy, the WAP proxy is required to translate protocol and convert content as described in [WAPArch]. This means that the WAP proxy used for the secure connection needs to be trusted by the origin server. "WAP Transport Layer End-to-end Security Specification" [E2ESec] defines a mechanism with which the origin server can instruct the user agent to navigate through the secure proxy.

7.13. User Agent Behaviour

7.13.1. Navigation History

7.13.1.1. Description

This section is normative.

The WAE User Agent MUST include a navigational history model that allows the author to manage backward navigation in a convenient and efficient manner. The user agent history is modelled as a stack of entries that represent the resources in the navigational path the user traversed to arrive at the current location. The stack is configured temporally, such that the newest entry is at the top of the stack and the oldest entry is at the bottom of the stack. The following operations MUST be performed on the history stack:

- Push - a new entry is pushed onto the history stack as an effect of forward navigation.
- Pop - the current entry (top of the stack) is popped as a result of backward navigation.

Additional operations may be defined for the processing of specific document types.

As each document or document fragment is accessed via an explicitly specified URL, (e.g., via the `href` attribute in `<a>`) an entry for the resource is added to the history stack even if it is identical to the most recent entry. At a minimum, each entry must record the resource request information that comprises the absolute URL of the resource, the method (get or post) used to access the resource. The exact composition of the stack entry is document type specific.

7.13.1.2. Conformance Rules

This section is normative

Conformance: The User Agent MUST implement the navigation history model and support all the operations on it defined in Section 7.13.1.1. (WAESpec-UAB-C-001).

7.13.2. The BACK Key

7.13.2.1. Description

This section is informative

The *BACK key* is a user interface object (physical key, soft key, or rendered user interface control) provided by the user agent to return to the previously viewed document. The BACK key is always available to the end user. The BACK key is commonly known as the "back button" on an HTML browser.

Given the WAP Forum's goal to provide a language with W3C convergence and the ability to navigate to a previously viewed document is a de-facto standard in desktop HTML browsers, the WAE 2 User Agent must provide the end user access to a BACK key at all times. This BACK key may be represented as a physical key, a soft key, or any rendered user interface control as long as it is available to the end user at all times.

7.13.2.2. The Back Key in WML1

This section is informative.

Unless otherwise overridden, the default behaviour of the BACK key is as follows:

- When the BACK key is activated by the end user, the User Agent must execute a `prev` task as defined in [WML1].

Through the use of the `do` element, the author may override the default behavior of the BACK key. When the author specifies a `do` element with the pre-defined role of `prev`, the default behavior of the WML1 User Agent is overridden with the task specified by the `do` element. If there is more than one `do` element with the type attribute equal to `prev`, then the first `do` element in document order overrides the BACK key.

7.13.2.3. Conformance Rules

This section is normative.

Conformance: The WAE User Agent MUST provide the end user access to a BACK key at all times (WAESpec-UAB-C-002).

Conformance: When the BACK key is activated the User Agent MUST perform a pop operation as defined in 7.13.1. (WAESpec-UAB-C-003).

Conformance: When the BACK key is activated the WML1 User Agent MUST execute a prev task as defined in [WML1]. (WAESpec-UAB-C-004).

Conformance: For a WML1 `do` element with the `type` attribute equal to `prev`, the WML1 User Agent MUST override the default behaviour of the BACK key, as defined in 7.13.2.2 with the task associated with the `do` element as defined in [WML1]. If there is more than one `do` element with the `type` attribute equal to `prev`, then the first `do` element in document order overrides the BACK key. (WAESpec-UAB-C-005).

7.14. WTA

This section is informative.

The Wireless Telephony Application [WTA] is an application framework for telephony services. WTA extends the standard WAE user agent by adding the capability to interface with mobile network services available to a mobile telephony device, e.g. setting up and receiving phone calls.

WTA provides markup and script interfaces to a specific set of local, telephony related, functions in the client. This interface is called the "Wireless Telephony Application Interface" [WTAI] and comprises two parts, the Public WTAI and the Network WTAI.

The Public WTAI comprises a set of simple non event generating services such as making a call, sending DTMF tones and writing a phone book entry. See [WTAI] section 8 for more details.

The Network WTAI comprises a set of common and network specific functions for call handling which necessitates the handling of events. See [WTAI] and the associated network specific addenda for details.

WTA user agents are logically separate from the WAE user agent (see [WTA]), thus all conformance criteria for WTA are contained in [WTA]. However the WAE user agent may support the Public WTAI.

7.14.1. Conformance Rules

This section is normative.

Conformance: A WAE user agent MAY support the WTAI Public URI functions as defined in [WTAI]. (WAESpec-WTA-C-001)

Conformance: A WAE user agent MAY support the WTAI Public WMLScript functions as defined in [WTAI]. (WAESpec-WTA-C-002)

Appendix A. Static Conformance Requirements (Normative)

Static Conformance Requirements define a minimum set of features that can be implemented to ensure that WAE User Agents and WAE Servers are able to inter-operate. While both WAE User Agent behaviour and WAE server behaviour are described in the WAE Specification, not all items apply to both entities, given separate tables for each entity. See [IOPProc] for syntax used in this section.

References to WML1 and WML2 in these static conformance requirements refer to [WML1] and [WML2] respectively whereas the SpecScrName, as defined in [IOPProc] and documented at http://www.openmobilealliance.org is WML for both.

A.1 WAE User Agent

A.1.1 WAE Media Types

Item	Function	Reference	Status	Requirement
WAESpec-MT-C-001	User agent use of MIME Media type	6.1	**M**	
WAESpec-MT-C-002	Support for MIME Media Types	6.1	**M**	

Item	Function	Reference	Status	Requirement
WAESpec-ML-C-003	Support for textual form of WML2	6.2	O	WML2:MCF AND WAESpec-ML-C-004
WAESpec-ML-C-004	Maintains WML context	6.2	O	
WAESpec-ML-C-005	Support for textual form of XHTMLMP	6.2	M	XHTMLMP:MCF
WAESpec-ML-C-006	Support for the textual form of WML1	6.2	O	WML1:MCF AND WAESpec-ML-C-004 AND WAESpec-UAB-C-004 AND WAESpec-UAB-C-005
WAESpec-ML-C-007	Support for the binary form of WML1	6.2	O	WML1:MCF AND WBXML:MCF AND WAESpec-ML-C-004 AND WAESpec-UAB-C-004 AND WAESpec-UAB-C-005

Item	Function	Reference	Status	Requirement
WAESpec-STY-C-001	Allows authors to control presentation of the document	6.3	O	WAESpec-STY-C-002
WAESpec-STY-C-002	Support for Wireless CSS	6.3	O	WCSS:MCF

Item	Function	Reference	Status	Requirement
WAESpec-WMLS-C-001	Support for WMLScript execution	6.4	O	WMLScript:MCF AND WAESpec-WMLS-C-002
WAESpec-WMLS-C-002	Support for WMLScript Standard Library	6.4	O	WMLScriptLibs:MCF

Item	Function	Reference	Status	Requirement
WAESpec-ESMP-C-001	Support for ECMAScript Mobile Profile	6.4.4	M	ESMP:MCF AND XHTMLMP-SCRIPT-C-001

Item	Function	Reference	Status	Requirement
WAESpec-IMG-C-001	Support for graphical images	6.6	O	WAESpec-IMG-C-002
WAESpec-IMG-C-002	Support for WBMP	6.6	O	WAEMT:MCF

Item	Function	Reference	Status	Requirement
WAESpec-VCARD-C-001	Exchanges electronic business cards	6.7	O	WAESpec-VCARD-C-002 AND WAESpec-VCARD-C-003 AND WAESpec-VCARD-C-004 AND WAESpec-VCARD-C-005
WAESpec-VCARD-C-002	Support of text/x-vCard MIME Type	6.7	O	
WAESpec-VCARD-C-003	vCard data port	6.7	O	WDP:MCF
WAESpec-VCARD-C-004	Ability to display 'Name' and 'Telephone Number' properties	6.7	O	
WAESpec-VCARD-C-005	Inclusion of 'Name' and 'Telephone Number' properties	6.7	O	

Item	Function	Reference	Status	Requirement
WAESpec-VCAL-C-001	Exchanges calendar and scheduling information	6.8	O	WAESpec-VCAL-C-002 AND WAESpec-VCAL-C-003 AND WAESpec-VCAL-C-004
WAESpec-VCAL-C-002	Support of text/x-vCalendar MIME Type	6.8	O	
WAESpec-VCAL-C-003	vCalendar data port	6.8	O	WDP:MCF
WAESpec-VCAL-C-004	Ability to display the vEvent object	6.8	O	

Item	Function	Reference	Status	Requirement
WAESpec-MUL-C-001	application/vnd.wap.multipart.mixed	6.9.2	O	
WAESpec-MUL-C-002	multipart/mixed	6.9.2	O	

WAESpec-MUL-C-003	application/vnd.wap.multipart.related	6.9.3	O	
WAESpec-MUL-C-004	multipart/related	6.9.3	O	
WAESpec-MUL-C-005	application/vnd.wap.multipart.alternative	6.9.4	O	
WAESpec-MUL-C-006	multipart/alternative	6.9.4	O	
WAESpec-MUL-C-007	Support for the multipart message returning a set of values from a form	6.9.5	M	WAESpec-MUL-C-008 OR WAESpec-MUL-C-009
WAESpec-MUL-C-008	application/vnd.wap.multipart.form-data	6.9.5	O	
WAESpec-MUL-C-009	multipart/form-data	6.9.5	O	

A.1.2 WAE Features

Item	Function	Reference	Status	Requirement
WAESpec-HTS-C-001	Support for Hypermedia Transfer Service	7.1.1	M	WAESpec-HTS-C-002 OR WAESpec-HTS-C-003
WAESpec-HTS-C-002	Support for WSP	7.1.1	O	WSP:MCF AND WAESpec-MUL-C-008
WAESpec-HTS-C-003	Support for W-HTTP	7.1.1	O	W-HTTP:MCF AND WAESpec-MUL-C-009
WAESpec-HTS-C-004	Support for Caching Model	7.1.2	M	CacheMod:MCF

Item	Function	Reference	Status	Requirement
WAESpec-URI-C-001	Minimum URI length	7.2	M	
WAESpec-URI-C-002	HTTP URL Scheme	7.2.1	M	
WAESpec-URI-C-003	HTTPS URL Scheme	7.2.2	M	WAESpec-URI-C-004 OR WAESpec-URI-C-005 OR WAESpec-URI-C-006
WAESpec-URI-C-004	HTTPS URI Scheme over W-HTTP	7.2.2	O	TLS:MCF
WAESpec-URI-C-005	HTTPS URI Scheme over WSP	7.2.2	O	WTLS:MCF
WAESpec-URI-C-006	Report an error when no TLS or WTLS support available	7.2.2	M	

Item	Function	Reference	Status	Requirement
WAESpec-DL-C-001	Support for OMA Download	7.1.5	O	DLOTA:MCF

WAESpec-DRM-C-001	Support for OMA DRM	7.3.2	O	DLRM:MCF

Item	Function	Reference	Status	Requirement
WAESpec-PUSH-C-001	Support for Push	7.5.2	O	WAESpec-PUSH-C-009 AND WAESpec-PUSH-C-010 AND WAESpec-PUSH-C-011 AND (WAESpec-PUSH-C-002 OR WAESpec-PUSH-C-003) AND (WAESpec-PUSH-C-004 OR WAESpec-PUSH-C-005) AND (WAESpec-PUSH-C-006 OR WAESpec-PUSH-C-007)
WAESpec-PUSH-C-002	application/vnd.wap.multipart.mixed	7.5.2	O	
WAESpec-PUSH-C-003	multipart/mixed	7.5.2	O	
WAESpec-PUSH-C-004	application/vnd.wap.multipart.related	7.5.2	O	
WAESpec-PUSH-C-005	multipart/related	7.5.2	O	
WAESpec-PUSH-C-006	application/vnd.wap.multipart.alternative	7.5.2	O	
WAESpec-PUSH-C-007	multipart/alternative	7.5.2	O	
WAESpec-PUSH-C-008	undefined push behaviour	7.5.2	O	
WAESpec-PUSH-C-009	Support for Push Message	7.5.2	O	PushMessage:MCF
WAESpec-PUSH-C-010	Support for Push OTA	7.5.2	O	PushOTA:MCF
WAESpec-PUSH-C-011	Support for Service Indication	7.5.2	O	ServiceInd:MCF

Item	Function	Reference	Status	Requirement
WAESpec-MMS-C-001	Support for MMS	7.4	O	MMSCTR:MCF

Item	Function	Reference	Status	Requirement
WAESpec-I18N-C-001	Support for UTF-8	7.6.1	M	
WAESpec-I18N-C-002	Support for UTF-16	7.6.1	M	
WAESpec-I18N-C-003	Treat the character encoding of an XML document as defined in [RFC3023].	7.6.1	M	

Item	Function	Reference	Status	Requirement
WAESpec-PICT-C-001	Support for Pictograms	7.6.2	O	WAPInterPic:MCF

Item	Function	Reference	Status	Requirement
WAESpec-UAC-C-001	Informs supported media type using Accept header	7.7.1	O	
WAESpec-UAC-C-002	Informs supported character encoding using Accept-Charset header	7.7.1	O	
WAESpec-UAC-C-003	Informs supported content-codings using Accept-Encoding header	7.7.1	O	
WAESpec-UAC-C-004	Informs supported language using Accept-Language header	7.7.1	O	

Item	Function	Reference	Status	Requirement
WAESpec-EFI-C-001	Support for EFI	7.10.1	O	EFI:MCF

Item	Function	Reference	Status	Requirement
WAESpec-PSTOR-C-001	Support for Persistent Storage	7.12.3	O	
WAESpec-PSTOR-C-002	Managing access to stored objects	7.12.3	O	

Item	Function	Reference	Status	Requirement
WAESpec-SEC-C-001	Support for HTTP/1.1 Basic Authentication	7.12.1	M	
WAESpec-SEC-C-002	Support for WMLScript Crypto Library	7.12.3	O	Crypto:MCF AND WAESpec-WMLS-C-001
WAESpec-SEC-C-003	Support for ESMP Crypto	7.12.5	O	WAESpec-ESMP-C-001 AND ESMPCR:MCF

A.1.3 User Agent Behaviour

Item	Function	Reference	Status	Requirement
WAESpec-UAB-C-001	Navigation History	7.13.1	M	
WAESpec-UAB-C-002	Access to Back key at all times	7.13.2	M	
WAESpec-UAB-C-003	User Agent performs pop operation on BACK	7.13.2	M	
WAESpec-UAB-C-004	WML1 User Agent executes `prev` task on BACK	7.13.2	O	

WAESpec-UAB-C-005	WML1 `do type=prev` behaviour	7.13.2	O	

Item	Function	Reference	Status	Requirement
WAESpec-WTA-C-001	Support for WTAI Public URI functions	7.14.1	O	WTAI-PU-C-001 AND WTAI-PU-C-002 AND WTAI-PU-C-003
WAESpec-WTA-C-002	Support for WTAI Public WMLScript functions	7.14.1	O	WTAI-PS-C-001 AND WTAI-PS-C-002 AND WTAI-PS-C-003 AND WMLScript:MCF

A.2 WAP proxy and Server

A.2.1 WAE Media Types

Item	Function	Reference	Status	Requirement
WAESpec-MT-S-001	Support for appropriate WAE MIME Media types	6.1	M	
WAESpec-MT-S-002	Proxy transparency	6.1	M	

Item	Function	Reference	Status	Requirement
WAESpec-WMLS-S-001	Ability to transform WMLScript into bytecode	6.4	M	WAESpec-SEC-S-001 AND WAESpec-EFI-S-001
WAESpec-WMLS-S-002	Informs WAE User Agents of compilation errors using the HTTP status code 502	6.4	O	

Item	Function	Reference	Status	Requirement
WAESpec-ESMP-S-001	Support for ECMAScript Mobile Profile	6.4.4	M	

Item	Function	Reference	Status	Requirement
WAESpec-MUL-S-001	convert application/vnd.wap.multipart.form-data into multipart/form-data	6.9.5	O	
WAESpec-MUL-S-002	convert application/vnd.wap.multipart.mixed into multipart/mixed	6.9.2	O	
WAESpec-MUL-S-003	convert application/vnd.wap.multipart.related into multipart/related	6.9.3	O	

WAESpec-MUL-S-004	convert application/vnd.wap.multipart.alternative into multipart/alternative	6.9.4	O	

A.2.2 WAE Features

Item	Function	Reference	Status	Requirement
WAESpec-HTS-S-001	Support for Hypermedia Transfer Service	7.1.1	M	WAESpec-HTS-S-002 OR WAESpec-HTS-S-003
WAESpec-HTS-S-002	Support for WSP	7.1.1	O	WSP:MSF AND WAESpec-HTS-S-004 AND WAESpec-MUL-S-002
WAESpec-HTS-S-003	Support for W-HTTP	7.1.1	O	W-HTTP:MSF
WAESpec-HTS-S-004	Client header handling in the connect service primitive	7.1.4	O	
WAESpec-HTS-S-005	Client header handling in the resume service primitive	7.1.4	O	

Item	Function	Reference	Status	Requirement
WAESpec-URI-S-001	Minimum URI length	7.2	M	
WAESpec-URI-S-002	HTTP URI Scheme	7.2.1	M	
WAESpec-URI-S-003	HTTPS URI Scheme	7.2.2	M	WAESpec-URI-S-004 OR WAESpec-URI-S-005 OR WAESpec-URI-S-006
WAESpec-URI-S-004	HTTPS URI Scheme over W-HTTP	7.2.2	O	TLS:MSF
WAESpec-URI-S-005	HTTPS URI Scheme over WSP	7.2.2	O	WTLS:MSF
WAESpec-URI-S-006	Report an error when no TLS or WTLS support is available	7.2.2	M	

Item	Function	Reference	Status	Requirement
WAESpec-MMS-S-001	Support for MMS	7.4	O	MMSCTR:MSF

Item	Function	Reference	Status	Requirement
WAESpec-I18N-S-001	Ability to transform character encoding	7.6.1	O	
WAESpec-I18N-S-002	Support for UTF-8	7.6.1	M	
WAESpec-I18N-S-003	Support for UTF-16	7.6.1	M	
WAESpec-I18N-S-004	Treat the character encoding of an XML document	7.6.1	M	

Item	Function	Reference	Status	Requirement
WAESpec-UAC-S-001	Support for Accept header	7.7.1	M	
WAESpec-UAC-S-002	Support for Accept-Charset header	7.7.1	M	
WAESpec-UAC-S-003	Support for Accept-Encoding header	7.7.1	M	
WAESpec-UAC-S-004	Support for Accept-Language header	7.7.1	M	
WAESpec-UAC-S-005	Support for the behaviour defined in HTTP/1.1	7.7.1	O	
WAESpec-UAC-S-006	Honouring the User Agent's preferences	7.7.1	O	
WAESpec-UAC-S-007	Avoidance of sending content that the User Agent does not accept	7.7.1	O	
WAESpec-UAC-S-008	Avoidance of unnecessary transcoding	7.7.1	M	

Item	Function	Reference	Status	Requirement
WAESpec-SEC-S-001	Ability to compile a script compilation unit that refers to the Crypto Library	7.12.3	M	WMLScriptCrypto:MSF

Item	Function	Reference	Status	Requirement
WAESpec-EFI-S-001	Ability to compile a script compilation unit that refers to the EFI Library	7.10	M	EFI:MSF

Appendix B. Change History (Informative)

B.1 Approved Version History

Reference	Date	Description
WAP-236-WAESpec-20011109-a	09-Nov-2001	Release level for WAP 2.0. Rollup of WAP-236_100-WAESpec-20011109-a SIN into WAESpec
WAP-236-WAESpec-20020207-a	07-Feb-2002	Update for SIN. Rollup of WAP-236_101-WAESpec-20020207-a SIN into WAESpec.
OMA-WAP-WAESpec-V2_2	20 Oct 2006	Approved by TP OMA Ref# OMA-TP-2006-0370R01-INP_Browsing_V2_2_for_Final_Approval.doc