WAP WAE

Proposed Version 3-Feb-1999

Wireless Application Protocol
Wireless Application Environment Overview

Disclaimer:

This document is subject to change without notice.

Contents

1. Scope

Wireless Application Environment (WAE) is a result of the Wireless Application Protocol (WAP) efforts to promote industry-wide standards and specifications for developing applications and services that operate over wireless communication networks. WAE specifies an application framework for wireless devices such as mobile telephones, pagers and PDAs. The framework extends and leverages other WAP technologies, including Wireless Transaction Protocol (WTP) and Wireless Session Protocol (WSP), as well as other Internet technologies such as XML, URLs, scripting and various content formats. The effort is aimed at enabling operators, manufacturers and content developers to meet the challenges of implementing advanced differentiating services and applications in a fast and flexible manner.

This document provides a general overview of the overall WAE architecture. Available WAE specifications are outlined and described in a subsequent section, "*The WAE Document Suite.*" For additional information on the WAP architecture, refer to *Wireless Application Protocol Architecture Specification* [WAP].

2. Document Status

This document is available online in the following formats:

- PDF format at http://www.wapforum.org/

2.1 Copyright Notice

2.2 Errata

Known problems associated with this document are published at http://www.wapforum.org/

2.3 Comments

Comments regarding this document can be submitted to the WAP Forum in the manner published at http://www.wapforum.org/

3. References

3.1 Normative References

[ISO10646] "Information Technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane", ISO/IEC 10646-1:1993.

[RFC2068] "Hypertext Transfer Protocol - HTTP/1.1", R. Fielding, et al., January 1997. URL: ftp://ftp.isi.edu/in-notes/rfc2068.txt

[RFC2396] "Uniform Resource Identifier (URI): Generic Syntax", T. Berners-Lee, et al., August 1998. URL: ftp://ftp.isi.edu/in-notes/rfc2396.txt

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

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

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

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

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

[WBXML] "WAP Binary XML Content Format", WAP Forum, April 30, 1998. URL: http://www.wapforum.org/

[WML] "Wireless Markup Language Specification", WAP Forum, April 30, 1998. URL: http://www.wapforum.org/

[WMLScript] "WMLScript Language Specification", WAP Forum, April 30, 1998. URL: http://www.wapforum.org/

[WMLStdLib] "WMLScript Standard Libraries Specification", WAP Forum, April 30, 1998. URL: http://www.wapforum.org/

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

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

[WTAI] "Wireless Telephony Application Interface Specification", WAP Forum, April 30, 1998. URL: http://www.wapforum.org/

[WTLS] "Wireless Transport Layer Security Specification", WAP Forum, April 30, 1998. URL: http://www.wapforum.org/

[WTP] "Wireless Transaction Protocol", WAP Forum, April 30, 1998. URL: http://www.wapforum.org/

[XML] "Extensible Markup Language (XML), W3C Proposed Recommendation 8-December-1997, PR-xml-971208", T. Bray, et al, December 8, 1997. URL: http://www.w3.org/TR/PR-xml

3.2 Informative References

[HDML2] "Handheld Device Markup Language Specification", P. King, et al., April 11, 1997. URL: http://www.uplanet.com/pub/hdml_w3c/hdml20-1.html

[HTML4] "HTML 4.0 Specification, W3C Recommendation 18-December-1997, REC-HTML40-971218", D. Raggett, et al., September 17, 1997. URL: http://www.w3.org/TR/REC-html40

[ISO8879] "Information Processing - Text and Office Systems - Standard Generalised Markup Language (SGML)", ISO 8879:1986.

[ECMASCRIPT] Standard ECMA-262: "ECMAScript Language Specification", ECMA, June 1997.

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

4. Definitions and Abbreviations

4.1 Definitions

The following are terms and conventions used throughout this specification.

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 (ie, instructions) and operands for a targeted hardware (or virtual) machine.

Card - a single WML unit of navigation and user interface. May contain information to present to the user, instructions for gathering user input, etc.

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 (eg, 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 - actual representation of content.

Content Generator - a service that generates or formats content. Typically content generators are on origin servers.

Deck - a collection of WML cards. A WML deck is also an XML document. May contain WMLScript.

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.

JavaScript - a *de facto* standard language that can be used to add dynamic behaviour to HTML documents. JavaScript is one of the originating technologies of ECMAScript.

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.

Peer-to-peer - direct communication between two terminals typically thought of as clients without involving an intermediate server. Also known as client-to-client communication.

Resource - A network data object or service that can be identified by a URL. Resources may be available in multiple representations (eg, 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.

SGML - the Standardised Generalised Markup Language (defined in [ISO8879]) is a general-purpose language for domain-specific mark-up languages.

Terminal - a device typically used by a user to request and receiving information. Also called a mobile terminal or mobile station.

Transcode - the act of converting from one character set to another, eg, conversion from UCS-2 to UTF-8.

User - a user is a person who interacts with a user agent to view, hear or otherwise use a resource.

User Agent - a user agent is any software or device that interprets content (eg, WML). This may include textual browsers, voice browsers, search engines, etc.

WMLScript - a scripting language used to program the mobile device. WMLScript is an extended subset of the JavaScript™ scripting language.

XML - the Extensible Markup Language is a World Wide Web Consortium (W3C) proposed standard for Internet mark-up languages, of which WML is one such language. XML is a restricted subset of SGML.

4.2 Abbreviations

For the purposes of this specification, the following abbreviations apply.

API	Application Programming Interface
BNF	Backus-Naur Form
CGI	Common Gateway Interface
ECMA	European Computer Manufacturers Association
ETSI	European Telecommunication Standardisation Institute
GSM	Global System for Mobile Communication
HDML	Handheld Markup Language [HDML2]
HTML	HyperText Markup Language [HTML4]
HTTP	HyperText Transfer Protocol [RFC2068]
IANA	Internet Assigned Number Authority
IMC	Internet Mail Consortium
LSB	Least Significant Bits
MMI	Man-Machine-Interface
MSB	Most Significant Bits
MSC	Mobile Switch Centre
PDA	Personal Digital Assistant
RFC	Request For Comments
SAP	Service Access Point
SGML	Standardised Generalised Markup Language [ISO8879]
SSL	Secure Socket Layer
TLS	Transport Layer Security
URI	Uniform Resource Identifier [RFC2396]
URL	Uniform Resource Locator [RFC2396]
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WAE	Wireless Application Environment
WAP	Wireless Application Protocol [WAP]
WBMP	Wireless BitMaP
WSP	Wireless Session Protocol [WSP]
WTA	Wireless Telephony Applications
WTAI	Wireless Telephony Applications Interface
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol
WWW	World Wide Web
XML	Extensible Markup Language

5. WAE Documentation

The following section outlines the set of WAE documentation available.

5.1 The WAE Document Suite

Several documents specify WAE:

- *Wireless Application Environment Specification* [WAE]:
  The Wireless Application Environment specification is the root document in the WAE normative document hierarchy. The document specifies and references core WAE elements.
- *Wireless Markup Language Specification* [WML]
  The Wireless Markup Language specification describes the mark-up language, WML, including its semantics, its document type definition (DTD) and its encoding extensions.
- *WAP Binary XML Format Specification* [WBXML]:
  The WAP Binary XML Forum specification describes the XML document encoding and transfer framework used by WAE.
- *WMLScript Specification* [WMLScript]:
  The WMLScript specification describes the scripting language, WMLScript, including its lexical and syntactic grammar, its transfer format and a reference bytecode interpreter.
- *WMLScript Standard Libraries Specification* [WAEStdLib]:
  The WMLScript Standard Libraries specification describes standard libraries available to WMLScript programs including a language library, a string library, a dialog library, a floating-point library, a browser library and a URL library.
- *Wireless Telephony Application Specification* [WTA]:
  The Wireless Telephony Application Specification specifies the technologies included in the Wireless Telephony Application reference architecture.
- *Wireless Telephony Application Interface* [WTAI]
  The Wireless Telephony Application Interface specification describes standard telephony-specific extensions to WAE including WML and WMLScript interfaces to such items as call control features, address book and phonebook services.

5.2 Document Organization

The remaining sections of this document present an overview of:

- WAE's history, goals, initial accomplishments and future plans;
- the major components of the WAE architecture (see the various WAE specifications for more details); and
- the major components of the WTA architecture (see [WTA] and [WTAI] for more details).

6. WAE Effort

The following section outlines the WAE effort including its background, initial and expected future directions.

6.1 Background

The WAE effort is an undertaking to build a general-purpose application environment based fundamentally on World Wide Web (WWW) technologies and philosophies. It is part of the overall WAP effort. The primary objective of the WAE effort is to establish an interoperable environment that will allow operators and service providers to build applications and services that can reach a wide variety of different wireless platforms in an efficient and useful manner.

The output of the WAE effort is a collection of technical specifications that are either new or based on existing and proven technologies. Among the existing technologies leveraged by the WAE effort are:

- Unwired Planet's Hand Held Mark-up Language (HDML),
- World Wide Web's Consortium's (W3C) Hypertext Mark-up Language (HTML),
- ECMA-262 Standard "ECMAScript Language Specification" [ECMASCRIPT] that is based on JavaScript™,
- IMC's calendar data exchange format (vCalendar) [VCAL] and phonebook data exchange format (vCard) [VCARD],
- A wide range of WWW technologies such as URLs and HTTP [RFC2068], along with
- A wide range of Mobile Network technologies such as GSM call control services and generic IS-136 services such as send flash.

The resulting WAE technologies are not fully compliant to all of the motivating technologies. Where necessary, modifications were made to better integrate the elements into a cohesive environment and better optimise the interaction and user interface for small screen limited capability terminals that communicate over wireless networks.

6.2 Direction

The main objectives of the WAE effort are:

1. To define an application architecture model:
   - That fits within the WAP architecture and meets WAP's overall objectives.
   - That is suitable for building interactive applications that function well on devices with limited capabilities including limited memory, small screen size, limited battery life and restricted input mechanisms.
   - That is suitable for building interactive applications that function well in narrow-band environments with medium-to-high latencies.
   - That employs appropriate security and access control features to allow safe execution of anonymous and third party content.
   - That leverages established and common standards and technologies that make WAE implementation simpler as well as allow third party developers to create and deploy applications inexpensively.
   - That is global and supports established internationalisation technologies and practices.
2. To define a general-purpose application programming model:
   - That is rich and enables interactive applications on current and future wireless devices.
   - That is based on the Internet's World-Wide-Web programming model including both browsing and scripting services.

- That provides access to common mobile device functionality and services such as phonebooks, messaging services and call control services.
- That enables applications to be accessible to a wide range of devices.
- That enables creation of applications that behave well on all WAP compliant devices.
- That allows developers to leverage specific functionality of specific devices.

3. To provide Network Operators the means to enhance and extend network services.
4. To enable multi-vendor interoperability.

6.2.1 Initial Phase Accomplishments

The initial focus of the WAE effort was the client. The following areas were established by the effort:

- A reference architecture definition.
- Lightweight mark-up and scripting language specifications.
- Encoding schemes for various content data such as mark-up documents, scripting programs, images, phonebook data and calendar data.
- Secure access mechanisms that third party content can use to access services local to the device.
- Generic interfaces to common local services such as messaging, phonebook and calendar services.
- Generic telephony-based interfaces to local services such as call control services.
- Network-specific extensions to telephony-based services.

6.2.2 Future Direction

While not formally established, future WAE efforts are expected in the following areas; the list is neither exhaustive nor prescriptive:

- Extensions to the defined languages as well as their encoding and transfer schemes (eg, vCard and vCalendar encoding, dynamic client-side content building, additional intrinsic events, user input validation, etc.).
- Advanced internationalisation issues (eg, multilingual and bi-directional text).
- New media types.
- Integration with other existing and emerging technologies (eg, Smart cards, SIM cards, Java Environments, etc.)
- Caching semantics.
- Document structure for capturing user agent capabilities.
- End-to-end security.
- Additional session schemes.
- Additional network specific telephony interface extensions such as IS-95.
- Server-side application level constructs.
- Integration and interfacing to intelligent networks and switching networks.

6.3 Goals and Requirements

The following list summarises the requirements of the Wireless Application Environment (WAE):

- WAE must enable simple yet efficient, meaningful and powerful application development and execution environments.

- WAE must provide a general framework. WAE cannot assume that a browser is the controlling agent in the device, nor can it assume that a browser is running at all times. Other applications may exist in the device. In which case, WAE must not prevent such applications from coexisting or even integrating with a browser. In addition, those other applications should be able to access and leverage common WAE services on the device where appropriate.
- WAE must not dictate or assume any particular Man-Machine-Interface (MMI) model. WAE implementations must be able to introduce new MMI models or use existing MMI models. Implementers must be able to present end users with a consistent and meaningful MMI suitable to the targeted device.
- WAE must be suited for a wide variety of limited capability devices. WAE must have a small memory footprint and limited computational power requirement. WAE must be suitable for the current generation of wireless devices without jeopardising its ability to evolve and support future generations of those devices.
- WAE must promote as well as incorporate efficient means to reduce the amount and frequency of over-the-air data exchanges with origin servers. WAE must provide the means to communicate device capabilities to origin servers, which would enable origin server-side optimisations and further minimise over-the-air resource consumption. In addition, WAE network services must be based on WAP's network protocol stack.
- WAE must support internationalisation and localisation using standard or well-accepted practices and methods.
- WAE must not compromise WAP's security model. WAE must include meaningful access control mechanisms that ensure secure processing of network accessed content.
- WAE must promote and enable interoperable implementation between various manufacturers and content or service providers.
- WAE must include extensions to allow means for call control and messaging, as well as enabling a standard set of value-added call and feature control capabilities.
- WAE must allow network operators to introduce new operator-specific features to their implementations.

7. WAE Architecture Overview

The WAE architecture includes all elements of the WAP architecture related to application specification and execution. At this point, the WAE architecture is predominately focused on the client-side aspects of WAP's system architecture, namely items relating to user agents. Specifically, the WAE architecture is defined primarily in terms of networking schemes, content formats, programming languages and shared services. Interfaces are not standardised and are specific to a particular implementation. This approach allows WAE to be implemented in a variety of ways without compromising interoperability or portability. This approach has worked particularly well with a browser (a class of user agents) model such as that used in the World-Wide-Web (WWW). The Internet and the WWW are the inspiration and motivation behind significant parts of the WAE specification, and consequently, a similar approach is used within WAE.

7.1 The WWW Model

The Internet's World Wide Web (WWW) provides a very flexible and powerful logical model. Applications present content to a client in a set of standard data formats that are *browsed* by client-side user agents known as Web browsers (or simply browsers). Typically, a user agent sends requests for one or more named data objects (or content) to an origin server. An origin server responds with the requested data expressed in one of the standard formats known to the user agent (eg, HTML)

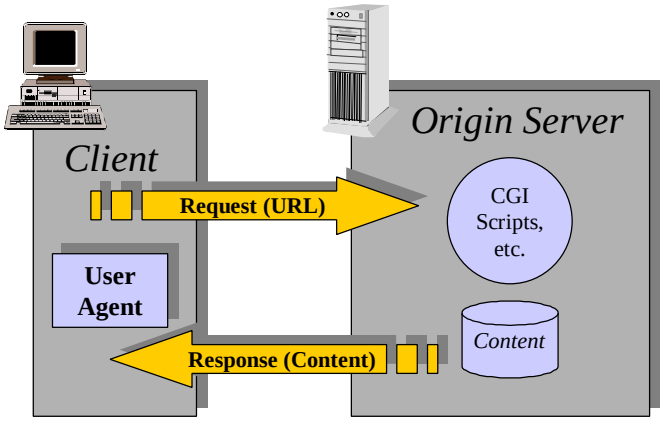

Figure 1: WWW Logical Model

The WWW standards include all of the mechanisms necessary to build a general-purpose environment:

- All resources on the WWW are named with Internet-standard *Uniform Resource Locators* (URLs).
- All classes of data on the WWW are given a specific type allowing the user agent to correctly distinguish and present them appropriately. Furthermore, the WWW defines a variety of standard content formats supported by most browser user agents. These include the Hypertext Mark-up Language (HTML), the JavaScript scripting language and a large number of other formats (eg, bitmap image formats).
- The WWW also defines a set of standard networking protocols allowing any browser to communicate with any origin server. One of the most commonly used protocols on the WWW today is the Hypertext Transport Protocol (HTTP).

The WWW infrastructure and model has allowed users to easily reach a large number of third party content and applications. It has allowed authors to easily deliver content and services to a large community of clients using various user agents (eg, Netscape Navigator ™ and Microsoft Internet Explorer ™.)

7.2 The WAE Model

WAE adopts a model that closely follows the WWW model. All content is specified in formats that are similar to the standard Internet formats. Content is transported using standard protocols in the WWW domain and an optimised HTTP-like protocol in the wireless domain. WAE has borrowed from WWW standards including authoring and publishing methods wherever possible. The WAE architecture allows all content and services to be hosted on standard Web origin servers that can be incorporate proven technologies (eg, CGI). All content is located using WWW standard URLs.

WAE enhances some of the WWW standards in ways that reflect the device and network characteristics. WAE extensions are added to support Mobile Network Services such as Call Control and Messaging. Careful attention is paid to the memory and CPU processing constraints that are found in mobile terminals. Support for low bandwidth and high latency networks is included in the architecture as well.

WAE assumes the existence of *gateway* functionality responsible for encoding and decoding data transferred from and to the mobile client. The purpose of encoding content delivered to the client is to minimise the size of data sent to the client over-the-air as well as to minimise the computational energy required by the client to process that data. The gateway functionality can be added to origin servers or placed in dedicated gateways as illustrated in Figure 2.

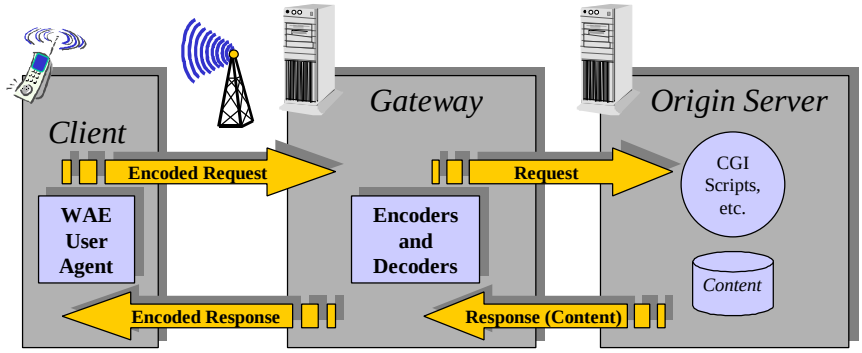

Figure 2: WAE Logical Model

The major elements of the WAE model include:

- *WAE User Agents*[1]*:*
  Client-side in-device software that provides specific functionality (eg, display content) to the end-user. User agents (such as browsers) are integrated into the WAP architecture. They interpret network content referenced by a URL. WAE includes user agents for the two primary standard contents: encoded Wireless Markup Language (WML) and compiled Wireless Markup Language Script (WMLScript.)
- *Content Generators:*
  Applications (or services) on origin servers (eg, CGI scripts) that produce standard content formats in response to requests from user agents in the mobile terminal. WAE does not specify any standard content generators but expects that there will be a great variety available running on typical HTTP origin servers commonly used in WWW today.
- *Standard Content Encoding:*
  A set of well-defined content encoding, allowing a WAE user agent (eg, a browser) to conveniently navigate web content. Standard content encoding includes compressed encoding for WML, bytecode encoding for WMLScript, standard image formats, a multi-part container format and adopted business and calendar data formats.

---

1 Except where noted, throughout this document, the term WAE user agent is used as a general term to denote any user agent that incorporates some or all of WAE's defined services. The document distinguishes types of user agents only where needed or appropriate.

- *Wireless Telephony Applications (WTA):*
  A collection of telephony specific extensions for call and feature control mechanisms that provide authors (and ultimately end-users) advanced Mobile Network Services.

The resulting WAE architecture fits within a model:

- That leverages the Internet (ie, the model takes advantage of standards, technology and infrastructure developed for the Internet),
- That leverages thin-client architecture advantages (eg, service deployment has significantly lower cost per device due to the device independent nature of WAE and the centralised management of the services at the origin servers),
- That provides end user advanced Mobile Network Services through Network Operator controlled telephony value-added services,
- That provides the means for vendors to build differentiating user-friendly services that can take advantage of WWW and Mobile Network Services and
- That provides an open extensible framework for building wireless services.

Typically, a user agent on the terminal initiates a request for content. However, not all content delivered to the terminal will result from a terminal-side request. For example, WTA includes mechanisms that allow origin servers to deliver generated content to the terminal without a terminal's request as illustrated in Figure 3.

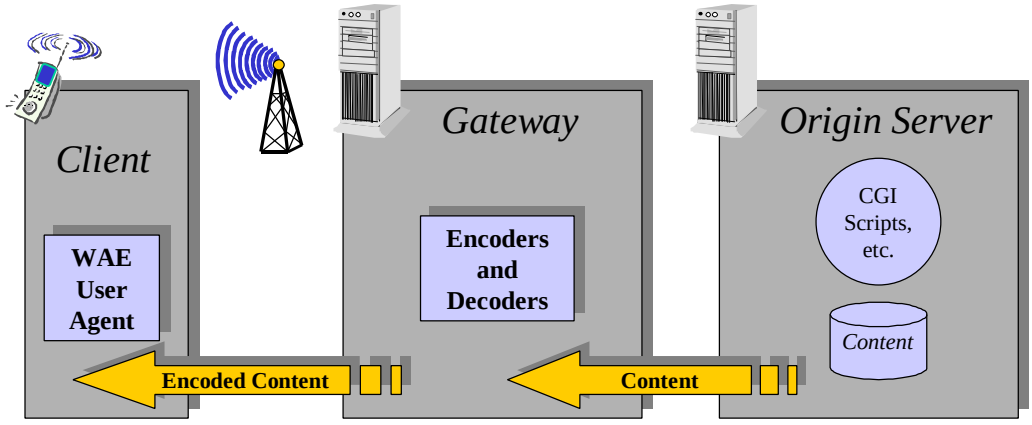

Figure 3: WAE Push-based Model

In some cases, what the origin server delivers to the device may depend on the characteristics of the device. The user agent characteristics are communicated to the server via standard capability negotiation mechanisms that allows applications on the origin server to determine characteristics of the mobile terminal device. WAE defines a set of user agent capabilities that will be exchanged using WSP mechanisms. These capabilities include such global device characteristics as WML version supported, WMLScript version supported, floating-point support, image formats supported and so on.

7.3 URL Naming

WAE architecture relies heavily on WWW's URL and HTTP semantics. WAE assumes:

- the existence of a generalised architecture for describing gateway behaviour for different types of URLs and
- support for connection to at least one WAP gateways.

In particular, the URL naming mechanisms used in WAE are motivated by the following scenarios:

- A secure service (eg, banking or brokerage), where an end-to-end secure connection using WTLS [WTLS] is necessary mandating a secure gateway controlled by the content provider.
- A content provider who wants to provide a caching gateway that will cache the encoded content in order to improve performance.
- A specialised service with a built-in server that will be accessible only to WAP devices and, therefore, wants to use WSP natively rather than incur the higher overhead of processing HTTP sessions.

WAE is based on the architecture used for WWW proxy servers. The situation where a user agent (eg, a browser) must connect through a proxy to reach an origin server (ie, the server that contains the desired content) is very similar to the case of a wireless device accessing a server through a gateway.

Most connections between the browser and the gateway use WSP, regardless of the protocol of the destination server. The URL, used to distinguish the desired content, always specifies the protocol used by the destination server regardless of the protocol used by the browser to connect to the gateway. In other words, the URL refers only to the destination server's protocol and has no bearing on what protocols may be used in intervening connections.

In addition to performing protocol conversion by translating requests from WSP into other protocols and the responses back into WSP, the gateway also performs content conversion. This is analogous to HTML/HTTP proxies available on the Web today. For example, when an HTTP proxy receives an FTP or Gopher directory list, it converts the list into an HTML document that presents the information in a form acceptable to the browser. This conversion is analogous to the encoding of content destined to WAE user agents on mobile devices.

Currently only one scheme is expected to be supported by WAE user agents:

- **http:** The browser, in this case, communicates with the gateway using WSP. The gateway in turn would provide protocol conversion functions to connect to an HTTP origin server.

As an example, a user, with a WAP-compliant telephone, requests content using a specific URL. The telephone browser connects to the operator-controlled gateway with WSP and sends a GET request with that URL. The gateway resolves the host address specified by the URL and creates an HTTP session to that host. The gateway performs a request for the content specified by the URL. The HTTP server at the contacted host processes the request and sends a reply (eg, the requested content). The gateway receives the content, encodes it, and returns it to the browser[2].

7.4 Components of WAE

As illustrated in Figure 4, WAE is divided into two logical layers:

- User agents, which includes such items as browsers, phonebooks, message editors, etc; and
- Services and Formats, which include common elements and formats accessible to user agents such as WML, WMLScript, image formats, vCard and vCalendar formats, etc.

WAE separates services from user agents and assumes an environment with multiple user agents. This logical view, however, does not imply or suggest an implementation. For example, WAE implementations may choose to combine all the services into a single user agent. Others, on the other hand, may choose to distribute the services among several user agents. The resulting structure of a WAE implementation is determined by the design decisions of its implementers and should be guided by the specific constraints and objectives of the target environment.

---

2 Encoding may not be necessary at the gateway in all cases. WAP Architecture supports other scenarios. See [WSP] for additional information.

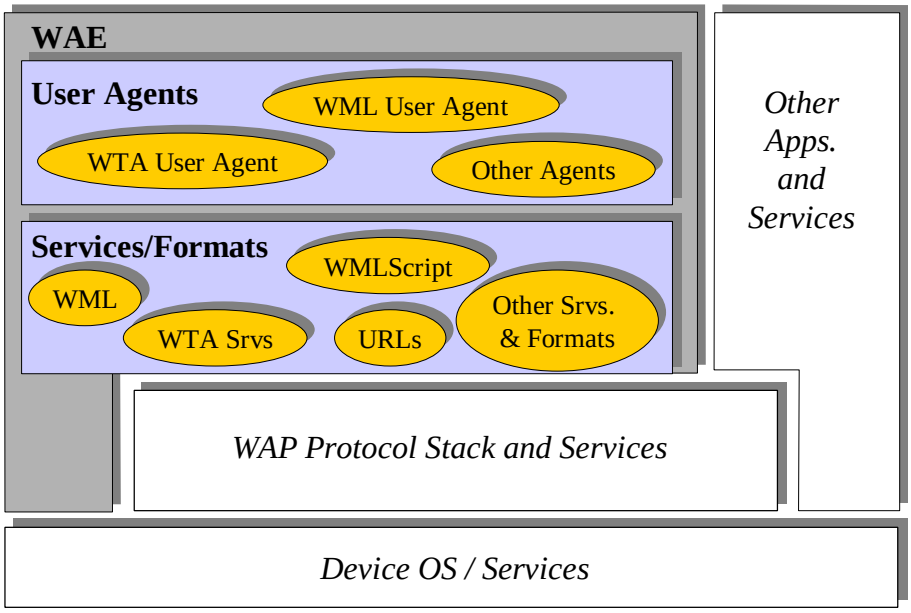

Figure 4: WAE Client Components

7.4.1 WAE User Agents

The WML user agent[3] is a fundamental user agent of the WAE. However, WAE is not limited to a WML user agent. WAE allows the integration of domain-specific user agents with varying architectures and environments. In particular, a Wireless Telephony Application (WTA) user agent has been specified as an extension to the WAE specification for the mobile telephony environments. The WTA extensions allow authors to access and interact with mobile telephone features (eg, call control) as well as other applications assumed on the telephones, such as phonebooks and calendar applications[4]. An overview of the WTA architecture is presented in a subsequent section, *WTA Architecture Overview.*

WAE does not formally specify any user agent. Features and capabilities of a user agent are left to the implementers. Instead, WAE only defines fundamental services and formats that are needed to ensure interoperability among implementations. An overview of those services and formats is included in subsequent sections.

7.4.2 WAE Services and Formats

The WAE Services and Formats layer includes the bulk of technical contribution of the WAE effort. The following section provides an overview of the major components of WAE including the Wireless Markup Language (WML), the Wireless Markup Scripting language (WMLScript), WAE applications and WAE supported content formats.

7.4.2.1 WML

WML is a tag-based document language. In particular, it is an application of a generalised mark-up language. WML shares a heritage with the WWW's HTML[HTML4] and Handheld Device Markup Language [HDML2]. WML is specified as an XML [XML] document type. It is optimised for specifying presentation and user interaction on limited capability devices such as telephones and other wireless mobile terminals.

WML and its supporting environment were designed with certain small narrow-band device constraints in mind including small displays, limited user-input facilities, narrow band network connections, limited memory resources and

---

3 Except where noted, throughout this document, a WML user agent (or WML browser) refers to a basic user agent that supports WML, WMLScript, or both. It does not necessarily indicate if the user agent supports both WML and WMLScript or only one of them.

4 Such applications (eg, phonebook and calendar applications) are not specified by WAE.

limited computational resources. Given the wide and varying range of terminals targeted by WAP, considerable effort was put into the proper distribution of presentation responsibility between the author and the browser implementation.

WML is based on a subset of HDML version 2.0 [HDML2]. WML changes some elements adopted from HDML and introduces new elements, some of which have been modelled on similar elements in HTML. The resulting WML implements a card and deck metaphor. It contains constructs allowing the application to specify documents made up of multiple *cards*. An interaction with the user is described in a set of cards, which can be grouped together into a document (commonly referred to as a *deck*). Logically, a user navigates through a set of WML cards. The user navigates to a card, reviews its contents, may enter requested information, may make choices, and then moves on to another card. Instructions imbedded within cards may invoke services on origin servers as needed by the particular interaction. Decks are fetched from origin servers as needed. WML decks can be stored in 'static' files on an origin server, or they can be dynamically generated by a content generator running on an origin server. Each card, in a deck, contains a specification for a particular user interaction.

WML is specified in a way that allows presentation on a wide variety of devices yet allows for vendors to incorporate their own MMIs. For example, WML does not specify *how* implementations request input from a user. Instead, WML specifies the *intent* in an abstract manner. This allows WML to be implemented on a wide variety of input devices and mechanisms. Implementations may, for example, choose to solicit user input visually like many WWW user agents, or it may choose to use a voice-based interface. The user agent must decide how to best present all elements within a card depending on the device capabilities. For example, certain user agents on devices with larger displays may choose to present all the information in a single card at once. Others, on the other hand, with smaller displays may break the content up across several units of displays.

WML has a wide variety of features, including:

- *Support for Text and Images*
  WML provides the authors with means to specify text and images to be presented to the user. This may include layout and presentation hints. As with other mark-up languages, WML requires the author to specify the presentation in very general terms and gives the user agent a great deal of freedom to determine exactly how the information is presented to the end user. WML provides a set of text mark-up elements including various *emphasis* elements (eg, bold, italic, big, etc.); various *line breaks* models (eg, line wrapping, line wrapping suppression, etc.); and *tab columns* that supports simple tabbing alignment.

- *Support for User Input*
  WML supports several elements to solicit user input. The elements can be combined into one or more cards. All requests for user input are made in abstract terms, allowing the user agent the freedom to optimise features for the particular device. WML includes a small set of input controls. For example, WML includes a *text entry* control that supports text and password entry. Text entry fields can be masked preventing 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. WML also includes *task invocation* controls. When activated, these controls initiate a navigation or a history management task such as traversing a link to another card (or script) or popping the current card off of the history stack. The user agent is free to choose how to present these controls. It may for example, bind them to physical keys on the device, render button controls in a particular region of the screen (or inline within the text), bind them to voice commands, etc.

- *Navigation and History Stack*
  WML allows several navigation mechanisms using URLs. It also exposes a first-class history mechanism. Navigation includes HTML-style hyperlinks, inter-card navigation elements, as well as history navigation elements.

- *International Support*
  WML's document character set is Unicode [UNICODE]. This enables the presentation of most languages and dialects.

- *MMI Independence*
  WML's abstract specification of layout and presentation enables terminal and device vendors to control the MMI design for their particular products.

- *Narrow-band Optimisation*
  WML includes a variety of technologies to optimise communication on a narrow-band device. This includes the ability to specify multiple user interactions (cards) in one network transfer (a deck). It also includes a variety of state management facilities that minimise the need for origin server requests. WML includes other mechanisms to help improve response time and minimise the amount of data exchanged over-the-air. For example, WML allows the author to parameterise (or pass variables to) a subsequent context. It supports variable substitution and provides out-of-band mechanisms for client-side variable passing without having to alter URLs. The out-of-band passing of variables without changing the way URLs appear attempts to improve client-side cache hits.

- *State and Context Management*
  WML exposes a flat context (ie, a linear non-nested context) to the author. Each WML input control can introduce variables. 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 deck and can be shared across multiple decks without having to use a server to save intermediate state between deck invocations.

7.4.2.2 WMLScript

WMLScript is a lightweight procedural scripting language. It enhances the standard browsing and presentation facilities of WML with behavioural capabilities, supports more advanced UI behaviour, adds intelligence to the client, provides 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 JavaScript™ WWW scripting language. It is an extended subset of JavaScript™ and forms a standard means for adding procedural logic to WML decks. WMLScript refines JavaScript™ for the narrow-band device, integrates it with WML, and provides hooks for integrating future services and in-device applications.

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

- The ability to check the validity of user 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 (eg, display an error message).

Key WMLScript features include:

- *JavaScript™-based scripting language:*
  WMLScript starts with an industry standard solution and adapts it to the narrow-band environment. This makes WMLScript very easy for a developer to learn and use.
- *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 space efficient bytecode that is transported to the client.
- *Integrated into WAE:*
  WMLScript is fully integrated with the WML browser. This allows the author to construct their services using

both technologies, using the most appropriate solution for the task at hand[5]. WMLScript has access to the WML state model and can set and get WML variables. This enables a variety of functionality (eg, validation of user input collected by a WML card).

- *International Support*
  WMLScript character set is Unicode [UNICODE]. This enables the presentation of most languages and dialects.
- *Efficient extensible library support:*
  WMLScript can be used to expose and extend device functionality without changes to the device software.

One objective in designing the WMLScript language was to be close to core 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 WMLScript language. The resulting WMLScript is a weakly typed language. Variables in the language are not formally typed in that a variable's type may change throughout the lifecycle of the variable depending on the data it contains. The following basic data types are supported: *boolean*, *integer*, *floating-point*, *string and invalid*. WMLScript attempts to automatically convert between the different types as needed. In additions, support for floating-point data types may vary depending on the capabilities of the target device.

WMLScript supports several categories of operations such as assignment operations, arithmetic operations, logical operations and comparison operations. WMLScript supports several categories of functions including *Local script functions* (ie, script functions defined inside the same script that the calling expression is in), *External script functions* (ie, script functions defined in another script not containing the calling expression) and *Standard library functions* (ie, 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.

7.4.2.3 URLs

WAE assumes a rich set of URL services that user agents can use. In particular, WAE relies heavily on HTTP and HTML URL semantics. In some cases, WAE components extend the URL semantics, such as in WML, where URL fragments has been extended to allow linking to particular WMLScript functions.

7.4.2.4 WAE Content Formats

WAE includes a set of agreed upon content formats that facilitate interoperable data exchange. The method of exchange depends on the data and the targeted WAE user agents. The two most important formats defined in WAE are the encoded WML and the WMLScript bytecode formats. WAE defines WML and WMLScript encoding formats that make transmission of WML and WMLScript more efficient as well as minimises the computational efforts needed on the client.

In addition, WAE defines and adopts other formats for data types including:

- *Images:*
  WAE assumes visual environments that support images will support several image formats. The selection of formats was an attempt to meet several competing requirements including: support of multiple choices of pixel depth, support of colorspace tables, small encoding, very low CPU and RAM decoding and presentation demands and availability of common tools and other developer support.
- *Multipart Messages:*
  WAE leverages a multipart-encoding scheme optimised for exchanging multiple typed content over WSP. See [WSP] for additional details.
- *User Agent-Specific Formats:*
  WAE adopts two additional content formats specific to exchanging data among user agents suitable for both client-server communication and peer-to-peer communication: *electronic business cards* (vCard.2.1) and *electronic*

---

5 A WAE user agent does not have to use both WML and WMLScript. Both are loosely coupled and can be used independently.

calendaring and scheduling exchange format (vCalendar 1.0) specified by the IMC. See [WAE].

WAE also defines WTA-specific formats that are previewed in subsequent sections and defined in [WTA].

7.5 WML and WMLScript Exchanges

The following figure, Figure 5, presents the different parts of the logical architecture assumed by a WML user agent:

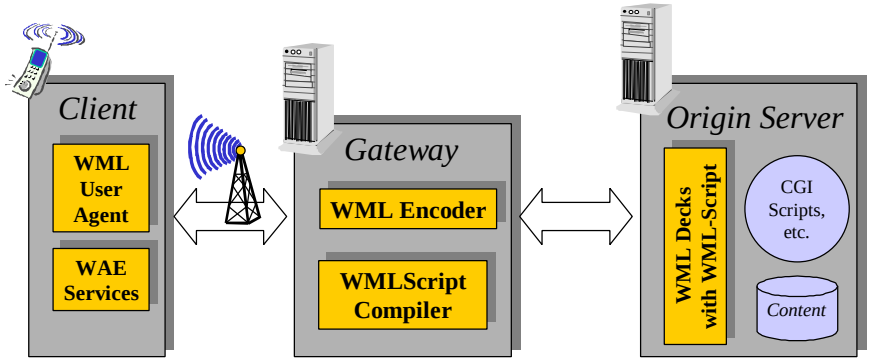

Figure 5: WML User Agent Logical Architecture

Origin servers provide application services to the end user. The service interaction between the end user and the origin server is packaged as standard WML decks and scripts. Services may rely on decks and scripts that are statically stored on the origin server, or they may rely on content produced dynamically by an application on the origin servers.

Several stages are involved when origin servers and WML user agents exchange WML and WMLScript. In particular, a user, wishing to access a particular service from an origin server, submits a request to the origin server using a WML user agent. The user agent requests the service from the origin server on behalf of the user using some URL scheme operation (eg, HTTP GET request method.)

The origin server honouring the user's request replies by sending back a single deck. Presumably this deck is initially in a textual format. On their way back to the client, textual decks are expected to pass through a gateway where they are converted into formats better suited for over-the-air transmission and limited device processing. In principle, once the gateway receives the deck from the origin server, the gateway does all the necessary conversions between the textual and binary formats. A WML encoder (or tokeniser) in the gateway converts each WML deck into its binary format. Encoded content is then sent to the client to be displayed and interpreted. Some optimisation may be done at the gateway based on any negotiated features with the client.

The user agent may submit one or more additional requests (using some URL scheme) for WMLScript as the user agent encounters references to them in a WML deck. On its way back, a WMLScript compiler takes the script as input and compiles it into bytecode that is designed for low bandwidth and thin mobile clients. The compiled bytecode is then sent to the client for interpretation and execution.

The existence of a gateway is not mandatory as illustrated in Figure 6. In particular, the location where the actual encoding and compilation is done is not of particular concern to WAE. It is conceivable that some origin servers will have built-in WML encoders and WMLScript compilers. It may also be possible, in certain cases, to statically store (or cache) particular services in tokenised WML and WMLScript bytecode formats eliminating the need to perform any on-the-fly conversion of the deck.

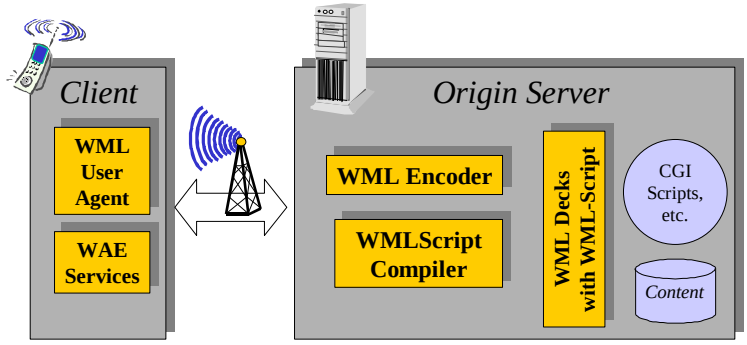

Figure 6: WML User Agent Logical Architecture (without a Gateway)

7.6 Internationalisation

The WAE architecture is designed to support mobile terminals and network applications using a variety of languages and character sets. This work is collectively described as internationalisation (referred to as I18N). It is a design goal of WAE to be fully global in its nature in that it supports any language.

WAE models a significant amount of its I18N architecture based on WWW and, in particular, on SGML and HTTP technologies. For example, it is assumed that HTTP headers are used to specify the current character encoding and language of any content delivered to the user agent.

The WAE architecture makes the following assumptions regarding I18N:

- WAE user agents will have a current language and will accept content in a set of well-known character encoding sets.
- Origin server-side applications can emit content in one or more encoding set and can accept input from the user agent in one or more encoding set.

The IANA registry of character sets and languages is used to define the encoding and language characteristics (see http://www.iana.org/iana/ for more information.) Content fetched by a WAE user agent via WSP will be described by two attributes: the *Character set* (ie, the encoding used in the content) and the *Language* (ie, the default language for the document.) These attributes are encapsulated in the WSP/HTTP *Content-Type* and *Content-Language* headers.

WAE has adopted the [ISO10646] (Unicode 2.0) standard as the basis for all character data. Unicode contains the majority of the characters and symbols present in human languages and is widely supported in the Internet community. WAE does not mandate which character encoding a user agent must support.

Most components of WAE contain I18N-specific support. For example:

- WML contains additional support allowing the user agent and the origin server to negotiate the transmission encoding of user input sent from the user agent to the server (eg, the *Accept-Charset* attribute).
- WMLScript defines string manipulation functionality to use Unicode collation order.
- All content and data types are transmitted via protocols, which support the declaration of language and encoding, or they include such information in the data format itself.

7.7 Security and Access Control

WAE leverages WTLS where services require authenticated and/or secure exchanges. In addition, 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 decks and WMLScript to grant public access to the content (ie, the deck or script can be referenced from other content) or restrict access to the content to set of “trusted” decks or scripts.

8. WTA Architecture Overview

WTA is a collection of telephony specific extensions for call and feature control mechanisms that make advanced Mobile Network Services available to authors and end-users[6]. WTA merges the features and services of data networks with the services of voice networks. It introduces mechanisms that ensure secure access to important resources within mobile devices. The WTA framework allows real-time processing of events important to the end-user while browsing. Within the WTA framework, the client and server co-ordinate the set of rules that govern event handling via an event table. WTA origin servers can adjust the client's rules by pushing (or updating) a client's event table if required as defined in [WTA].

The Wireless Telephony Application Framework has four main goals:

- Enable Network Operators to provide advanced telephony services that are well integrated and have consistent user interfaces.
- Enable Network Operators to create customised content to increase demands and accessibility for various services in their Networks.
- Enable Network Operators to reach a wider range of devices by leveraging generic WAE features that allow the operator to create content independent of device specific characteristics and environments.
- Enable third party developers to create network-independent content that access basic features (ie, non-privileged).

Most of the WTA functionality is reserved for the Network Operators, as in-depth knowledge and access to the mobile network are needed to fully take advantage of the mobile network's features. Nevertheless, a limited set of basic WTA functions, such as initiating phone calls, is available to all WTA authors[7].

8.1 WTA Framework Components

The following sections describe the key components of the WTA framework.

8.1.1 WTA Libraries

WTA exposes its services to content authors as a set of libraries and interfaces. WTA functionality is divided across several libraries according to its sensitivity and applicability. WTA defines three classes of WTA services[8]:

- *Common Network Services*
  WTA services that are available independent of network type. They are common to all networks (eg, answering an incoming call.) Access to these services is restricted to content running within a WTA user agent.
- *Network Specific Services*
  WTA services that target a specific type of network. These services are extensions to Common Network Services that expose unique and common features of a particular type of network (eg, IS-136 includes a *Send Flash* service.) Like Common Network Services, these services are restricted to content running in a WTA user agent.

---

6 Mobile network features, like "Roaming", are transparent to application developers and not exposed by WTA.

7 For the most part, WTA functionality extends typical WML and WMLScript functionality by introducing library extensions. Some of these library extensions (in particular, non-privileged functions) can be included in any WAE user agent. User agents announce their support of such features using standard capability negotiation mechanisms. See [WAE] for additional information.

8 Operator-specific services may be added in any implementation. These services, however, are out of the scope of the WAP effort.

- *Public Services*
  WTA services are available to any anonymous or third party content (eg, initiate call set-up.) There are no access restrictions on such services. Any user agent is free to access Public Services.

Classifying and separating services enables secure and reliable execution of content. It limits functions available to authors and developers at-large.

Access to WTA services can be done directly from either the WML language, using the WTAI URL scheme or from WMLScript functions by calling WTAI library functions.

8.1.2 WTA URL Scheme

WTA introduces a URL scheme that allows authors to invoke library services. The services may reside on the device or may be delegated[9] to a server. Using this scheme, authors can pass data to a service and receive data back from the services without having to leave the current browsing context. See [WTA] for a complete specification.

8.1.3 WTA Event Handling

The WTA framework provides a variety of means for authors to deal with telephony-based events in real-time and pseudo-real-time[10] manners. Fundamentally, telephony-based events can be sent to the WTA agent along with any required event-specific parameters and content. This allows Network Operator to deploy content (eg, decks) with call control and network event handling aspects. Clients can maintain event tables that describe how a user agent should deal with incoming events. This event table is co-ordinated with a WTA origin server controlled by the network operator.

For the most part, content sent with the event (or content already residing on the client) will be sufficient to handle most events. However, the framework does not prevent more advanced scenarios that require additional content to be retrieved from an origin server based on end-user demand. How a Network Operator chooses to handle events depends largely on the type of events, reliability and latency requirements and quality desired.

8.1.4 WTA Network Security

The Operator is assumed to have control as to what resources are be made accessible to any anonymous or third party content in both the mobile network and the client. The integrity of the mobile network and the client are enforced because of a restricted WTA content delivery. In particular, content with privileged WTA services can only be executed when it is delivered to the WTA user agent through a dedicated WTA port running WTLS protocols. This allows network operators to use standard network security elements to protect their networks. For example, origin servers, delivering content, can be identified, by the operator, as either trusted WTA content servers that are under the control of the device's operator, or as untrusted *third party* content servers, which may include any public origin server on the Internet. Network operators can then use standard firewall technologies to regulate access to a mobile's ports. Port access, can then be used to determine the credentials given to content, which determines its access privileges to WTA services in both the network and the client.

8.2 Telephony-Specific Exchanges

WTA user agents, defined by WAE as telephony-specific extensions, use similar exchange constructs as a WML user agent. However, WTA user agents rely on additional and extended interactions needed to deliver meaningful telephony-based services.

---

9 How a service request is handled is hidden from the author. The author does not know at the time of the request where the request will be resolved.

10 Developers need to be aware of the constraints of handling real-time scenarios in the particular network when authoring content that relies on the communication between the server and the client.

The elements of the logical WTA network, presented in the following figure, are:

- Content and Content Generators
- Firewalls (optional)
- Mobile Switching Framework

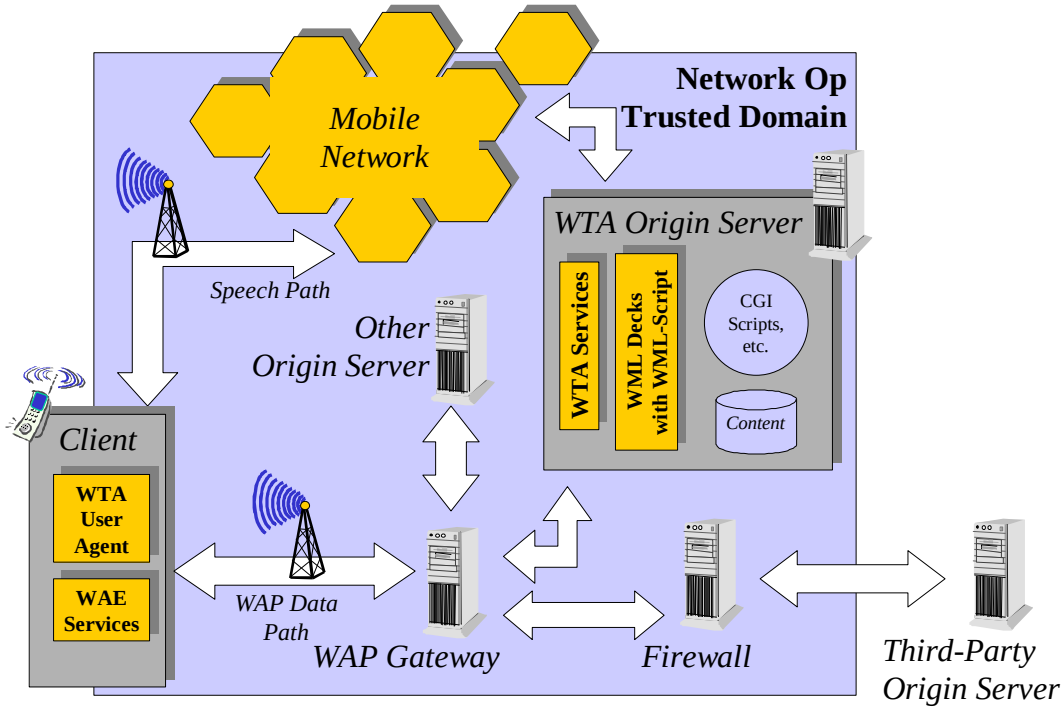

Figure 7: WTA Logical Architecture

The WTA user agent is connected to the mobile network using dedicated signalling connections. The WTA Server (an origin server) communicates with the client using the WAP protocol stack. The WTA server may be connected to the mobile network and is responsible for deploying content to its clients. In the case of call handling, for example, the mobile network sets up the call to the client, the server delivers the event-handling content, and the user agent invokes the event-handler content and manages the presentation of the call handling service to the user.

The WTA user agent is a content interpreter that extends a typical WML user agent. It supports extended libraries and executes WML decks and WMLScript similar to a WML user agent. However, unlike a typical WML user agent, the WTA user agent has a very rigid and real-time context management component. For example, the user agent drops outdated (or stale) events, does not place intermediate results on the history stack, and typically terminates after the event is handled.

8.2.1 WTA Origin Servers

The WTA Origin Server is assumed to be under the control of the Network Operator and is therefore to be regarded as a "Trusted Content Server". The operator's server is assumed to *control*, in varying degrees, the Mobile Network Switch. The success of the WTA content (eg, handling Call Control) is, to some extent, dependent on the operator's ability to access and control the features and characteristics of the Mobile Network.

The operator has information about latency, capacity and reliability for the different bearers in the Mobile Network. Since the operator is able to provide the WTA services without needing to rely on the Internet, the operator can have

more control over the behaviour of the services than a third party service provider can, and can better optimise their services to achieve good real-time characteristics.

8.2.2 Third Party Origin Servers

Content from third party providers does not handle any extensive set of WTA functions. Developing advanced WTA applications requires, in most cases, in-depth knowledge of the mobile network. Due to the limitations imposed by the operator as to which third party is granted access to the mobile network resources and WTA services, third party content providers are limited to handling WAE content using the subscriber's standard WML user agent.

8.2.3 Mobile Network

The network operator controls the mobile network. The mobile network handles switching and call set-up to the mobile subscribers (or terminals). The mobile network connects with the client using in-band or out-of-band signalling connections. The mobile network-to-client signalling is exposed to the content running in a user agent using WTA network events. Even though the mobile network-to-client messaging uses network-level system-specific signalling, at the content level, signalling is converted to more generic and abstract WTA network events.

Although the mobile network is involved in the execution of the WTA network services, operations and services in the mobile network are not within the scope of the WAP effort. WTA services only make assumptions on the availability of basic network features like call set-up, call accept, etc. Call control features in the mobile network are made available to the WTA user agent through the device's WTA interface.