WAP Architecture Draft Version 0.9 (1997-09)

Wireless Application Protocol Architecture Specification

Disclaimer:

This document is a preliminary draft document of a Wireless Application Protocol (WAP) architecture specification. It is subject to change. The document is being published to solicit feedback from interested parties on the current state of a proposed WAP architecture specification.

This document does not necessarily reflect any specification that will become a published WAP standard.

Contents

1. Scope

Wireless Application Protocol (WAP) is a result of continuous work to define an industry wide standard for developing applications over wireless communication networks. The scope for the WAP working group is to define a set of standards 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.

The upper layers of WAP will be independent of the underlying wireless network, while the transport layer might be adapted to specific features of underlying bearers. However, by keeping the transport layer interface, as well as the basic features, consistent global interoperability can be achieved using mediating gateways.

Scaleability is one of the most important issues in the WAP work. This includes both device scaleability as well as network scaleability. The framework (and applications) is suitable for both one-line display phones as well as advanced PDA devices. It fits very slow bearers as well as medium bandwidth bearers.

The WAP architecture is designed to be extensible and future proof. New bearers, like packet radio, can be added when available, and the applications can automatically benefit from this new environment. The layered architecture allows for future enhancements in each individual layer, with non or small changes to adjacent layers. Thus WAP will protect investments in servers and application software, and provide a stable protocol for ever better and more efficient applications.

2. Goals and requirements

The goals and requirements of the Wireless Application Protocol (WAP) can be summarized as follows:

- WAP should provide access to Internet, Intranet, and operator services. It should be leveraged on existing standards where possible.
- Access to local handset functionality must be available.
- The architecture must be layered, scaleable and extensible.
- The Man Machine Interface (MMI) must provide maximum flexibility and be vendor controlled.
- Optimized for narrow-band bearers with potentially high latency.
- Optimized for efficient use of device resources (low memory/CPU usage)
- Support for layered security is strongly desired.
- Architecture must support several types of wireless networks.
- To facilitate network-operator and 3rd party service provision to wireless devices.
- To define a general purpose application programming model to deliver services to wireless devices, such as phones.
- To define an architecture to support multi-vendor interoperability, defining the optional and mandatory components.

3. Architecture Overview

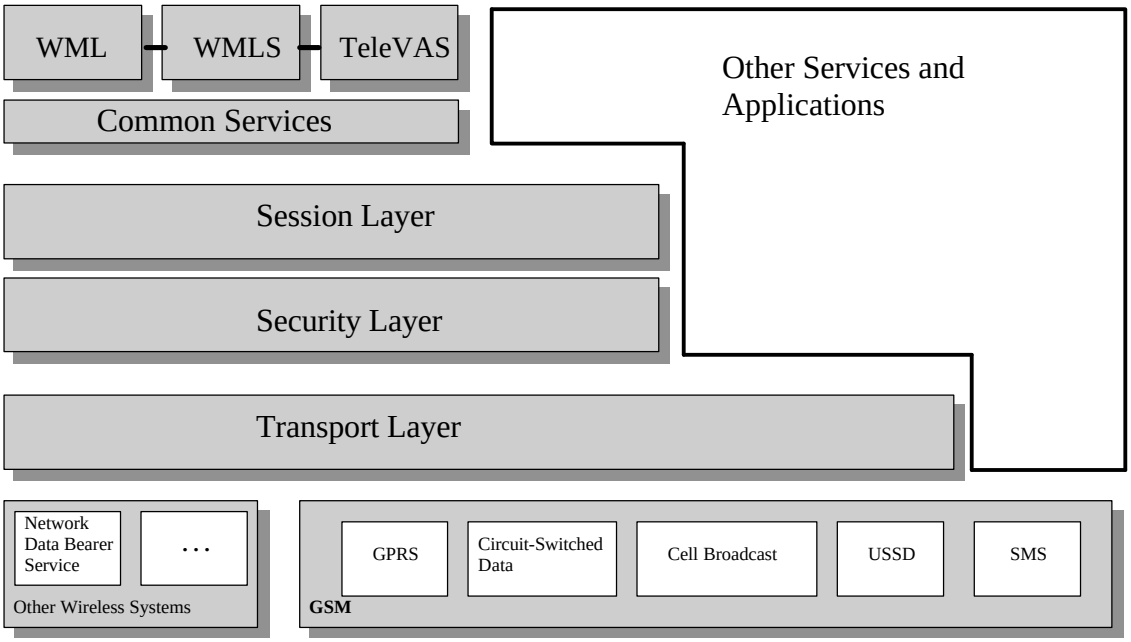

Figure 1: The WAP Architecture and External Interfaces

3.1 The WAP Architecture

The WAP architecture provides a scaleable and extensible environment for application development for mobile communication devices. This is achieved through a layered design of the network to application communication protocol stack. Each of the layers of the architecture are accessible by the layers above, as well as, by other service and applications

The WAP architecture enables any number of services and applications to utilize the features of the WAP stack. The applications specified in the WAP framework may utilize the common services layer internal to the application layer, while external applications may access the session and transport layers directly. Direct access from external applications to the security layer is currently optional.

3.2 WAP Compliant systems

The mandatory architectural components required in a WAP compliant system are highlighted (grayed) in the diagram (Figure 1). These include:

1. WML (Wireless Mark-up Language) Browser
2. WMLScript Interpreter
3. TeleVAS features
4. The Session Protocol Layer (WSP)
5. A Security Protocol Layer
6. The Transport Protocol Layer (WTP)
7. Content Formats

The scope of the WAP compliancy covers the layers from transport layer to application layer of the WAP stack.

The availability of certain wireless systems, or the availability of certain network data bearer service is out of the scope of WAP compliancy. This means that a system supporting only Unstructured Supplementary Services (USSD) of GSM is WAP compliant if it otherwise supports all the protocols required.

Also, the MMI (Man-Machine Interface) of the WAP compliant system is out of the scope of compliancy. Some low-end systems might provide access to all information in the handset through a browser, while others have the browser as one of several applications available to the end user.

4. Components

4.1 Application Layer

The Wireless Application Environment (WAE) architecture specifies a general-purpose application framework for wireless devices, e.g., phones and PDA's. The Wireless Application Environment (WAE) specifies an application framework which extends and leverages the other WAP technologies, including WTP and WSP.

4.1.1 Goals

The following list summarizes the goals of the Wireless Applications Environment (WAE):

- To define a general-purpose application programming model:
  - for delivering interactive applications on wireless devices such as cellular phones.
  - which follows the Internet World-Wide-Web programming model, and which includes both browsing and scripting services.
- To define a general purpose Telephony Value-Added Service (TeleVAS) framework within the context of the WAE application execution model., which enables Network Operators the means to enhance and extend advanced Network Services to the end user.
- To define a TeleVAS Services interface which includes application access control, TeleVAS Services API with access to mobile device functions like Phonebook, Messaging and Call Control.
- To define an application model, which:
  - is suitable for building interactive applications which function well with narrow band bearers (with 300 bits/s or more), medium-to-high latency networks (1s to 15 second round trip).
  - is suitable for building interactive applications in limited function devices. This includes limited memory, small screen size, limited battery life and restricted input mechanisms.
  - has good access control, and is suitable for devices which execute anonymous applications.
  - leverages existing and ad-hoc standards, with emphasis on those technologies which will make it simpler and cheaper for third-party developers to create and deploy applications.
  - is global in nature, both in the reliance on underlying technologies, and in the assumptions about application user interface (e.g. native language and locale).
- To clearly define the optional and mandatory components of the application model, and to provide a architecture supporting multi-vendor interoperability.

4.1.2 Requirements

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

- Network services will be based on Wireless Transport Protocol (WTP) and Wireless Session Protocol (WSP).
- General Application Framework – the browser is not the only application. Other applications will exist in the device, and should integrate well with the browser. In addition, those other applications should be able to use the WAE functionality.
- The foremost requirement to WAE is to provide a simple yet powerful application development environment.
- Device Capabilities – the application model needs a capability mechanism, providing applications with a means to determine device characteristics.
- Memory consumption – the application model must have a small memory footprint, suitable for implementing on the current generation of wireless devices.
- CPU consumption – the application model must function well with a limited amount of computational power.
- TeleVAS – the application model must include a means for call control and messaging, as well as enabling a standard set of value added call and feature control capabilities.
- International Support – the application model must be capable of supporting global applications, and must support Unicode character encodings.
- Access Control – the application model must be secure when executing anonymous applications or content.
- Statement of Compliance – the WAP specification must clearly delineate required and optional components of the application model.
- Vendor-controlled MMI - End users shall be presented with a consistent and vendor controlled application MMI.

4.1.3 Component Architecture Overview

The WAP application architecture is defined primarily in terms of networking protocols and content formats and shared services. Vendor specific programming interfaces and API are not standardized, and are specific to an individual implementation. This approach leads to a very flexible architecture, which can be implemented in a variety of ways, but which also provides good interoperability and portability at the network interfaces.

This approach works particularly well with a browser 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 WAP specification, and consequently, a similar approach to specification is used within WAP.

Standards and specifications are provided for the transport protocols, content exchange formats, and for the semantics of content. The initial version of the WAP application architecture will not specify the WAP programming interfaces or API, but it is possible that this will change in the future.

4.1.3.1 Introduction to World-Wide-Web Programming Model

The Internet World-Wide-Web (WWW) provides a very flexible and powerful programming model. Applications and content are presented in a set of standard data formats, and are *browsed* by applications known as Web Browsers. The Web Browser is a networked application – it sends requests for named data objects to a network server, and the network server responds with the data encoded in one of the standard formats.

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

- Naming – all servers and content on the WWW are named with a Internet-standard *Uniform Resource Locator* (URL).
- Typed data – all data on the WWW is given a specific type, allowing the Web Browser to correctly display it.
- Standard content formats – there are a variety of standard content formats supported by all browsers. These include the HyperText Markup Language (HTML) display language, the JavaScript scripting language, and a large number of other formats.
- Standard Protocols – standard networking protocols allowing any web browser to communicate with any web server. The most commonly used protocol on the WWW is the HyperText Transport Protocol (HTTP).

This standard infrastructure allows users to easily reach a large number of third-party applications, and allows application developers to easily program to a large community of clients (e.g. Netscape Navigator ™, Microsoft Internet Explorer ™ .

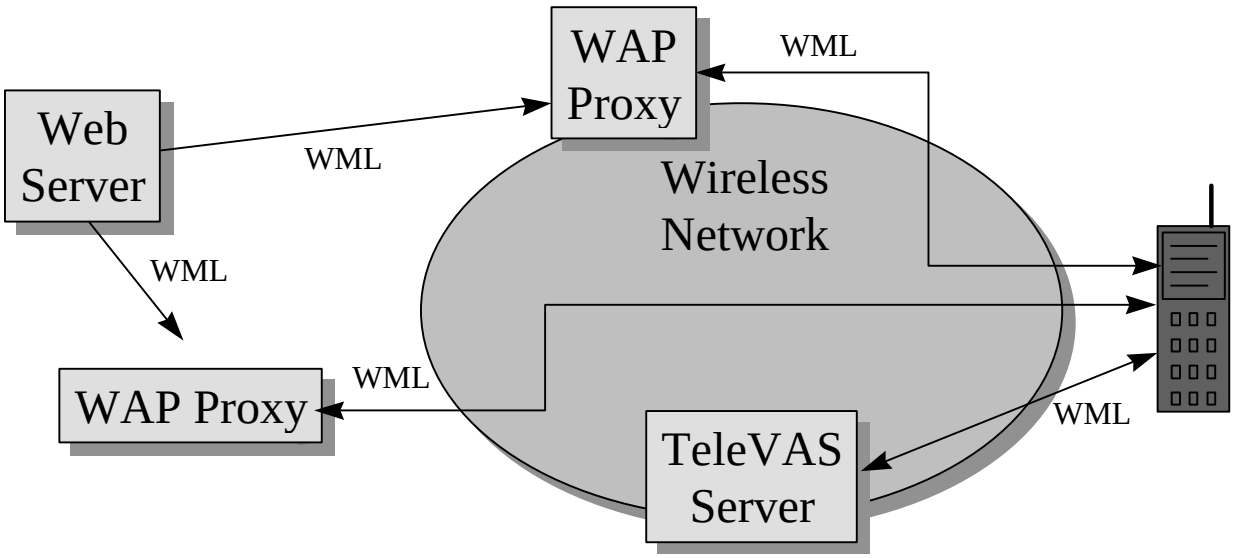

Figure 2: Standard Infrastructure

4.1.3.2 WAE Application Programming Model

The programming model of the Wireless Application Environment (WAE) closely follows the WWW programming model. All content is specified in formats which are similar to the standard Internet formats, and is transported using standard protocols on the WWW, while using an optimized HTTP-like protocol in the wireless domain. WAE has borrowed from WWW standards and programming semantics wherever possible. Where existing standards were not appropriate due to the unique requirements of small wireless devices, WAE has modified the standards , without loosing the benefits of Internet technology.

The WAE architecture is built on the concept of content interpreters and a shared service layer containing common features like location independent addressing (URLs and URL registry), event handling and TeleVAS services. The WAP Execution environment provides the device with a general-purpose WML browser, a WML scripting engine.

The WAE architecture allows all content and applications to be hosted on standard Web servers, and developed using proven technologies such as CGI and Java. All content is located using WWW-standard URLs (Uniform Resource Locators).

WAE enhances the WWW standards in ways that reflect the device and network characteristics. 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. And, support for low bandwidth, high latency networks is included.

4.1.3.3 Components of the WAE architecture

The application architecture includes a variety of components.

- Content Interpreters and Applications –in-device software that provides specific functionality to the end-user, and which is integrated into the WAP architecture. It is expected that all applications and content-interpreters will be accessible via URLs, or will interpret network content that is named by a URL. WAE will include interpreters for the two standard contents: the WML and WML-Script.
- Network-based content generators – network servers (Web servers) that generate standard content formats in response to requests from the mobile terminal. WAE does not specify any standard content generators, but expects that there will be a great variety available.
- Telephony Value-Added Services (TeleVAS) – a collection of call control and feature control mechanisms. The intent is to make advanced Mobile Network Services available to end users.
- Common Application Services – a variety of in-device services that are available to applications and browsable content. These include such services as a TeleVAS, URL registry, event distribution and device capability provision.
- Standard Content Encodings – a set of well-defined content encodings, allowing a WAP application (e.g. the browser) to conveniently navigate web content. These include compressed encodings for WML and bytecode encodings WML-Script, standard image formats and a multi-part container format.

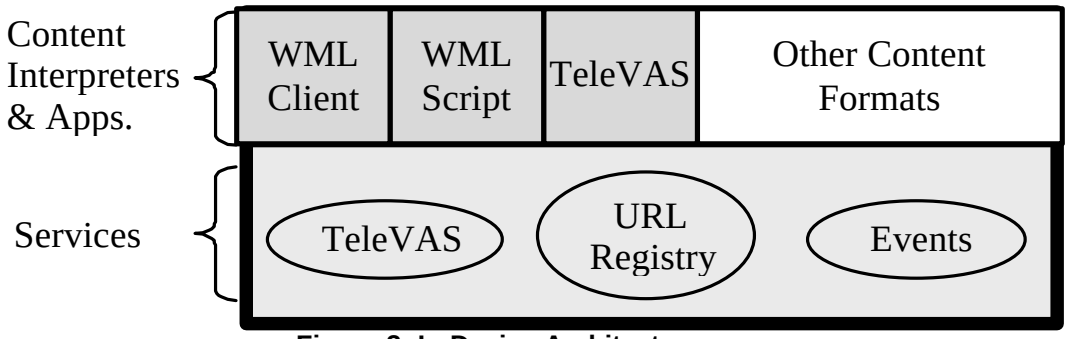

Figure 3: In-Device Architecture

4.1.4 Benefits

- Leverage the Internet – take advantage of standards, technology and infrastructure developed for the Internet.
- Thin-client Architecture – application deployment has significantly lower cost per device. This is due to the device independent nature of WAE, and the centralized management of the application.
- Leverage availability for end user to advanced Mobile Network Services through Network Operator controlled TeleVAS applications.
- Provides the means for vendors to build user friendly devices that can take advantage of WWW and Mobile Network Services.
- Provides an open, extensible and future proof framework for wireless applications.

4.1.5 Features and Characteristics

4.1.5.1 WML (Wireless Markup Language)

WML is a tag-based document language. WML shares a heritage with the World-Wide-Web's Hypertext Markup Language (HTML), and like HTML, WML is specified as an SGML document type. WML is optimized for specifying presentation and user interface on small screen, narrow-band devices such as phones and other wireless mobile terminals. WML can be stored in 'static' files on a Web server, or can be dynamically generated by an application.

WML contains constructs allowing the application to specify documents made up of multiple *cards*. An interaction with the user is described as a series of cards, which can be grouped together into a *deck*. 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. Decks are fetched from the server as needed.

Each card contains a specification for a particular user interaction. As with the HTML document language, WML is specified in a way that allows for presentation on a wide variety of devices yet allowing for vendor-controlled MMIs. WML also specifies requests for user input in a very abstract manner, which is feasible for a wide variety of input devices and mechanisms.

WML has a wide variety of features, including:

- Text and Images – WML provides the application with a means to specify text and images to present to the user. This may include layout and presentation hints. As with other markup languages, WML requires that the application programmer specify the presentation in very general terms, and gives the browser a great deal of freedom to determine exactly how the information is displayed.
- User Input – WML supports application requests for user input. This may take the form of a menu, a free-form text or number entry field, or a multi-field form. All requests for user input are also made in abstract terms, allowing the browser the freedom to optimize for the particular device.
- Navigation – WML allows several for several navigation mechanisms, all specified using URLs. Navigation includes HTML-style hyperlinks, as well as inter-card navigation tags.
- International Support – the WML document character set is Unicode, enabling the presentation of most languages and dialects.
- Vendor controlled MMI – WMLs abstract specification of layout and presentation enables a vendor-controlled MMI design.
- Narrow-band Optimization – WML includes a variety of technologies to optimize presentation on a narrow-band device. This includes the ability to specify multiple user interface *screens* (cards) in one network transfer (a deck). It also includes a variety of state management capabilities (e.g. variables) which remove the need for network server requests.

The reference WML architecture requires both a WML browser and a Web (HTTP) server. When the WML browser requires an additional deck (identified by a URL), it fetches that resource via a WSP request. The response contains WML, which is in turn presented to the user.

A WML browser reference processing model is therefore:

1. Display current card.
2. Wait for user input, indicating the new URL to navigate to.
3. Send a WSP request to the appropriate Web server, requesting the resource named by the URL.
4. Wait for the response
5. Go to step #1.

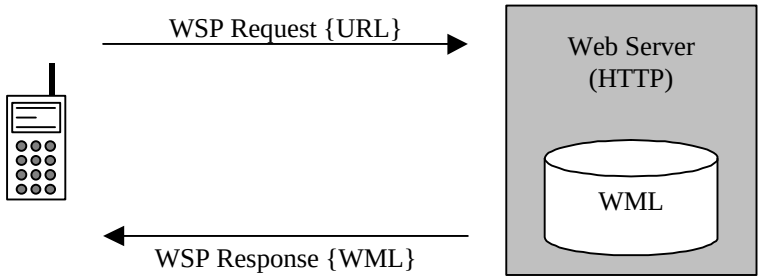

Figure 4

4.1.5.2 WML-Script Interpreter

WML-Script is a lightweight procedural scripting language. It enhances the standard browsing and presentation facilities of WML with behavioral capabilities, supports more advanced UI behavior, provides a convenient mechanism to access the device and its peripherals, and reduces the need for round-trips to the network server.

WML-Script is based on a subset of the JavaScript™ WWW scripting language. JavaScript is widely deployed in all major HTML browsers, and forms a standard means for adding procedural logic to HTML web pages. WML-Script refines JavaScript for the narrowband device, integrates it with the WML browser and provides hooks for integrating in-device applications (e.g. for accessing TeleVAS services).

WML-Script provides the application programmer with a wide variety of interesting capabilities:

- The ability to check the validity of user input before it is sent to the network server.
- The ability to conveniently access device facilities and peripherals.
- The ability to interact with the user without a round-trip to the network server (e.g. display an error message).

4.1.5.2.1 WML-Script Features

WML-Script has the following features:

- JavaScript-based scripting language – WML-Script starts with an industry standard solution, and adapts it to the narrowband environment. This will make WML-Script very easy for a Web developer to learn and use.
- Procedural Logic – WML-Script adds the power of procedural logic to the WAP application environment.
- Event-based – WML-Script may be invoked in response to certain user or environmental events.
- Interacts with WML – WML-Script is fully integrated with the WML browser. This allows an application programmer the ability to construct their application using both technologies, using the solution that is most appropriate for the task at hand.
- Efficient extensible library support – WML-Script can be used to expose and extend device functionality without changes to the device software.

4.1.5.2.2 WML-Script Integration

WML-Script is fully integrated into the WAP application framework. WML-Script has access to the WML state model, and can set and get WML variables. This enables a wide variety of functionality (e.g. validation of user input collected by a WML card).

4.1.5.3 TeleVAS

This chapter outlines the **Tele**phony **V**alue **A**dded **S**ervices (**TeleVAS).** It also includes a discussion of the context for executing TeleVAS functions from the WML browser environment and also briefly discusses how the TeleVAS functions tie into the WML-Script constructs.

TeleVAS is fully integrated with the WAP application framework and provides an efficient and secure way to access local functions like Call Control, Phonebook, Messaging etc. The TeleVAS functions specify a device independent interface to the underlying vendor specific operating system and telephony subsystem. TeleVAS functionality does not rely on network specific functionality, and will be equally applicable to GSM, CDMA or any PCS type of network.

The WAP TeleVAS services make it possible to create applications that enhance and extend services available in today's advanced Mobile Networks. Services in the network can be made more accessible to the end user through user friendly menus and TeleVAS applications that hide many of the complicated call control features.

Existing third party IVR solutions built on touch-tones can also benefit using a TeleVAS application wrapper that presents the user with a scrollable menu sending touch tones without the user having to manually enter each keypress. Thus, operators seamlessly integrate WWW and telephony applications.

The TeleVAS application is built using standard WML cards and WML scripts/libraries downloaded through the WAP URL services. TeleVAS applications make it possible for the operator to tailor existing network services and make new features available to the end user. The available network transports can be used more efficiently with smart applications using script and cards that persist in the local device memory for quick access. The WAP content/application download makes it possible to keep the users handset updated with customized TeleVAS applications as soon as the network services change.

TeleVAS provides controlled Network access and enforces user control and privacy to Mobile Services. The URL registry provides seamless access to remote/local TeleVAS functions.

TeleVAS will provide access control to local functions, and is expected to restrict access to a *well-known* carrier-operated TeleVAS server.

TeleVAS functions can be invoked from any type of WML card/script. It will be the decision of the application currently running as to whether the events and return codes are ignored or passed to the user. The WAP application will have the choice of ignoring events during critical parts of the execution cycle, such as during transactions. For example the application could refuse incoming phone calls during a banking transaction.

As the TeleVAS functions are invoked, for example by an incoming voice call, the application can decide to accept the call using a local URL. In case of error a specific exception event will be generated. The handling of this "error/network" event is specified using service layer primitives. It will be at the discretion of the manufacturers as to the implementation of TeleVAS Functions, whether they are in assembly, C, Java or other scripting languages.

4.1.5.3.1 Local And Remote TeleVAS URLs

A significant feature of TeleVAS is the location independence of the implementation. In other words, a handset or a server can be used to implement a specific feature, and the location does not change the programming interface or user interface. All TeleVAS URLs use a local URL alias, e.g., **device://file/function**.

TeleVAS content providers may wish to implement some functions via a remote URL. For example, an extensive directory lookup service might be easier to implement if a remote TeleVAS URL on a server was invoked. These services can be accessed through the URL **http://televas.domain/file/function.** Typically the remote call will take the form of an http address**.** For example, the case of an application such as voice mail, a combination of **device** and **remotedevice** URLs would be specified.

TeleVAS supports functions defined within the MS, as well as application dependent functions supported remotely by the Network Operator. The Local TeleVAS section of this document outlines some of the functions that *must* be supported by a handset.

URLs that calls remote functions will be defined in WML. Implementation of remote TeleVAS functions will be specific for the Network or TeleVAS Content Provider. For local TeleVAS functions, a default time-out parameter assures that network failures can be handled properly by the application in the MS.

The function calls, local and remote, is based on the Internet CGI concept (Common Gateway Interface). Using a CGI with an URL indicates to the Web server that the requested file is in fact an application that should be executed, prior to returning the new content/variables. If not specified the Web server instead returns the indicated WML content to the browser.

4.1.5.3.2 TeleVAS Functions

Call control is an example of local TeleVAS functions that are proposed for the first WAP version. The local functions also include associated events to provide feedback from the outcome of a function call.

Call Control

The TeleVAS Call Control Services include functions to set up call, call management and touch tone commands. For example:

- Accept Incoming Call
- Set Up Mobile Originated Call
- Disconnect Call
- Multi Party Call
- Call Transfer
- Send DTMF
- Set DTMF Mode
- Call Forwarding

4.1.5.3.3 TeleVAS Events

TeleVAS will define a set of events relating to the call and feature control. For example:

- Incoming Call Indication – An incoming call is detected.
- Call Waiting Indication – A new incoming call is detected while a call already is in progress.
- Disconnect Indication – signals that a call has been disconnected.

4.1.5.4 Common Application Services

Common Services are architectural elements that manifest themselves through WML or WML-Script. These services provide a general framework for handling user input, integrating applications, etc. Common services may include:

- Motivation: shared basic services
- Events: Sync & Async. - Provides common semantics for handling of user input as well as application specific events, generated by the server.
- URL-based registry naming model and registry – provides a standard naming and rendezvous model for applications and content.
- TeleVAS functionality – exposes an interface to advanced Mobile Services.
- Others added necessary

WAE intends to expand this area as needed to achieve a consistent and useful application framework.

4.1.5.5 Device Capabilities

Device capabilities are a mechanism that allows the application to determine characteristics of the mobile terminal device. It is expected that the session layer protocols (WSP) will include a mechanism for exchanging and caching capabilities. WAE will define a set of application-level capabilities that will be exchanged using the WSP mechanism. These capabilities will include such global device characteristics as:

- WML and WML-Script version supported by the device.
- Support for different image formats.
- Etc.

4.1.5.6 Other Content Formats

4.1.5.6.1 Images

WAE will define a common image format suitable for transmission to a mobile terminal. The primary requirements for the image format will include:

- Support for multiple pixel depths.
- Support for colorspace tables.
- Very small encoding

Very low CPU and RAM requirements for decoding and presentation

4.1.5.6.2 Multipart messages

WAE will include a well-defined multipart encoding specification, suitable for exchanging multiple typed entities over WSP. This will be based on the Internet MIME specification, but will be tuned and optimized for the narrowband environment

4.1.5.6.3 Compiled WML encoding and WMLScript bytecode

WAE will define binary (compiled) encodings for WML and WML-Script. These encodings will make transmission of WML and WML-Script more efficient.

4.1.5.6.4 Additional Content Formats

WAE will define additional content formats for the purposes of exchange of data between applications and devices. These content formats will focus on data and applications commonly found in intelligent devices.

Examples of content formats could be:

- Business Cards (phone book data)
- Calendar items
- Mail headers

4.1.6 Component Roadmap

The following areas are currently undergoing work in the WAE:

- Extensions to WAP TeleVAS Services with support for state of the art Telephony Devices with advanced local functions for GSM SIM card access, enhanced messaging and user customizable TeleVAS menus.
- Secure third party access to selected local functions using the WAP Access Control features
- Generic access to customized local Applications. (Calendar, Business cards etc).
- Network-Specific API – e.g., GSM SIM apidevice.
- Additional Content Encodings – WAE considers specification or adoption of standard content encodings in a variety of other areas (e.g. sound, etc.).

4.2 Session Layer

The WAP Session Layer provides efficient and compact mechanisms for exchanging typed data between WAP applications in a secure fashion.

4.2.1 Goals

The WAP Session Layer builds upon the bearer-independent, scaleable WAP Transport Layer to provide communication services useful to many applications. One of these services is a general mechanism for securely exchanging typed data between the client and server, including support for server-initiated transfers (‘push’). In WAP, the session concept covers both a security association and the idea of a relatively long-lived application association between a client and a server. The application session allows the optimization of communication by exchanging a certain set of static information during session creation eliminating the need to explicitly exchange it on each subsequent communication.

4.2.2 Requirements

The following requirements apply to the WAP Session Layer architecture:

- Lightweight session establishment and termination
- Multiple simultaneous sessions
- Typed data transfer
- An abstract service interface
- Provide security facilities for encryption, strong authentication, integrity, and key management
- Compliance with regulations on the use of cryptographic algorithms and key lengths in different countries
- Transport service will be based on WTP and compatible with TCP/IP, UDP transport service
- Simple mapping to the Internet architecture and Internet protocols
- Support WAP sessions in parallel with voice or data connections

- A layered architecture which can be extended with additional protocols when needed
- Common facility for push
- Support for capability negotiation
- Support for content negotiation
- Extensible to support QoS-based service models
- Efficient bandwidth utilization
- Small memory footprint
- Realistic processing power requirements

4.2.3 Component Architecture Overview

The requirements of potential WAP applications are different, so it may not be feasible to define a single session layer protocol that fits all needs. As a result, the WAP session layer defines a common security protocol layer on which more specialized session protocols can be layered. A WAP session layer protocol will provide an access point to upper level protocols. The security protocol layer may optionally provide its own access point, or it may be accessed through the session layer access point.

The session layer protocols use services provided by the transport layer abstract service interface. This provides for the possibility of using the session layer protocols on a transport different from the current WAP transport layer, as long as it provides similar characteristics. The session layer protocols do not duplicate functionality provided by the transport layer.

The considered session layer protocol candidates strive for the simplest possible client implementation by using an asymmetric client-server model in which complicated functions are handled by the server whenever possible. However, this does not prevent mobile stations from communicating with each other, as long as one of the mobile stations has sufficient resources to act as a server.

The architecture of the session layer is outlined in figure 5.

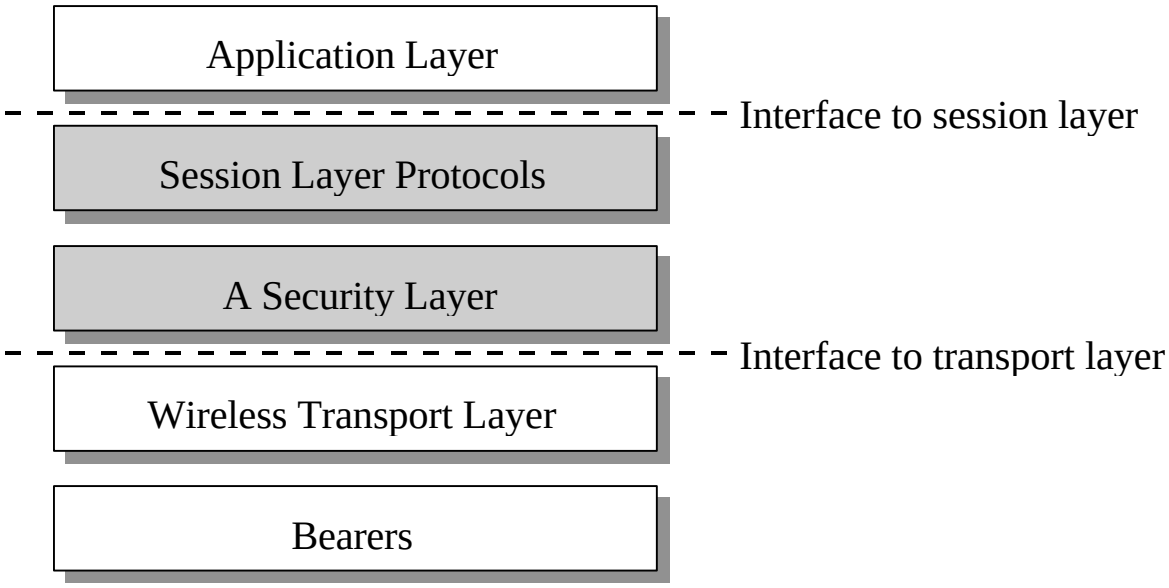

Figure 5: The Session Layer Architecture

4.2.4 Benefits

The major benefits are:

- Provides a simple but versatile communication model for applications
- Session concept allows applications to optimize communications by establishing shared state between client and server
- Leveraging standards allows easy interoperability and the reuse of Internet software infrastructure
- Capability negotiation allows applications to adjust to a wide range of devices
- Common push model for applications
- Single general security solution makes implementation of secure applications simpler, smaller and faster
- Asymmetric client/server architecture enable very thin clients/handsets/devices
- Specified key exchange protocol and algorithm negotiations allow better interoperability
- Compatibility with TCP/IP, UDP transport provide implementation flexibility.

4.2.5 Features and Characteristics

- Session management, including capability negotiation
  The session layer protocols will support the session models required by the application layer. This includes session establishment and termination using mechanisms provided by the transport layer protocols. Also, device capability negotiation will be provided, so that applications may adapt their usage of device resources like display size, input mechanisms etc. This will provide applications with a possibility to scale across a range of devices, without the need to have separate implementations for each device.
- Multiple simultaneous sessions
  The session layer protocols will provide support for multiple simultaneous sessions. This will enable devices to run several applications at a time, and will support applications requiring multiple connections.
- Sessions concurrent with voice or data connections if supported by bearer network
  A WAP session may be established at any time the network is available. This includes the possibility to establish a WAP session while a voice or data call is active, or vice versa. The WAP session(s) and voice or data call will be handled as completely independent of each other.
- Provides both push and pull data transfer
  Different models of data transfer will be supported by the session layer protocols. This will ensure that service providers can build flexible information services, adapted to market requirements. The session layer protocols will map all data transfer models to the basic request-response session model, using additional protocol elements where applicable.
- Strong authentication and encryption, which may be adapted to suit different legislation and requirements in different markets
  The security part of the session layer will incorporate different encryption and authentication mechanisms in order to cope with different legislation and requirements in different markets. Thus, products with different levels of security functionality will be able to negotiate a set of common capabilities.
- Specified key management enables establishment of authenticated and secure communications between two peers without *a priori* knowledge
  Mechanisms for establishing authenticated and secure communications shall be comparable and provide the same level of security as current Internet protocols. In the WAP scenario, emphasis has been put on finding algorithms that are as little processor- and memory intensive as possible, while still retaining strong security and ease of use.
- Typed data transfer, including composite objects
  The session layer protocols provide typed data transfer for the application layer, comparable to that of HTTP 1.1.
- Extensible system for data typing
  The session layer protocol will support extensible data typing for future proof solutions. Emerging data types and file formats will be supported in order to keep a strong coupling to the Internet community, and to allow for new, WAP-specific applications and data types.

- Content type negotiation
  The session layer protocol will support content type negotiation, allowing a server application to decide whether a client can support a certain type of content. This will ensure that applications and users will not have to handle content that is not suitable for the actual device (e.g. a voice mail message on a device that does not support playback from a data file). The mechanisms for content type negotiation will be comparable to that of HTTP 1.1.
- Supports both datagram and connection-oriented applications
  In order to support maximum flexibility and extensibility, both datagram and connection-oriented transport protocols and applications are supported by the session layer protocols.
- Modular, layered, flexible architecture
  The entire WAP protocol stack is designed for leveraging current and future Internet protocols. This requires the architecture to be modular and layered in order to cope with potential new protocols to be included in the WAP framework, and in order to provide access points for applications not using the entire WAP protocol stack
- Scaleable in terms of features and resource requirements
  A key feature of the WAP protocol stack is that applications should be able to run on a wide range of devices, while utilizing each separate device to the maximum of its' capabilities. This leaves room for optimized implementations of the protocol stack, while retaining interoperability with a wide range of applications and devices.
- Efficient use of air-time
  In a wireless environment, air-time is a precious resource. The session layer protocols are therefore optimized for efficient use of air-time, both from a cost perspective and a service perspective.
- Small memory footprint
  The session layer protocols are designed to have a small memory footprint in an actual implementation, in order to be implementable in a large range of device classes.

4.2.6 Component Roadmap

Initially the session layer specifications will cover a layered security solution and at least a session layer protocol addressing the needs of interactive browsing applications. This protocol will be an HTTP 1.1 derivative, augmented with push functionality and a compact binary header encoding.

Future work items under consideration are a protocol for large object transfer, a protocol for one-way information delivery, suitable also for multicast and broadcast transmission. More optimal methods for mobile-to-mobile communication may also be considered in the future.

4.3 Transport Layer

The Transport layer protocol family in the WAP architecture is Wireless Transport Protocol, WTP. The WTP layer operates above the data capable bearer services supported by multiple network types. In the initial phase, bearers from the GSM network will be used. WTP will offer a consistent service to the upper layer protocol (Session and Security) of WAP and communicate transparently over one of the available bearer services.

The Wireless Transport Protocols consists of a connection oriented protocol (WTP/C) and a datagram oriented protocol (WTP/D). The Protocols in the WTP family are optimized for very slow bearers in telecommunications.

WTP/D is a simple transport protocol. The WTP/D protocol is relayed transparently on the underlying bearers, i.e. the datagram information is moved unchanged from client to server and as well as from server to client. Any application layer protocol is relayed transparently on the WTP/D transport mechanism.

WTP/C is a connection oriented transport protocol. The WTP/C is optimized for low bandwidth wireless bearers. WTP/C is more efficient on request-reply applications than traditional connection oriented protocols.

4.3.1 Goals

The following list summarizes the goals for the Wireless Transport Protocol Group:

- to act as a common interface to physical transport mechanisms across multiple wireless network types and multiple bandwidth and latency options within a single network type. Network type includes full duplex, half duplex and simplex technologies.
- to provide a clear, port-based abstract interface to upper layer protocols in a manner that enables the session layer to implement scaleability for applications across transport and device types.
- to eliminate the need for applications to be designed for specific available underlying bearers.
- to be extensible to a variety of digital wireless networks and future transport options.
- to support both connection-oriented and connectionless modes.
- to optimize for narrow to medium bandwidth channels.
- to clearly specify the mandatory and optional features of the protocol to ensure multi-vendor interoperability.
- to allow peer-to-peer and client/server applications to operate over different transport within a single network type.
- to be capable of implementation in a low memory footprint, suitable for “standard” or “low-IQ” handsets.

4.3.2 Requirements

The following is a list of requirements for the Wireless Transport Protocol Group:

4.3.2.1 WTP Common Requirements

- Light Weight: The protocol must be suitable for narrow-band channels and should be feasible to implement it with low memory foot-prints and low computational needs.
- Wireless Network Support: The following networks must be supported: Phase 1 SMS, Phase 2 SMS, Phase 1 USSD, Phase 2 USSD, cell broadcast, and Phase 1 GPRS; circuit switched data. For the near term, Phase 2 SMS and Phase 2 USSD must be supported.
- Protocol Specification: Abstract Service Primitives will be defined. Application Programming Interfaces (API’s) will not be specified.
- Selection of underlying bearer: Abstract Service Primitives must be provided which support the selection of an underlying bearer.
- Intimate Knowledge of Available Transports: The need for applications to be aware of the specifics of the available transports must be eliminated.
- Port Numbers: Port numbers must be supported and implemented at the protocol level.
- Efficiency:
  - Concatenation (Segmentation and Reassembly) must be supported where applicable. This could be a datagram implementation.
  - Header Compression should be supported.

4.3.2.2 WTP/C specific Requirements

- Reliability: WTP/C must provide reliability at the protocol level so that the applications that use WTP/C do not have to program reliability into them. Reliability must be provided across the connection.
- Negotiate Window Size: Sender must control the size of the window, but the receiver must be able to request changes in the window size on a per connection basis.
- Efficiency:

- Connection Setup and Teardown: The protocol must be efficient in setting up and tearing down connections, including carrying data at the same time.

4.3.2.3 Long-term Requirements

- Support for additional wireless networks: Additional support for non-GSM (e.g. CDMA, US-TDMA and CDPD) must be specified.

4.3.3 Component Architecture Overview

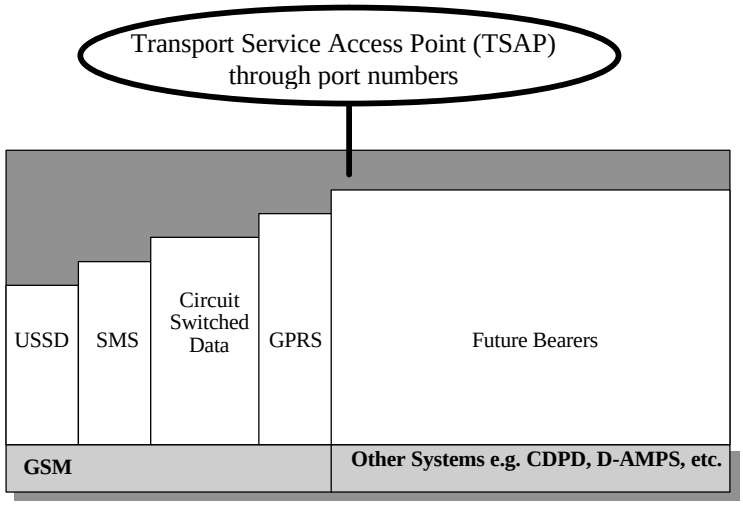

Figure 6: Wireless Transport Protocol Architecture

The varying heights of each of the bearer services shown in Figure 6 is used to illustrate the difference in functions provided by the bearers and thus the difference in WTP protocol necessary to operate over those bearers.

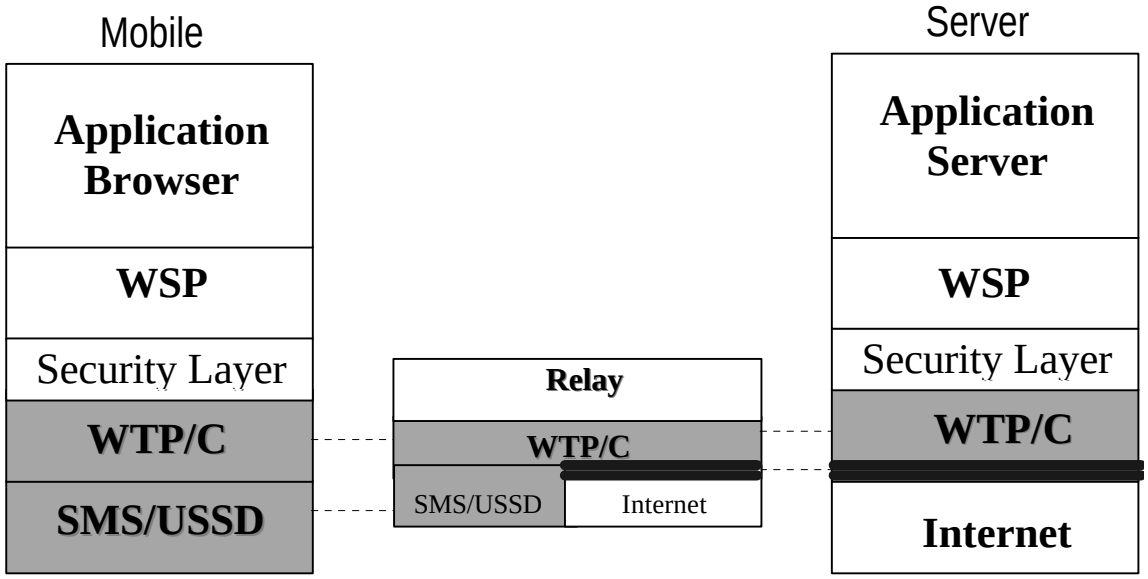

Figure 7: The WTP/C is an end to end reliable, connection oriented, transport protocol between communicating devices.

In order to provide an end to end reliable transport service the WTP/C is using different underlying bearers in the wireline and the wireless segment of the transport. The relay acts as a bearer conversion gateway, using the available network as efficiently as possible.

In Figure 7, the WTP/C uses SMS or USSD as an underlying bearer in the wireless segment to reach the relay. From the relay, the WTP/C could use any wireline transport protocol (ex. some Internet bearer protocol) to reach the server.

Communications between layers are accomplished by means of service primitives. Service primitives represent, in an abstract way, the logical exchange of information and control between the transport layer and adjacent layers.

4.3.4 Benefits

- Provides a simple yet reliable communication model for applications and other layers to use
- QoS reporting permits the applications to optimize for the underlying network bearer
- Eliminates the need for upper layers to be designed for specific available underlying bearers
- Supports segmentation and reassembly
- The definition of the protocol allows multi-vendor interoperability between mobile phones and servers yet allowing mobile hone manufacturers to continue to run their own proprietary environments inside the phones.

4.3.5 Features and Characteristics

4.3.5.1 Features

- Supports both datagram and connection-oriented transports
- Optimized for narrow to medium bandwidth channels
- Allows negotiation of window size on a per connection basis

4.3.5.2 Characteristics

- Modular, layered, flexible architecture
- Scaleable in terms of network types
- Efficient use of underlying bearers
- Small memory footprint
- Support for port numbers implemented at protocol level

4.3.6 Component Roadmap

Initially the transport layer specifications will cover datagram oriented and connection oriented transport protocols for GSM network types: Phase 2 SMS and Phase 2 USSD.

Future work items under consideration are protocol definitions for Phase 1 GPRS and Circuit Switched Data. Other non-GSM network types will also be considered in the future. In the near term, its a goal to allow peer-to-peer and client/server applications to operate over different transport within a single network type. In the future this will be expanded to cover one-to-many applications to operate over different transport within a single network type. It should be noted that *one-to-many means broadcast and unacknowledged multicast only.*

Application Programming Interfaces (APIs) will be developed in the future modeled after existing transport interfaces.

5. Technical Roadmap

The intent for the WAP workgroups is to develop the WAP standard for the following wireless network standards; GSM 900, GSM 1800, GSM 1900, PDC, CDMA, US-TDMA, IS-95, USDC(IS-136), iDEN(ESMR), DataTAC and Mobitex. The initial environment for the WAP workgroups is the GSM network with SMS (Short Messaging Service), USSD(Unstructured Supplementary Services Data), CSD(Circuit Switched Data) and GPRS(Global Packet Radio System) bearers. A consistent and parallel step is to adapt WAP to the other network standards previously identified. The standardization process of the protocols will also be opened to a larger community as processes and rules have been agreed. Eventually each protocol or layer is to be handed over to appropriate standards bodies. As the WAP work spans over a large field the maintenance of the standards will be divided between multiplestandards bodies.

Appendix A: Contact Information

Contacts		
WAP		Internet: www.xwap.com
Ericsson	Tel: +46 8 757 2159	Internet: www.ericsson.se
Motorola	Tel: +44 1256 790122	Internet: www.motorola.com
Nokia	Tel: +358 10 5051	Internet: www.nokia.com
Unwired Planet	Tel: +1 415 596 5251	Internet: www.uplanet.com

Appendix B: Version History

Document history		
Date	**Status**	**Comment**
15-Sep-97	Version 0.9	Final draft for publication at www.xwap.com
This document is written in Microsoft Word for Windows 95 and saved as MS Word Version 6.0 The document was then converted using Adobe Acrobat 3.01 to Portable Document Format (PDF)		