## UNIT I – MOBILE DATA INTRODUCTION

### Introduction to Mobile Data

Mobile data is Internet content delivered to mobile devices such as smartphones and tablets over a wireless cellular connection.

Cellular providers have offered mobile data through a number of different technologies including Global System for Mobile Communications (GSM) (in 1G, 2G, 3G UMTS and 4G LTE Advanced), Code Division Multiple Access (CDMA) and Time Division Multiple Access (TDMA).

UMTS (Universal Mobile Telecommunications Service) is a third-generation (3G) broadband, packet-based transmission of text, digitized voice, video, and multimedia at data rates up to 2 megabits per second (Mbps). Once UMTS is fully available, computer and phone users can be constantly attached to the Internet wherever they travel and, as they roam, will have the same set of capabilities. UMTS is based on the Global System for Mobile (GSM) communication standard.

**LTE** is an abbreviation for Long Term Evolution. **LTE** is a **4G** wireless communications standard developed by the 3rd Generation Partnership Project (3GPP) that's designed to provide up to 10x the speeds of 3G networks for mobile devices such as smartphones, tablets, netbooks, notebooks and wireless hotspots.

4G technologies are designed to provide IP-based voice, data and multimedia streaming at speeds of at least 100 Mbit per second and up to as fast as 1 GBit per second.

### Global System for Mobile Communications (GSM)

It is a digital cellular technology used for transmitting mobile voice and data services. The concept of GSM emerged from a cell-based mobile radio system at Bell Laboratories in the early 1970s.

GSM is the most widely accepted standard in telecommunications and it is implemented globally. GSM is a circuit-switched system that divides each 200 kHz channel into eight 25

kHz time-slots. GSM operates on the mobile communication bands 900 MHz and 1800 MHz frequency bands in most parts of the world.

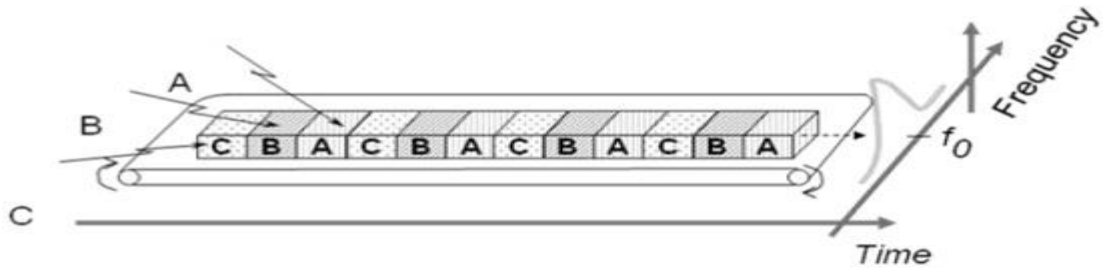
GSM makes use of narrowband Time Division Multiple Access (TDMA) technique for transmitting signals. GSM was developed using digital technology. It has an ability to carry 64 kbps to 120 Mbps of data rates. Digital cellular technology involves the combination of digital signal processing with cellular radio technology. Digital cellular radio technology was developed to allow more customers to be served by a reduced number of towers

GSM uses a variation of time division multiple access (TDMA) and is the most widely used of the three digital wireless telephony technologies: TDMA, GSM and code-division multiple access (CDMA). GSM digitizes and compresses data, then sends it down a channel with two other streams of user data, each in its own time slot.

**Time Division Multiple Access (TDMA):**

Time Division Multiple Access (TDMA) is a technology used in digital cellular telephone communication. It allows several users to share the same frequency channel by dividing the signal into different time slots in order to increase the amount of data that can be carried.

In the following example, the frequency band has been shared by three users. Each user is assigned definite **timeslots** to send and receive data. In this example, user **'B'** sends after user **'A,'** and user **'C'** sends thereafter.

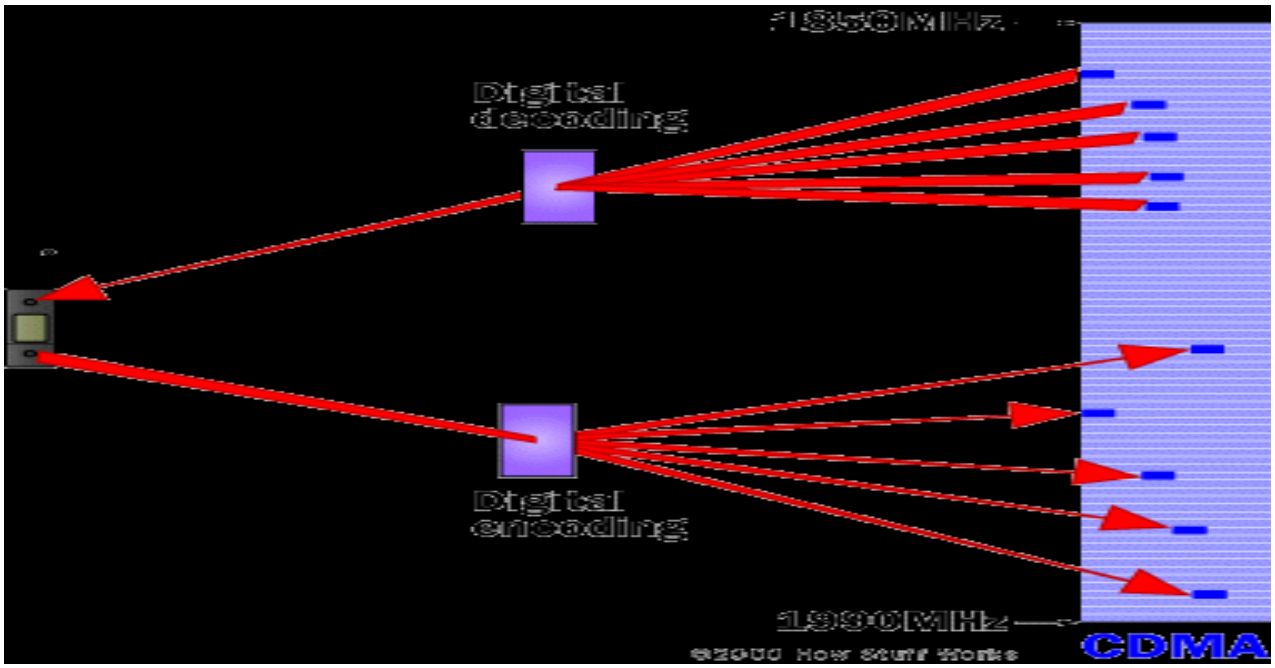



**CODE DIVISION MULTIPLE ACCESS (CDMA):**

CDMA is an example of multiple access, where several transmitters can send information simultaneously over a single communication channel. This allows several users to share a band of frequencies (see bandwidth). To permit this without undue interference between the users, CDMA employs spread spectrum technology and a special coding scheme (where each transmitter is assigned a code).

Spread spectrum is a form of wireless communications in which the frequency of the transmitted signal is deliberately varied.

A CDMA transmitter assigns a unique code to each wireless connection and then broad casts its data out on the character simultaneously with all other connections. The receiver is able to decode each conversation by knowing the unique code assigned to each connection.




**Applications where we use mobile data:**

Email, Uploading pictures, web surfing, gaming, music streaming, YouTube and various apps like facebook, whats app etc.,

**The rise of Mobile data**

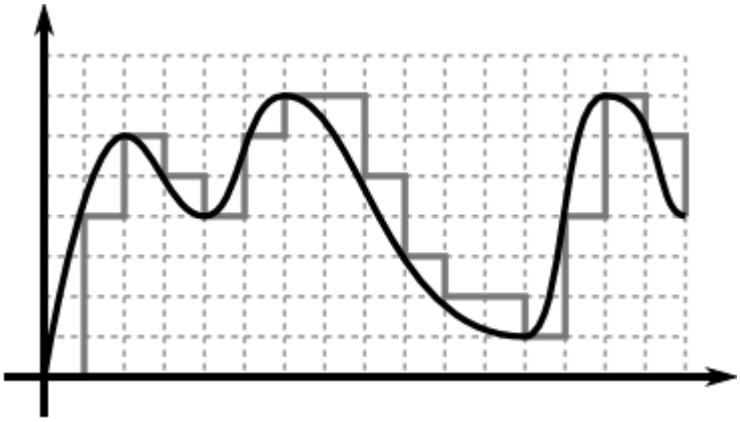
- The wireless telegraph was invented by Guglielmo Marconi in 1896. Communication was made by encoding alphanumeric characters in analog signal.
- Communication satellites launched in 1960s
- Advances in wireless technology exists like radio, television, mobile telephone, communication satellites.
- Most recently satellite communications, wireless networking, cellular technology came into existence.

Analog and Digital technologies are the technologies that are used in the Wireless Communication Traditionally cellular phones have utilized analog transmission signals. But

Analog technology has the noise pick up issue, which makes the technology inefficient. In order to diminish this noise and to provide greater calling capacity per channel, the cellular industry is beginning to use digital transmission signals. Digital technology has two forms: time division multiple access (TDMA) and code division multiple access (CDMA).

**Analog Signal**

An analog signal is a continuous signal that contains time-varying quantities. Unlike a digital signal, which has a discrete value at each sampling point, an analog signal has constant fluctuations. The illustration below shows an analog pattern (represented as the curve) alongside a digital pattern (represented as the discrete lines).



An analog signal can be used to measure changes in some physical phenomena such as light, sound, pressure, or temperature. For instance, an analog microphone can convert sound waves into an analog signal. Even in digital devices, there is typically some analog component that is used to take in information from the external world, which will then get translated into digital form (using an analog-to-digital converter).

**Digital Signal**

A digital signal refers to an electrical signal that is converted into a pattern of bits. A digital signal has a discrete value at each sampling point. The precision of the signal is determined by how many samples are recorded per unit of time.

A digital signal is easily represented by a computer because each sample can be defined with a series of bits that are either in the state 1 (on) or 0 (off).

A **digital signal** is a signal that is being used to represent data as a sequence of discrete values; at any given time it can only take on one of a finite number of values. This contrasts with

an analog signal, which represents continuous values; at any given time it represents a real number within a continuous range of values.

**Evolving Wireless Applications**

- Asset Positioning
- Tele-metering
- M-Commerce
- Internet Access
- Sensor networks

**Generations of Wireless Networks**

**1G** Vs. **2G** Vs. **3G** Vs. **4G** Vs. **5G**. For eg, **1G** offers 2.4 kbps, **2G** offers 64 Kbps and is based on GSM, **3G** offers 144 kbps-2 mbps whereas **4G** offers 100 Mbps - 1 Gbps and is based on LTE **technology.**

**1 Generation wireless standards: AMPS (Advanced Mobile Phone Service)**

This was the first generation of cell phone technology. 1G is an analog technology and the phones generally had poor battery life and voice quality was large without much security, and would sometimes experience dropped calls. These are the analog telecommunications standards that were introduced in the 1980s and continued until being replaced by 2G digital telecommunications. The maximum speed of 1G is 2.4 Kbps.

Advanced Mobile Phone Service (AMPS) is a standard system for analog signal cellular telephone service. AMPS allocate frequency ranges within the 800 and 900 Megahertz (MHz) spectrum to cellular telephone. Each service provider can use half of the 824-849 MHz range for receiving signals from cellular phones and half the 869-894 MHz range for transmitting to cellular phones.

**2 Generation: CDMA, GSM**

The main difference between the two mobile telephone systems (1G and 2G), is that theradio signals used by 1G network are analog, while 2G networks are digital. Main motive of this generation was to provide secure and reliable communication channel. It implemented the concept of CDMA and GSM. Provided small data service like sms and mms.

2G capabilities are achieved by allowing multiple users on a single channel via multiplexing. During 2G Cellular phones are used for data also along with voice. The advance in technology from 1G to 2G introduced many of the fundamental services that we still use today, such as SMS, internal roaming , conference calls, call hold and billing based on services e.g. charges based on long distance calls and real time billing. The max speed of 2G with General Packet Radio Service (GPRS) is 50 Kbps or 1 Mbps

**3 Generation: WCDMA, CDMA 2000, TDSCDMA**

Web browsing, email, video downloading, picture sharing and other **Smartphone technology** were introduced in the third generation. Introduced commercially in 2001, the goals set out for third generation mobile communication were to facilitate greater voice and data capacity, support a wider range of applications, and increase data transmission at a **lower cost**.

The 3G standard utilises a new technology called **UMTS** as its core network architecture - Universal Mobile Telecommunications System. This network combines aspects of the 2G network with some new technology and protocols to deliver a significantly faster data rate. Based on a set of standards used for mobile devices and mobile telecommunications use services and networks that comply with the International Mobile Telecommunications-2000 ( **IMT-2000** ) specifications by the International Telecommunication Union. One of requirements set by IMT-2000 was that speed should be at least 200Kbps to call it as 3G service.

3G has Multimedia services support along with **streaming** are more popular. In 3G, Universal access and portability across different device types are made possible (Telephones, PDA's, etc.). 3G increased the efficiency of frequency spectrum by improving how audio is **compressed** during a call, so more simultaneous calls can happen in the same frequency range.

A 3G phone cannot communicate through a **4G network** , but newer generations of phones are practically always designed to be backward compatible, so a 4G phone can communicate through a 3G or even **2G network** .

**4 Generation (MIMO, OFDM, LTE)**

Its purpose is to provide **high speed**, high quality and high capacity to users while improving security and lower the cost of voice and data services, multimedia and internet over IP.

Potential and current applications include amended mobile web access, **IP telephony**, gaming services, high-definition mobile TV, video conferencing, 3D television, and cloud computing.

The key technologies that have made this possible are **MIMO** (Multiple Input Multiple Output) and **OFDM** (Orthogonal Frequency Division Multiplexing). The two important 4G standards are WiMAX (has now fizzled out) and **LTE** (has seen widespread deployment). LTE (Long Term Evolution) is a series of upgrades to existing UMTS technology and will be rolled out on Telstra's existing 1800MHz frequency band. The max speed of a 4G network when the device is moving is 100 Mbps or **1 Gbps**

**5 Generation**

5G is a generation currently **under development**, that's intended to improve on 4G.**5G** promises significantly faster data rates, higher connection density, much lower latency, among other improvements. Some of the plans for 5G include **device-to-device** communication, better battery consumption, and improved overall wireless coverage. The max speed of 5G is aimed at being as fast as **35.46 Gbps**, which is over 35 times faster than 4G.

Key technologies to look out for: **Massive MIMO**, Millimeter Wave Mobile Communications etc. Massive MIMO, milimetre wave, small cells, **Li-Fi** all the new technologies from the previous decade could be used to give 10Gb/s to a user, with an unseen low latency, and allow connections for at least **100 billion devices**. Different estimations have been made for the date of commercial introduction of 5G networks. Next Generation Mobile Networks Alliance feel that 5G should be rolled out by **2020** to meet business and consumer demands.

**Key services for the Mobile internet**

**How does Mobile internet works?**

Cell phones have an in-built antenna which is used to send packets of digital information back and forth with cell-phone towers through radio waves. Mobile phones connect to a cell tower in the area, and instead of connecting to another phone it connects to the Internet and can fetch or retrieve data.

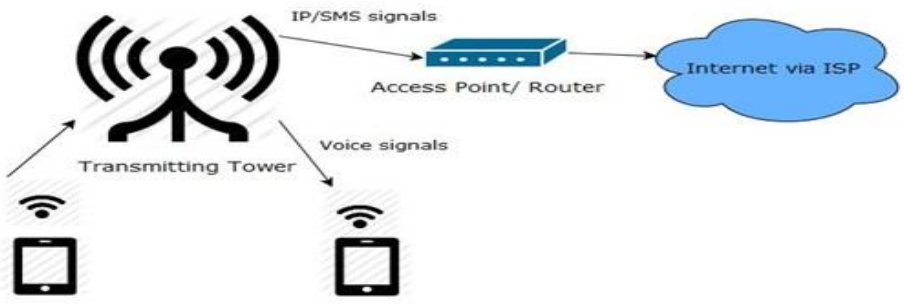
The mobile phone network is an example of a cellular network. A cellular network has a cluster of geographic locations together known as a 'cell' which connect to the Internet through satellites. Each cell has a transmitting tower at its centre through which information is passed to and fro via digital radio waves.

**Connecting to the internet:**

There are usually two ways to connect to the internet through your mobile phone – Via a cellular telephone service provider or by using standard Wi-Fi.

A Wi-Fi enabled device lets you surf the Web at free Wi-Fi hotspots. Through a cellular service provider, the phone connects to the Internet through data transfer the same way a PC does, but with a wireless link. We can access the web applications using a Wireless Application Protocol (WAP)-enabled cell phone. WAP is the universal standard for wireless communications and applications.

For operating mobile phone networks, Global System for Mobile Communications (GSM) and Code Division Multiple Access (CDMA) are the most commonly deployed. GSM and CDMA use different algorithms which allow multiple mobile phone users to share the same digital radio frequency without causing interfering for each other.




The voice and data channels of cell phones are separated for maximum efficiency - Mobile Voice goes in one channel and IP or SMS signalling over Mobile Internet in another. The General Packet Radio Service (GPRS) network provides a gateway to the internet through different frequency channels for uploading and downloading.

Now, let's see what goes behind the transfer of data between a wireless device and the Internet. The main component is Radio frequency (RF) energy which can be transmitted throughout a building passing through walls and other objects. This RF energy is transmitted to carry the information between your phone and the Internet. A Modem gets the information onto and off

the RF carrier by modulation and demodulation. The information through the RF carrier is sent in packets which have a source and destination address, very similar to the postal delivery service.

A router directs each packet to its destination and also provides a wireless access point to the Internet. A Wireless Access Point enables sharing an Internet connection by letting several computers wirelessly share Internet access through a single connection. The Internet Service Provider administers an Internet access point, for example a cellular radio tower, which may need to be accessible over long distances.

**Overview of WAP**

- **Wireless** − Lacking or not requiring a wire or wires pertaining to radio transmission.

  **(**The **transmission** of signals through space at **radio** frequencies by **means** of radiated electromagnetic waves.)

- **Application** − A computer program or piece of computer software that is designed to do a specific task.

- **Protocol** − A set of technical rules about how information should be transmitted and received using computers.

WAP is the set of rules governing the transmission and reception of data by computer applications on or via wireless devices like mobile phones.

Wireless is a term used to describe telecommunications in which electromagnetic waves (rather than some form of wire) carry the signal over part or the entire communication path. In the 21-century, wireless communication and high-speed communication incorporating computation power, mobile network/internet access capability, and consumer electronics, become emerging technologies. Wireless communications are transmitted through the air via radio waves of various frequencies. Radio frequency radiation (RFR) is one of several types of electromagnetic radiation. Data transmission in a Wireless Communication is done by means of an unguided medium. Antennas are used to transmit the signal. There are different kinds of antennas like Whip, Panel and Dish. Antennas need to be placed at specific heights in relation to one another in order to transmit and receive signals. As a result, height is a determining factor in the design and siting of wireless communications facilities.

Browsing on the internet is not restricted anymore to desktop computers, people can now also use their phones. This is done by WAP, the Wireless Application Protocol. WAP is a protocol stack for wireless communication networks, specified by the WAP forum. WAP is an application protocol is used to access services and information.

WAP uses WTLS, a wireless variant of the SSL/TLS protocol, to secure the communication between the mobile phone and other parts of the WAP architecture.

WAP is designed to work with all wireless network technologies (e.g., GSM (Global System for Mobile Communications, CDMA (Code Division Multiple Access), and TDMA (Time Division Multiple Access)). WAP is based on existing Internet standards, such as IP, XML, HTML, and HTTP etc.,

**Wireless Transport Layer Security** (**WTLS**) is a security protocol, part of the Wireless Application Protocol (WAP) stack. It sits between the WTP and WDP layers in the WAP communications stack.

Wireless Transport Layer Security (WTLS) is the security level for Wireless Application Protocol (WAP) applications. Based on Transport Layer Security (TLS) v1.0 (a security layer used in the Internet, equivalent to Secure Socket Layer 3.1), WTLS was developed to address the problematic issues surrounding mobile network devices - such as limited processing power and memory capacity, and low bandwidth - and to provide adequate authentication, data integrity, and privacy protection mechanisms.

**Origin of WAP**

On June 26, 1997, Ericsson, Motorola, Nokia, and Unwired Planet took the initiative to start a rapid creation of a standard for making advanced services within the wireless domain a reality. In December 1997, WAP Forum was formally created and after the release of the WAP 1.0 specifications in April 1998, WAP Forum membership was opened to all.

The WAP Forum now has over 500 members and represents over 95 percent of the global handset market. Companies such as Nokia, Motorola and Ericsson are all members of the forum.

The objective of the forum is to create a license-free standard that brings information and telephony services to wireless devices.

A **WAP browser** is a web browser for mobile devices such as mobile phones that uses the protocol. Introduced with much hype in 1999, WAP achieved some popularity in the early 2000s. A re-engineered 2.0 version was released in 2002.

**WAP Architecture:**

The WAP architecture provides a scalable and extensible environment for application development for mobile communication devices. This is achieved through a layered design of the entire protocol stack (Figure 4). Each of the layers of the architecture is accessible by the layers above, as well as by other services and applications.

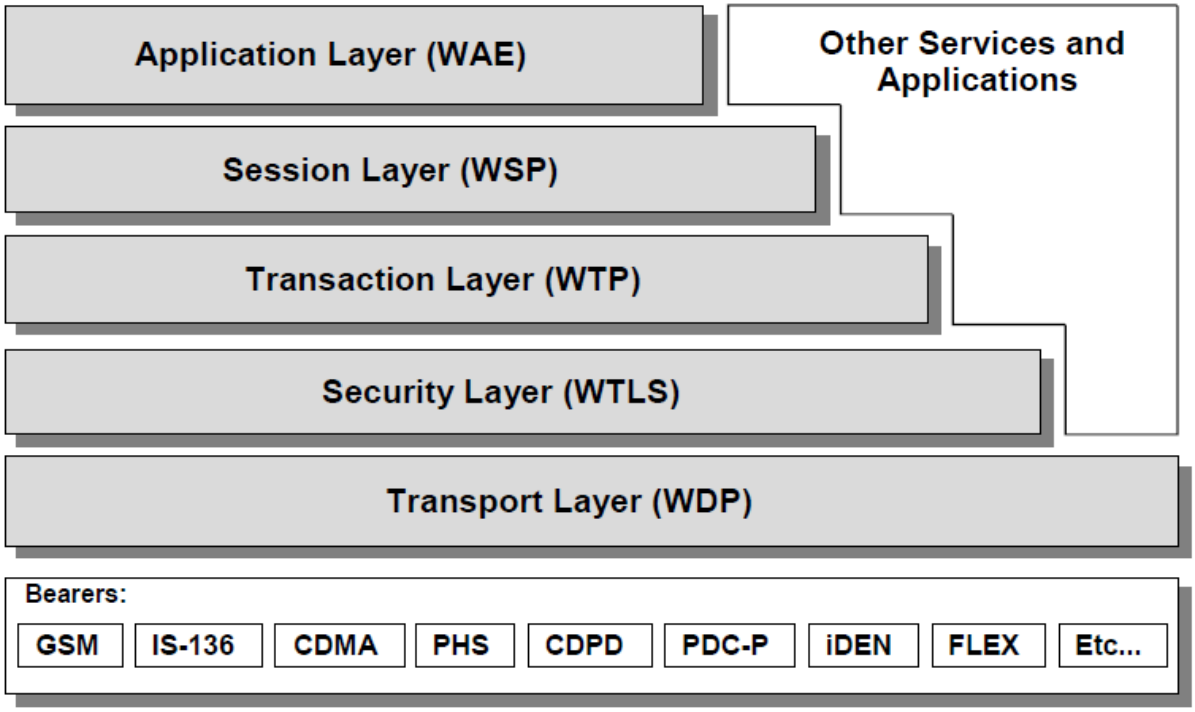



**Figure 4: The WAP Architecture**

The WAP layered architecture enables other services and applications to utilise the features of the WAP stack through a set of well-defined interfaces. External applications may access the session, transaction, security and transport layers directly. The following sections provide a description of the various elements of the protocol stack architecture.

**Wireless Application Environment (WAE)**

The Wireless Application Environment (WAE) is a general-purpose application environment based on a combination of World Wide Web (WWW) and Mobile Telephony technologies. 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. WAE includes a micro-browser environment containing the following functionality:

· Wireless Markup Language (WML) – a lightweight markup language, similar to HTML, but optimised for use in hand-held mobile terminals;

· WMLScript – a lightweight scripting language, similar to JavaScript™;

· Wireless Telephony Application (WTA, WTAI) – telephony services and programming interfaces; and

· Content Formats – a set of well-defined data formats, including images, phone book records and calendar information.

**Wireless Session Protocol (WSP)**

The Wireless Session Protocol (WSP) provides the application layer of WAP with a consistent interface for two session services. The first is a connection-oriented service that operates above the transaction layer protocol WTP. The second is a connectionless service that operates above a secure or non-secure datagram service (WDP). The Wireless Session Protocols currently consist of services suited for browsing applications (WSP/B). WSP/B provides the following functionality:

· HTTP/1.1 functionality and semantics in a compact over-the-air encoding,

· Long-lived session state,

· Session suspend and resume with session migration,

· A common facility for reliable and unreliable data push, and

· Protocol feature negotiation.

The protocols in the WSP family are optimised for low-bandwidth bearer networks with relatively long latency.

WSP/B is designed to allow a WAP proxy to connect a WSP/B client to a standard HTTP server.

**Wireless Transaction Protocol (WTP)**

The Wireless Transaction Protocol (WTP) runs on top of a datagram service and provides as a light-weight transaction-oriented protocol that is suitable for implementation in “thin” clients (mobile stations). WTP operates efficiently over secure or non-secure wireless datagram networks and provides the following features:

· Three classes of transaction service:

· Unreliable one-way requests,
· Reliable one-way requests, and
· Reliable two-way request-reply transactions;
· Optional user-to-user reliability - WTP user triggers the confirmation of each received message;
· Optional out-of-band data on acknowledgements;
· PDU concatenation and delayed acknowledgement to reduce the number of messages sent;
· Asynchronous transactions.

**Wireless Transport Layer Security (WTLS)**

WTLS is a security protocol based upon the industry-standard Transport Layer Security (TLS) protocol, formerly known as Secure Sockets Layer (SSL). WTLS is intended for use with the WAP transport protocols and has been optimised for use over narrow-band communication channels. WTLS provides the following features:

· Data integrity – WTLS contains facilities to ensure that data sent between the terminal and an application server is unchanged and uncorrupted.

· Privacy – WTLS contains facilities to ensures that data transmitted between the terminal and an application server is private and cannot be understood by any intermediate parties that may have intercepted the data stream.

· Authentication – WTLS contains facilities to establish the authenticity of the terminal and application server.

· Denial-of-service protection – WTLS contains facilities for detecting and rejecting data that is replayed or not successfully verified. WTLS makes many typical denial-of-service attacks harder to accomplish and protects the upper protocol layers.

WTLS may also be used for secure communication between terminals, eg, for authentication of electronic business card exchange.

Applications are able to selectively enable or disable WTLS features depending on their security requirements and the characteristics of the underlying network (eg, privacy may be disabled on networks already providing this service at a lower layer).

**Wireless Datagram Protocol (WDP)**

The Transport layer protocol in the WAP architecture is referred to as the Wireless Datagram Protocol (WDP).

The WDP layer operates above the data capable bearer services supported by the various network types. As a general transport service, WDP offers a consistent service to the upper

layer protocols of WAP and communicate transparently over one of the available bearer services.

Since the WDP protocols provide a common interface to the upper layer protocols the Security, Session and Application layers are able to function independently of the underlying wireless network. This is accomplished by adapting the transport layer to specific features of the underlying bearer. By keeping the transport layer interface and the basic features consistent, global interoperability can be achieved using mediating gateways.

**Bearers**

The WAP protocols are designed to operate over a variety of different bearer services, including short message, circuit-switched data, and packet data. The bearers offer differing levels of quality of service with respect to throughput, error rate, and delays. The WAP protocols are designed to compensate for or tolerate these varying levels of service.

Since the WDP layer provides the convergence between the bearer service and the rest of the WAP stack, the WDP specification [WDP] lists the bearers that are supported and the techniques used to allow WAP protocols to run over each bearer. The list of supported bearers will change over time with new bearers being added as the wireless market evolves.

**Other Services and Applications**

The WAP layered architecture enables other services and applications to utilise the features of the WAP stack through a set of well-defined interfaces. External applications may access the session, transaction, security and transport layers directly. This allows the WAP stack to be used for applications and services not currently specified by WAP, but deemed to be valuable for the wireless market. For example, applications, such as electronic mail, calendar, phone book, notepad, and electronic commerce, or services, such as white and yellow pages, may be developed to use the WAP protocols.

**The WAP Model**

The World-Wide Web Model The Internet World-Wide Web (WWW) architecture provides a very flexible and powerful programming model (Figure 1). Applications and content are presented in standard data formats, and are browsed by applications known as web browsers. The web browser is a networked application, i.e., it sends requests for named data objects to a

network server and the network server responds with the data encoded using the standard formats.

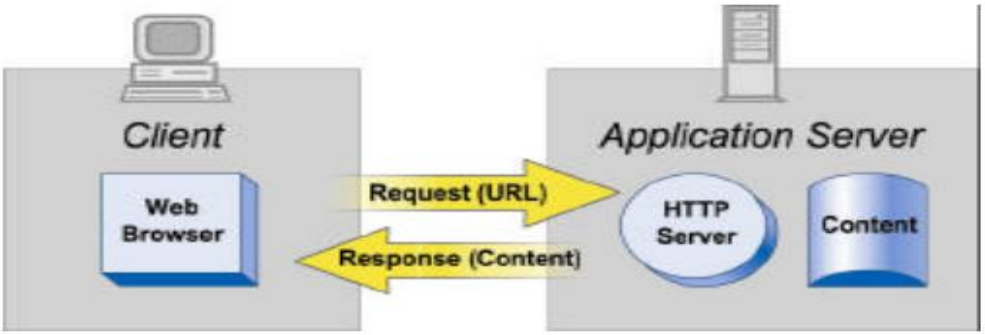



Figure 1. World-Wide -Web Programming Model

The WWW standards specify many of the mechanisms necessary to build a general-purpose application environment, including:

**Standard naming model** – All servers and content on the WWW are named with an Internet-standard Uniform Resource Locator (URL)

**Content typing** – All content on the WWW is given a specific type thereby allowing web browsers to correctly process the content based on its type.

**Standard content formats** – All web browsers support a set of standard content formats. These include the HyperText Markup Language (HTML), scripting languages [JavaScript], and a large number of other formats.

**Standard Protocols** – Standard networking protocols allow any web browser to communicate with any web server. The most commonly used protocol on the WWW is the HyperText Transport Protocol (HTTP), operating on top of the TCP/IP protocol suite

**The WAP Model**

The WAP programming model (Figure 2) is similar to the WWW programming model. This provides several benefits to the application developer community, including a familiar programming model, a proven architecture, and the ability to leverage existing tools (eg, Web servers, XML tools, etc.). Optimisations and extensions have been made in order to match the characteristics of the wireless environment. Wherever possible, existing standards have been adopted or have been used as the starting point for the WAP technology.

**WAP Step-By-Step**

1. A user requests a URL by entering it into a WAP device.

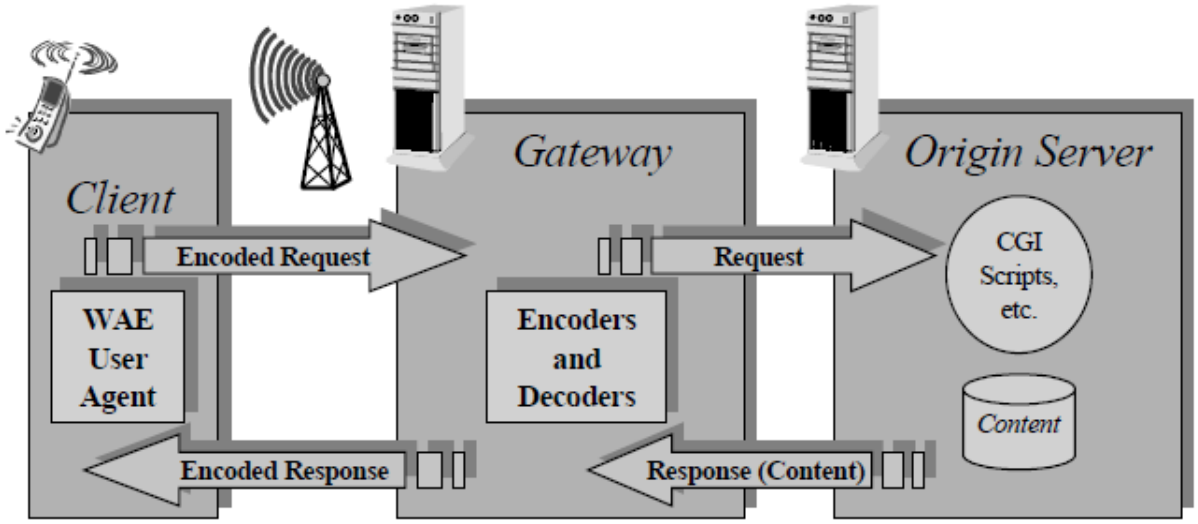
2. A WAP device encodes the request into an encrypted, compact binary format suitable for transmission over a wireless link and sends it to the WAP gateway.

3. The gateway examines the message converts it into a valid HTTP-based URL request and forwards it.

4. When wmlserver.com receives the request, it fulfills it by returning the requested document back to the gateway.

5. The gateway converts the HTTP response back into an encrypted, binary format and ships it off to the WAP device.

6. The WAP device decodes the response and displays the results on the WAP device's screen.




WAP defines a set of standard components that enable communication between mobile terminals and network

servers, including:

**Standard naming model** – WWW-standard URLs are used to identify WAP content on origin servers. WWW-standard URIs are used to identify local resources in a device, eg call control functions.

**Content typing** – All WAP content is given a specific type consistent with WWW typing. This allows WAP user agents to correctly process the content based on its type.

**Standard content formats** – WAP content formats are based on WWW technology and include display markup, calendar information, electronic business card objects, images and scripting language.

**Standard communication protocols** – WAP communication protocols enable the communication of browser requests from the mobile terminal to the network web server.

## Example WAP Network

The following is for illustrative purposes only. An example WAP network is shown in Figure 3.

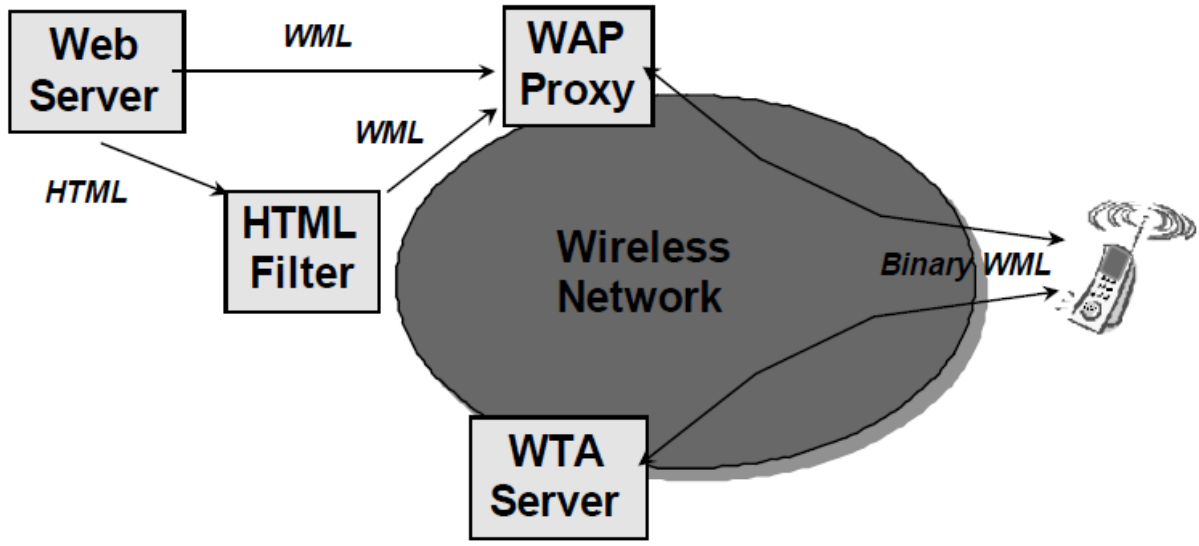



**Figure 3. Example WAP Network**

In the example, the WAP client communicates with two servers in the wireless network. The WAP proxy translates WAP requests to WWW requests thereby allowing the WAP client to submit requests to the web server.

The proxy also encodes the responses from the web server into the compact binary format understood by the client.

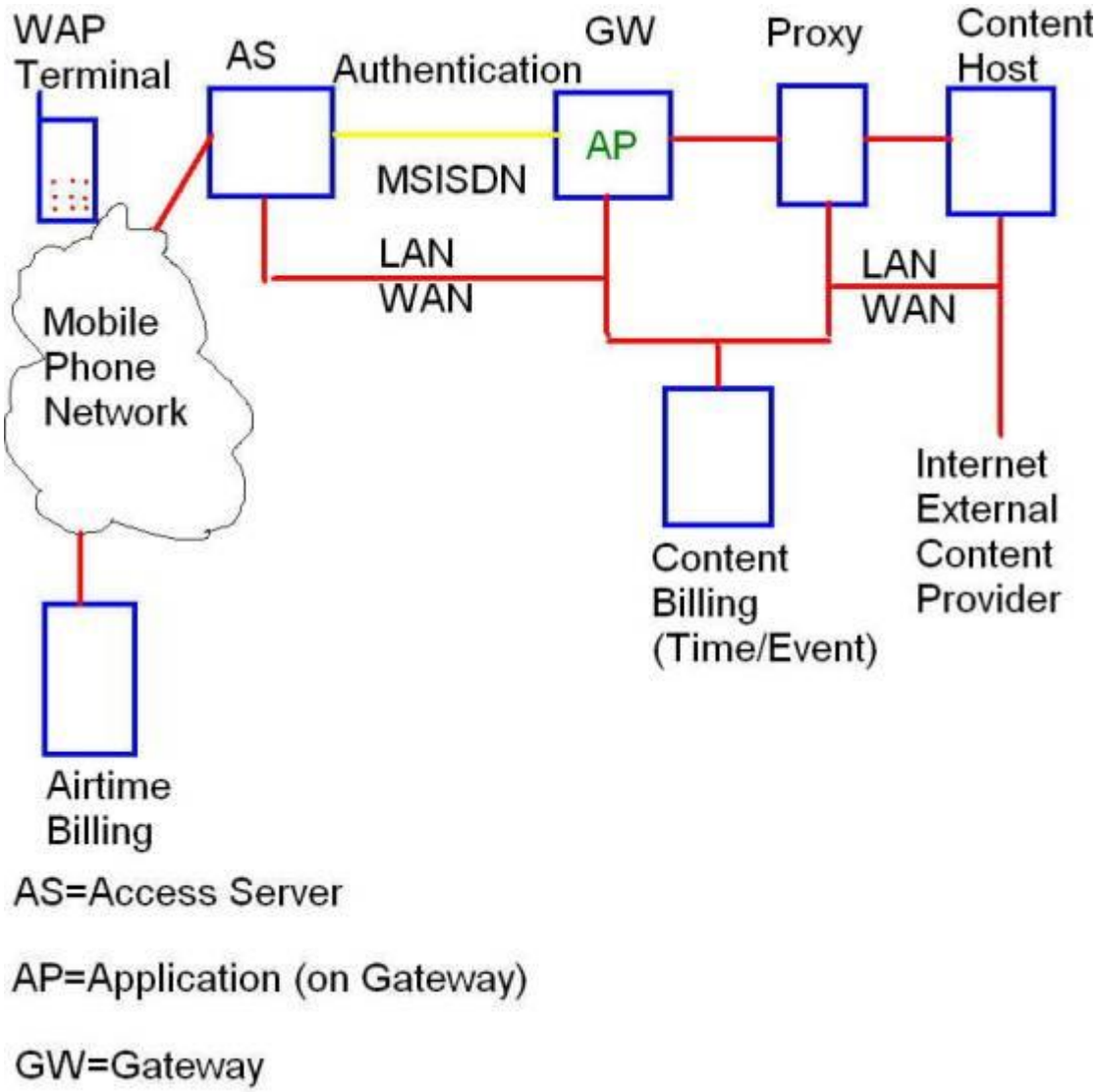
If the web server provides WAP content (e.g., WML), the WAP proxy retrieves it directly from the web server.
However, if the web server provides WWW content (such as HTML), a filter is used to translate the WWW content into WAP content. For example, the HTML filter would translate HTML into WML.

The Wireless Telephony Application (WTA) server is an example origin or gateway server that responds to requests from the WAP client directly. The WTA server is used to provide WAP access to features of the wireless network provider's telecommunications infrastructure.

**Components of WAP Standards**

Here we will describe the typical components for the setup of a WAP service.

The following figure shows how the components in a WAP service interact.




**WAP terminals**

Users of WAP services access WAP content with a WAP terminal.

The following devices can serve as WAP terminals:

- Mobile phones with built-in GSM modem, WAP stack, and microbrowser (e.g. Nokia)
- Palmtops with GSM phone connected and additional WAP/ browser software

Depending on the features of the device used and the capabilities of its display, there are different quality degrees with WAP services. However, the following features are common to all terminals:

- Access to pull services, or browsing, using GSM Circuit Switched Data (CSD), usually at 9600 bps, Point to Point Protocol (PPP) via the V.110 protocol.
- Configurable access profile (PPP authentication parameters, access number of access server, Gateway IP address, homepage, etc...)
- Optional support of push services using connectionless bearers (for example the Short Message Service)
- Access to/ from network carrier
- Access to the WAP Gateway in Circuit Switched Data (CSD) connections is enabled through Integrated Services Digital Network (ISDN) bearer channels, packed in E1 PRI connections with 30 channels in each connection. The number of available bearer channels determines the number of WAP terminals that can actively access the WAP Gateway simultaneously.

**Airtime Billing**

Billing of the base connection time (base rate for WAP online time) must always be performed by the network carrier.

**Access Server**

The access server is the PPP termination point for the WAP terminal. One of its responsibilities is allocating dynamic IP addresses to the active WAP terminals.

The access server ensures correct routing of WAP packets from and to the WAP Gateway and performs firewall duties.

By cooperating with the access server and the RADIUS server (enables authentication methods), the WAP Gateway can call up the MSISDN of a WAP terminal in an open session in order to perform personalization and authentication functions. The access server offers RADIUS support for the implementation of authentication mechanisms.

**WAP Gateway**

The WAP Gateway is at the core of the WAP service. It implements the WAP stack (WDP, WTLS, WTP, WSP), the necessary content encoders, and a variety of management functions.

By cooperating with the WAP/ HTTP proxy and using the authentication information originating from the access server, time or event-based billing can be implemented on the Gateway side, in addition to airtime billing.

The WAP Gateway is a device for converting the TCP/IP protocols to the different WAP protocols and vice versa. It is able to translate HTML to WML. The WAP Gateway is often a proxy server, meaning that it acts both as server and client for the purpose of making requests on behalf of other clients. In this case it resides between WAP clients and web servers often supporting only access to HTML pages via HTTP. These clients and servers have no means of direct communication, which is really the whole problem with the WAP technology. For instance, the gateway will lie behind a firewall, that is, the same as the web servers, but there is no firewall that protects the area between the WAP client and the gateway. There are solutions where the gateway is an origin server. That is probably a more convenient solution, though it's not the issue that should be discussed in this paper.

**WAP/ HTTP proxy**

The WAP/ HTTP proxy is the interface to the services that actually provide the requested WAP content. These services (WAP Originating Services) can either be requested via the public Internet, or they can be installed on the content host.

Using a built-in cache mechanism, the proxy optimizes access to external services and ensures fast access times.

**Content Host**

The content host is a dedicated HTTP server that is connected to the WAP proxy server via a LAN or WAN and facilitates fast access to (partially) exclusive WAP services. External content providers can install and administrate WAP services on the content host using standard protocols, such as File Transfer Protocol (FTP).

**The end-to-end concept**

The end-to-end concept is dealing with choosing the proper boundaries for functions. That could mean figuring out where to for instance put the translation of HTML to WML in a WAP

network. The end-to-end effect of a specific device in a certain system simply means: how will this device have an effect on the whole system, and specifically on the systems two terminal points.

**How a simple WAP network works**

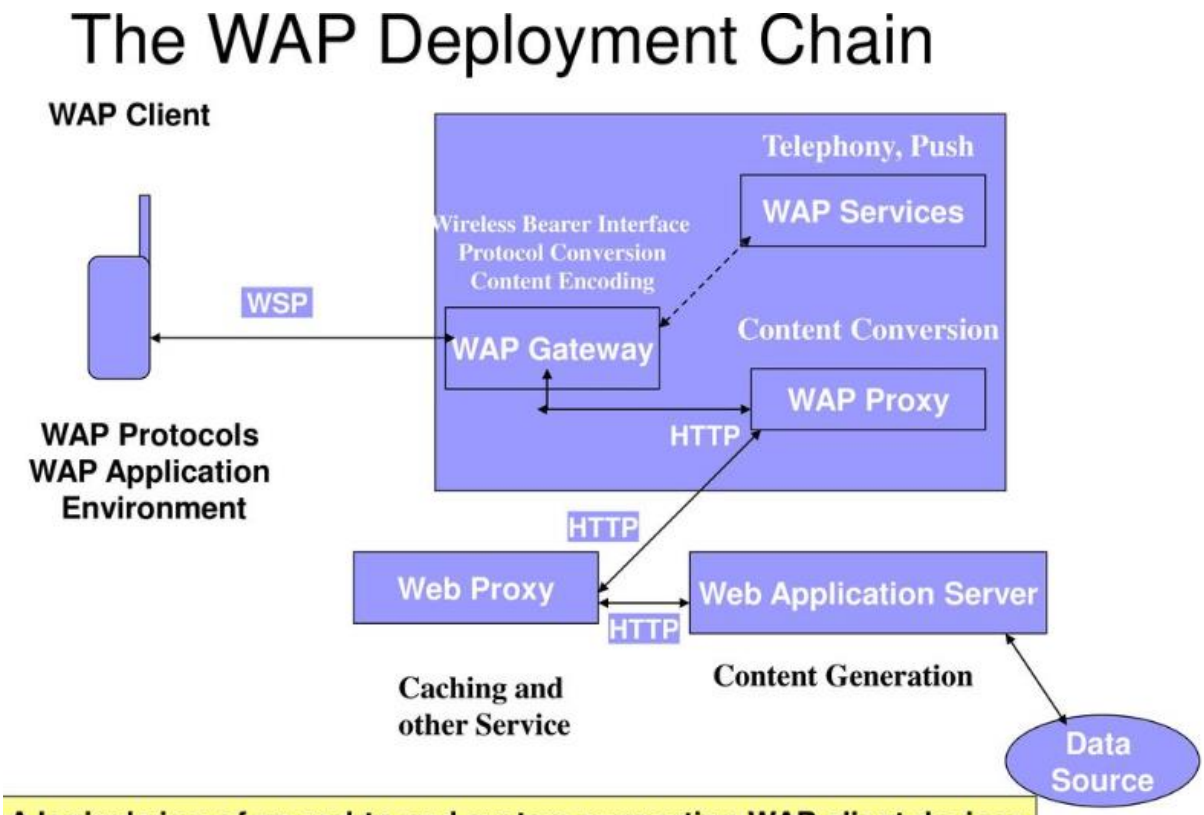
Imagine a simple WAP network, with a web server supporting only access to HTML pages via HTTP, a WAP Gateway, and a WAP client. The WAP client wants to browse some pages lying on the web server. The wap client sends a simple WSP request that is translated to a GET or POST request (in this case an HTTP GET URL) at the gateway. The gateway usually has an built in browser, sometimes called a web wrapper, that browses the requested page and gathers the corresponding HTML document from the web server. The gateway then converts the HTML document to WML, caches the page and sends it to the WAP client in binary form.

So that's briefly what happens. Now let's take a look at the issues we should deal with, namely how the gateway affects the end-to-end performance of the network. To start with we could look at some problems.

**Network Infrastructure Services Supporting WAP Clients**

**Mobile Client -> WAP std -> access data servers**

To support the delivery of info to WAP clients, N/w infrastructure performs services like: Generate content in std forms Convert that contents into formats (WML, WML Script, etc) that are recognized by the WAP application environment Compile and encode the content into binary form using WBXML, WML Script byte codes Bridge protocols between the WAP suite and the Internet suite Bridge the physical wireless n/w with the wireline n/w Other services like content caching, filtering and access control , telephony and server-initiated content push




A logical view of an end-to-end system supporting WAP client devices

**WAP client**

Executes WAP Appl. Env.

WML browser, WML Script engine, push client, telephony appl. Env are located

Mobile, PDA

**WAP gateway**

WSP -> HTTP and back, binary encoding.

IBM Secure way Wireless Gateway, Phone.com UP. Link server , Ericsson, Nokia

**WAP proxy**

Necessary manipulations to WAP content on behalf of the WAP client – content conversion, filtering, customization

**WAP services**

Provide capabilities not provided by web infrastructure like WAP push, telephony server

**Web proxy**

Standard proxy providing services like content caching, insertion and site-based content filtering – IBM's Web Traffic Express

**Web Server or Web Application Server**

Generate content to be delivered to the client in various formats like WML, HTML, XML, WML Script, vCard, vCalendar

IBM’s WebSphere Application Server

**Data Source**

Repository for data being sent to the user or the system that processes transaction requests RDBMS, IBM’s DB2 or Oracle’s 8i

**WAP Architecture Design Principles**

Layers of WAP Protocol

**Application Layer**

**Wireless Application Environment** (WAE). This layer is of most interest to content developers because it contains among other things, device specifications, and the content development programming languages, WML, and WML Script.

**Session Layer**

Wireless Session Protocol (WSP). Unlike HTTP, WSP has been designed by the WAP Forum to provide fast connection suspension and reconnection.

**Transaction Layer**

Wireless Transaction Protocol (WTP). The WTP runs on top of a datagram service, such as User Datagram Protocol (UDP) and is part of the standard suite of TCP/IP protocols used to provide a simplified protocol suitable for low bandwidth wireless stations.

**Security Layer**

Wireless Transport Layer Security (WTLS). WTLS incorporates security features that are based upon the established Transport Layer Security (TLS) protocol standard. It includes data integrity checks, privacy, service denial, and authentication services.

**Transport Layer**

Wireless Datagram Protocol (WDP). The WDP allows WAP to be bearer-independent by adapting the transport layer of the underlying bearer. The WDP presents a consistent data format to the higher layers of the WAP protocol stack, thereby offering the advantage of bearer independence to application developers.

Each of these layers provides a well-defined interface to the layer above it. This means that the internal workings of any layer are transparent or invisible to the layers above it. The layered architecture allows other applications and services to utilise the features provided by the WAP-stack as well. This makes it possible to use the WAP-stack for services and applications that currently are not specified by WAP.

To begin, we will look at how WAP conforms to the Open Systems Interconnection (OSI) model as defined by the International Standards Organization (ISO). The OSI model consists of seven distinct layers, six of which are depicted in Figure 6 as they relate to the WAP architecture. The physical layer is not shown; it sits below the network layer and defines the physical aspects such as the hardware and the raw bit-stream. For each of the other six layers, WAP has a corresponding layer, which will now be described in more depth.

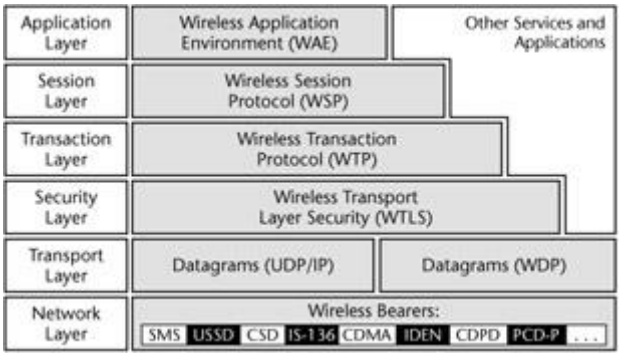



Figure 6: WAP architecture and its relationship to the OSI model.

***Wireless Application Environment (WAE)***

The Wireless Application Environment (WAE) is the application layer of the OSI model. It provides the required elements for interaction between Web applications and wireless clients using a WAP microbrowser. These elements are as follows:

- A specification for a microbrowser that controls the user interface and interprets WML and WMLScript.
- The foundation for the microbrowser in the form of the Wireless Markup Language (WML). WML has been designed to accommodate the unique characteristics of wireless devices, by incorporating a user interface model that is suitable for small form-factor devices that do not have a QWERTY keyboard.
- A complete scripting language called WMLScript that extends the functionality of WML, enabling more capabilities on the client for business and presentation logic.
- Support for other content types such as wireless bitmap images (WBMP), vCard, and vCalendar.

WAP 2.x extends WAE by adding the following elements:

- A new markup language specification called WML2 that is based on XHTML-Basic. Backward compatibility with WML1 has been maintained.

- Support for stylesheets to enhance presentation capabilities. Stylesheet support is based on the Mobile Profile of Cascading Style Sheets (CSS) from the W3C, and supports both inline and external style sheets.

Note WAP 2.x WAE has backward compatibility to WML1. This is accomplished either via built-in support for both languages or by translating WML1 into WML2 using eXtensible Stylesheet Language Transformation (XSLT). The method used depends on the implementation by the device manufacturer.

## UNIT II THE WIRELESS MARKUP LANGUAGE

### INTRODUCTION

The topmost layer in the WAP (Wireless Application Protocol) architecture is made up of WAE (Wireless Application Environment), which consists of WML and WML scripting language.

WAP homepages are not very different from HTML homepages. The markup language used for WAP is WML (Wireless Markup Language). WML uses tags - just like HTML - but the syntax is stricter and conforms to the XML 1.0 standard. (eXtensible Markup Language) WML pages have the extension *.WML

### WML Tags

WML is mostly about text. Tags that would slow down the communication with handheld devices are not a part of the WML standard. The use of tables and images is strongly restricted.

Since WML is an XML application, all tags are case sensitive (<wml> is not the same as <WML>), and all tags must be properly closed.

WML is an XML language used to specify content and user interface for WAP devices like PDA and Mobile Phones. The WAP forum provides a DTD for WML.

- WML stands for **W**ireless **M**arkup **L**anguage
- WML is an application of XML, which is defined in a document-type definition.
- WML is based on HDML and is modified so that it can be compared with HTML.
- WML takes care of the small screen and the low bandwidth of transmission.
- WML is the markup language defined in the WAP specification.
- WAP sites are written in WML, while web sites are written in HTML.
- WML is very similar to HTML. Both of them use tags and are written in plain text format.

- WML files have the extension ".wml". The MIME type of WML is "text/vnd.wap.wml".
- WML supports client-side scripting. The scripting language supported is called WMLScript.

**WAP Site Design Considerations:**

Wireless devices are limited by the size of their displays and keypads. It's therefore very important to take this into account when designing a WAP Site.

While designing a WAP site you must ensure that you keep things simple and easy to use. You should always keep in mind that there are no standard microbrowser behaviors and that the data link may be relatively slow, at around 10Kbps. However, with GPRS, EDGE, and UMTS, this may not be the case for long, depending on where you are located.

The following are general design tips that you should keep in mind when designing a service:

- Keep the WML decks and images to less than 1.5KB.
- Keep text brief and meaningful, and as far as possible try to precode options to minimize the rather painful experience of user data entry.
- Keep URLs brief and easy to recall.
- Minimize menu levels to prevent users from getting lost and the system from slowing down.
- Use standard layout tags such as <big> and <b>, and logically structure your information.
- Don't go overboard with the use of graphics, as many target devices may not support them.

**WML Decks and Cards:**

WML pages are often called "decks". A deck contains a set of cards. A card element can contain text, markup, links, input-fields, tasks, images and more. Cards can be related to each other with links.

When a WML page is accessed from a mobile phone, all the cards in the page are downloaded from the WAP server. So, if the user goes to another card of the same deck, the mobile browser does not have to send any requests to the server since the file that contains the deck is already stored in the wireless device.

You can put links, text, images, input fields, option boxes and many other elements in a card.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card id="one" title="First Card">
<p>
This is the first card in the deck
</p>
</card>
<card id="two" title="Second Card">
<p>
This is the second card in the deck
</p>
</card>
</wml>
```

The first line of this text says that this is an XML document and the version is 1.0. The second line selects the document type and gives the URL of the document type definition (DTD).

A DTD defines the valid building blocks of an XML **document**. It is set of markup declarations that **define** a **document type** for a XML, HTML etc. A DTD can be declared inline inside an XML **document**

Unlike HTML 4.01 Transitional, text cannot be enclosed directly in the <card>...</card> tag pair. So you need to put a content inside ... as shown above.

As you can see from the example, the WML document is an XML document. The DOCTYPE is defined to be wml, and the DTD is accessed at www.wapforum.org/DTD/wml_1.1.xml.

The document content is inside the <wml>...</wml> tags. Each card in the document is inside <card>...</card> tags, and actual paragraphs are inside ... tags. Each card element has an id and a title.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>

<card id="no1" title="Card 1">
<p>Hello World!</p>
</card>

<card id="no2" title="Card 2">
<p>Welcome to our WAP Tutorial!</p>
</card>
</wml>
```

The result MIGHT look like this in your mobile phone display (note that only one card is displayed at a time):

Result:

```
------ Card 1 ------

Hello World!
```

**Paragraphs and Line Breaks**

A WML card can be set up to display the paragraph and line break functions of WML:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Paragraphs">
<p>
This is a paragraph
</p>
<p>
This is another<br/>with a line break
</p>
</card>
</wml>
```

The result MIGHT look like this in your mobile phone display:

```
------ Paragraphs ------
```

This is a paragraph

This is another
with a line break

**Text Formatting**

A WML card can be set up to display the text formatting functions of WML:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Formatting">

<p>
normal<br/>
<em>emphasized</em><br/>
<strong>strong</strong><br/>
<b>bold</b><br/>
<i>italic</i><br/>
<u>underline</u><br/>
<big>big</big><br/>
<small>small</small>
</p>

</card>
</wml>
```

The result look like this in your mobile phone display

----- Formatting -----

normal
*emphasized*
**strong**
**bold**
*italic*
underline
big
small

The <b>, <i> and <u> tags mean bold, italic and underline respectively.

The <small> and <big> tags are used to reduce and increase the text size respectively.

The emphasis () and strong emphasis () styles are similar. A WAP browser will display text with the emphasis or strong emphasis style in some way that makes it more noticeable. The actual effect depends on the mobile device.

**Tables**

A WML card can be set up to display the table functions of WML:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Table">
<p>
<table columns="3">
<tr>
<td>Cell 1</td>
<td>Cell 2</td>
<td>Cell 3</td>
</tr>
</table>
</p>
</card>
</wml>
```

The result look like this in your mobile phone display:

```
------ Table ------

Cell 1   Cell 2   Cell 3
```

**Links**

A WML card can be set up to display the anchor functions of WML.

**<anchor> TAG:**

The <anchor> tag always has a task ("go", "prev", or "refresh") specified. The task defines what to do when the user selects the link. In this example, when the user selects the "Next page" link, the task says "go to the file test.wml":

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Anchor Tag">
<p>
<anchor>Next page
<go href="test.wml"/>
</anchor>
</p>
</card>
</wml>
```

<a>

The <a> tag always performs a "go" task, with no variables. The example below does the same as the <anchor> tag example:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="A Tag">
<p>
<a href="test.wml">Next page</a>
</p>
</card>
</wml>
```

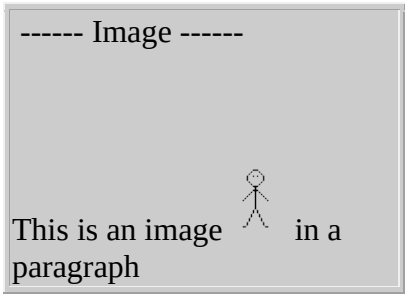
**Image**

A WML card can be set up to display an image:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Image">
<p>
This is an image
<img src="/images/stickman.wbmp" alt="stickman" />
in a paragraph
</p>
</card>
</wml>
```

The result look like this in your mobile phone display:




Note that .wbmp is the only image type that can be displayed in a WAP browser.

**WML entities**

WML entities are to represent symbols that either can't easily be typed in or that have a special meaning in WML.

For example, if you put a < character into your text normally, the browser thinks it's the start of a tag; the browser then complains when it can't find the matching > character to end the tag.

Following table displays the three forms of entities in WML. Named entities are something you may be familiar with from HTML: they look like & or <, and they represent a

single named character via a mnemonic name. Entities can also be entered in one of two numeric forms (decimal or hexadecimal), allowing you to enter any Unicode character into your WML.

**Character Entities**

| Result | Description | Entity Name | Entity Number |
|---|---|---|---|
| & | Ampersand | &amp; | &#38; |
| ' | Apostrophe | &apos; | &#39; |
| > | greater-than | &gt; | &#62; |
| < | less-than | &lt; | &#60; |
| | non-breaking space |   |   |
| " | quotation mark | &quot; | &#34; |
| | soft hyphen | &shy; | &#173; |

| Named Entity | Decimal Entity | Hexa Entity | Character |
|---|---|---|---|
| &quot; | &#34; | &#x22; | Double quote (") |
| &amp; | &#38; | &#x26; | Ampersand (&) |
| &apos; | &#39; | &#x27; | Apostrophe (') |
| &lt; | &#60; | &#x3C; | Less than (<) |
| &gt; | &#62; | &#x3E; | Greater than (>) |
|   |   |   | Nonbreaking space |
| &shy; | &#173; | &#xAD; | Soft hyphen |

Note that all entities start with an ampersand ( &) and end with a semicolon ( ;). This semicolon is very important: some web pages forget this and cause problems for browsers that want correct HTML. WAP browsers also are likely to be stricter about errors like these.

**WML Elements**

WML is defined by a set of *elements* that specify all markup and structural information for a WML deck. Elements are identified by tags, which are each enclosed in a pair of angle brackets.

Unlike HTML, WML strictly adheres to the XML hierarchical structure, and thus, elements must contain a start tag; any content such as text and/or other elements; and an end tag. Elements have one of the following two structures:

- **<tag> content </tag>** : This form is identical to HTML.
- **<tag />:** This is used when an element cannot contain visible content or is empty, such as a line break. WML document's prolog part does not have any element which has closing element.

Following table lists the majority of valid elements. A complete detail of all these elements is given in WML Tags Reference.

Deck & Card Elements

| WML Elements | Purpose |
|---|---|
| <!--> | Defines a WML comment |
| <wml> | Defines a WML deck (WML root) |
| <head> | Defines head information |
| <meta> | Defines meta information |
| <card> | Defines a card in a deck |

| | |
|---|---|
| <access> | Defines information about the access control of a deck |
| <template> | Defines a code template for all the cards in a deck |

Text Elements

| WML Elements | Purpose |
|---|---|
| <br> | Defines a line break |
| <p> | Defines a paragraph |
| <table> | Defines a table |
| <td> | Defines a table cell (table data) |
| <tr> | Defines a table row |
| <pre> | Defines preformatted text |

Text Formatting Tags

| WML Elements | Purpose |
|---|---|
| <b> | Defines bold text |
| <big> | Defines big text |
|  | Defines emphasized text |

| | |
|---|---|
| <i> | Defines italic text |
| <small> | Defines small text |
|  | Defines strong text |
| <u> | Defines underlined text |

Image Elements

| WML Elements | Purpose |
|---|---|
| <img> | Defines an image |

Anchor Elements

| WML Elements | Purpose |
|---|---|
| <a> | Defines an anchor |
| <anchor> | Defines an anchor |

Event Elements

| WML Elements | Purpose |
|---|---|
| <do> | Defines a do event handler |
| <onevent> | Defines an onevent event handler |
| <postfield> | Defines a postfield event handler |

| | |
|---|---|
| <ontimer> | Defines an ontimer event handler |
| <onenterforward> | Defines an onenterforward handler |
| <onenterbackward> | Defines an onenterbackward handler |
| <onpick> | Defines an onpick event handler |

Task Elements

| WML Elements | Purpose |
|---|---|
| <go> | Represents the action of switching to a new card |
| <noop> | Says that nothing should be done |
| <prev> | Represents the action of going back to the previous card |
| <refresh> | Refreshes some specified card variables. |

Input Elements

| WML Elements | Purpose |
|---|---|
| <input> | Defines an input field |
| <select> | Defines a select group |
| <option> | Defines an option in a selectable list |

| | |
|---|---|
| <fieldset> | Defines a set of input fields |
| <optgroup> | Defines an option group in a selectable list |

Variable Elements

| WML Elements | Purpose |
|---|---|
| <setvar> | Defines and sets a variable |
| <timer> | Defines a timer |

**WML Authoring**

**Design Orientation**

**Design Attributes**

An established way of orienting design activity is one based around the customer. Such a process encourages the developer to view their product from the perspective of the customer. The product should be simple to use, easy to learn, pleasant to view and satisfying. We define the following attributes that should be considered in the design of WAP applications:

- 'Usability' is defined as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (P2; ISO 9 241-11:1998(E)).
- Effectiveness means accuracy and completeness with which customers achieve specified goals (e.g. obtaining a specific share price from a stock exchange).
- Efficiency means resources expended in relation to the accuracy and completeness with which customers achieve goals (e.g. making fewer key presses to arrive at a destination; taking less time to achieve a goal, costing less to obtain some specific information).
- Satisfaction means freedom from discomfort, and positive attitudes towards the use of the product. Pleasure from using a product is often a powerful motivator for its continued use.

- Context of use refers to customers, their tasks, equipment and the physical and social environment in which their activity takes place. When designing applications choose a typical customer and model how they use the product.

**Design Hierarchy**

- In every design activity there are a number of objectives to satisfy, and here the focus is in reducing the variance in the ways an application is presented to users. Table1 shows that the burden of work should be shifted from the customer up the hierarchy towards the WAP server, where it is better to handle it. However, a great deal of effort is required to address variance at the highest levels as many parties have already deployed equipment, but the developer is often a single party and able to change application behaviour. The developer's role, through the use of a specification language, is to:

  • provide customers with an application whose behaviour is consistent at the interface; and

  • meet customers' expectations of using such an application.

A generic WAP content authoring guide for application developers has advantages for all stakeholders as it addresses the high degree of variance in interfaces.

**Table 1: WAP design hierarchy**

| Entities in application Delivery | Gain from addressing variance | Effort required to address Variance |
|---|---|---|
| WAP server | Promotes value | Most effort |
| WAP Handset | Promotes accessibility | |
| Developer | Promotes consistency and reduces development costs | |
| Customer / User | Promotes usability and satisfaction | Least effort |

**URL Identity content**

A **URL** (Uniform Resource Locator) is a unique identifier used to locate a resource on the internet. It is also referred to as a web address. **URLs** consist of multiple parts -- including a protocol and domain name -- that tell a web browser how and where to retrieve a resource.

**WML and URLs**

The World Wide Web is a network of information and devices. Three areas of specification ensure widespread interoperability:

· A unified naming model. Naming is implemented with Uniform Resource Locators (URLs), which provide standard way to name any network resource. See [RFC2396].

· Standard protocols to transport information (e.g., HTTP).

· Standard content types (e.g., HTML, WML).

WML assumes the same reference architecture as HTML and the World Wide Web. Content is named using URLs and is fetched over standard protocols that have HTTP semantics, such as [WSP]. URLs are defined in [RFC2396]. The character set used to specify URLs is also defined in [RFC2396].

In WML, URLs are used in the following situations:

· When specifying navigation, e.g., hyperlinking.

· When specifying external resources, e.g., an image or a script.

**Fragment Anchors**

WML has adopted the HTML de facto standard of naming locations within a resource. A WML fragment anchor is specified by the document URL, followed by a hash mark (#), followed by a fragment identifier. WML uses fragment anchors to identify individual WML cards within a WML deck. If no fragment is specified, a URL names an entire deck. In some contexts, the deck URL also implicitly identifies the first card in a deck.

**Relative URLs**

WML has adopted the use of relative URLs, as specified in [RFC2396]. [RFC2396] specifies the method used to resolve relative URLs in the context of a WML deck. The base URL of a WML deck is the URL that identifies the deck.

Notice that WMLScripts are not embedded in WML pages. The WML pages only contains references to script URLs.

In the example below; if you select the go label the external script will direct you to http://www.KAHE.com/wap.wml:

```
<?xml version="1.0"?>
```

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="no1" title="Go to URL">
<do type="options" label="Go">
<go href="check.wmls#go_url('KAHE')"/>
</do>
</card>
</wml>
```

The red line above contains a reference to a WMLScript. The script is in a file called **check.wmls**, and the name of the function is **go_url**.

Here is the WML page called **check.wmls**:

```
extern function go_url(the_url)
{
if (the_url=="KAHE")
{
WMLBrowser.go("http://www.KAHE.com/wap.wml")
}
}
```

Note that the function is using the extern keyword. When using this keyword the function can be called by other functions or WML events outside the .wmls file. To keep a function private, drop the extern keyword.

WMLScript » URL » URL.getReferer

*Syntax:*

*URL.getReferer()*

Returns the smallest relative URL.

The URL.getReferer function returns the smallest relative URL for the resource (the referer) that called the current file. This returned URL is relative to the base URL of the current file. The empty string "" is returned if the current file does not have a referer. Note that local function calls cannot change the referrer.

This function does not take a parameter.

**Examples**

Code:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.WAPforum.org/DTD/wml_1.1.xml">
<wml>

<card id="card1">
   <p>
   getReferer example
   </p>
   <do type="accept">
      <go href="GetRefererExample.wmls#findgetreferer()" />
   </do>
</card>

<card id="card2">
<p>
   referer = $(refer)
</p>
</card>

</wml>
```

Explanation:

Code for GetRefererExample.wmls

Language(s): WML

Code:

```
extern function findgetreferer()
{
   var ref = URL.getReferer();
   WMLBrowser.setVar("refer", ref);
   WMLBrowser.go("GetRefererExample.wml#card2");
};
```

Explanation:

Code for GetRefererExample.wmls

Language(s): WML

## Markup Basics

A **markup language** is a **language** which is used to represent structured data.

In computer text processing, a **markup language** is a system for annotating a document in a way that is syntactically distinguishable from the text.[1] The idea and terminology evolved from the "marking up" of paper manuscripts (i.e., the revision instructions by editors), which is traditionally written with a red or blue pencil on authors' manuscripts.[2] In digital media, this

"blue pencil instruction text" was replaced by tags, which indicate what the parts of the document *are*, rather than details of how they might be shown on some display. This lets authors avoid formatting every instance of the same kind of thing redundantly (and possibly inconsistently). It also avoids the specification of fonts and dimensions which may not apply to many users (such as those with varying-size displays, impaired vision and screen-reading software).

Early markup system typically included typesetting instructions, as troff, TeX and LaTeX do, while Scribe and most modern markup systems name components, and later process those names to apply formatting or other processing, as in the case of XML.

Some markup languages, such as the widely used HTML, have pre-defined presentation semantics—meaning that their specification prescribes generally how to present the structured data on particular media. Others, such as XML and its predecessor SGML, allow but do not impose such prescriptions — all of the while allowing users to define any custom document components as they wish.

The concept of markup has been extended to include formatting instructions inserted into a file or document text so that software can format the text or a printer can print the text or document. Generally, these formatting instructions take the form of tags or commands that start with a special character.

The "granddaddy" of all computer markup languages is SGML (Standard Generalized Markup Language). SGML causes a bit of confusion for programmers. SGML is not a compiled or procedural computer programming language in the same way that BASIC, COBOL or SAS are programming languages. Rather, SGML is actually a standard for how to describe a document. But, with SGML, the way to describe a document also involves a method for marking up the document content. And the method for marking up a document, involves using the SGML standard to define the kind of tags or elements that you will routinely use to always mark up documents of a certain type. If this sounds confusing and complicated, that's because SGML is all about grammar, semantics and definition. SGML is an ISO standard (ISO 8879:1986), which means that it went through a period of evaluation, design and definition before being adopted by the International Standards Organization. Interestingly enough, the "official" title

of the SGML standard from the www.iso.org site is: "Information processing -- Text and office systems -- Standard Generalized Markup Language (SGML)".

What do markup files look like anyway? It depends. The one constant in markup files is that the information or content is usually readable. Sometimes, the formatting instructions that surround the text are easily decipherable, sometimes the formatting instructions seem opaque. Generally speaking, most markup files are designed to be used with an application, viewer or processor that understands what the formatting instructions mean and whether the formatting instructions are limited to a simple description of document structure or whether the formatting instructions also detail specific presentation characteristics. In the following screen shots, I'll show SGML, HTML, RTF, SYLK, CSV, troff, and LaTeX markup examples. Most of these files, were either created with Notepad or created with an appropriate tool and then viewed with Notepad or a text editor. Files like the SGML, troff or LaTex examples might be passed to a typesetting program or printer, 1 while the HTML file would be viewed with a web browser. An RTF (Rich Text Format) file would be normally viewed with a word processing program.

**HTML**

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages.

**Keyhole Markup Language**

Keyhole Markup Language (KML) is an XML notation for expressing geographic annotation and visualization within Internet-based, two-dimensional maps and three-dimensional Earth browsers. KML was developed for use with Google Earth, which was originally named Keyhole Earth Viewer. It was created by Keyhole, Inc, which was acquired by Google in 2004.

**Markdown**

Markdown is a lightweight markup language with plain text formatting syntax. Its design allows it to be converted to many output formats, but the original tool by the same name only supports HTML. Markdown is often used to format readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor.

**Mathematical Markup Language (MathML)**

Mathematical Markup Language (MathML) is a mathematical markup language, an application of XML for describing mathematical notations and capturing both its structure and content. It aims at integrating mathematical formulae into World Wide Web pages and other documents. It is part of HTML5 and an ISO standard ISO/IEC DIS 40314 since 2015.

**Scalable Vector Graphics (SVG)**

Scalable Vector Graphics (SVG) is an XML-based vector image format for two-dimensional graphics with support for interactivity and animation. The SVG specification is an open standard developed by the World Wide Web Consortium (W3C) since 1999.

SVG images and their behaviors are defined in XML text files. This means that they can be searched, indexed, scripted, and compressed. As XML files, SVG images can be created and edited with any text editor, as well as with drawing software.

All major modern web browsers—including Mozilla Firefox, Internet Explorer, Google Chrome, Opera, Safari, and Microsoft Edge—have SVG rendering support.

**XHTML**

**eXtensible HyperText Markup Language** (**XHTML**) is part of the family of XML markup languages. It mirrors or extends versions of the widely used HyperText Markup Language (HTML), the language in which Web pages are formulated.

While HTML, prior to HTML5, was defined as an application of Standard Generalized Markup Language (SGML), a flexible markup language framework, XHTML is an application of XML, a more restrictive subset of SGML. XHTML documents are well-formed and may therefore be parsed using standard XML parsers, unlike HTML, which requires a lenient HTML-specific parser.

XHTML 1.0 became a World Wide Web Consortium (W3C) recommendation on January 26, 2000. XHTML 1.1 became a W3C recommendation on May 31, 2001. The standard known as XHTML5 is being developed as an XML adaptation of the HTML5 specification.

The **Standard Generalized Markup Language** (**SGML**; ISO 8879:1986) is a standard for defining generalized markup languages for documents. ISO 8879 Annex A.1 defines generalized markup:-

Generalized markup is based on two postulates:

- Markup should be declarative: it should describe a document's structure and other attributes, rather than specify the processing to be performed on it. Declarative markup is less likely to conflict with unforeseen future processing needs and techniques.
- Markup should be rigorous so that the techniques available for processing rigorously-defined objects like programs and databases can be used for processing documents as well.

HTML was theoretically an example of an SGML-based language until HTML 5, which browsers cannot parse as SGML for compatibility reasons.

DocBook SGML and LinuxDoc are examples which were used almost exclusively with actual SGML tools.

**Extensible Markup Language** (**XML**) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The W3C's XML 1.0 Specification[2] and several other related specifications[3]—all of them free open standards—define XML.[4]

The design goals of XML emphasize simplicity, generality, and usability across the Internet.[5] It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures[6] such as those used in web services.

Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data.

## WML Basics

## WML Variables

Because multiple cards can be contained within one deck, some mechanism needs to be in place to hold data as the user traverses from card to card. This mechanism is provided via WML variables.

WML is case sensitive. No case folding is performed when parsing a WML deck. All enumerated attribute values are case sensitive. For example, the following attribute values are all different: id="Card1", id="card1", and id="CARD1".

Variables can be created and set using several different methods. Following are two examples:

The <setvar> element:

The <setvar> element is used as a result of the user executing some task. The >setvar> element can be used to set a variable's state within the following elements: <go>, <prev>, and <refresh>.

This element supports the following attributes:

| Attribute | Value | Description |
|---|---|---|
| name | string | Sets the name of the variable |
| value | string | Sets the value of the variable |
| class | class data | Sets a class name for the element. |
| id | element ID | A unique ID for the element. |

The following element would create a variable named *a* with a value of 1000:

```
<setvar name="a" value="1000"/>
```

The input elements:

Variables are also set through any input element like *input,select, option,* etc. A variable is automatically created that corresponds with the named attribute of an input element.

For example, the following element would create a variable named *b*:

```
<select name="b">
<option value="value1">Option 1</option>
<option value="value2">Option 2</option>
</select>
```

Using Variables:

Variable expansion occurs at runtime, in the microbrowser or emulator. This means it can be concatenated with or embedded in other text.

**WML Task**

A WML task is an element that specifies an action to be performed by the browser, rather than something to be displayed. For example, the action of changing to a new card is represented by a <go> task element, and the action of returning to the previous card visited is represented by a <prev> task element. Task elements encapsulate all the information required to perform the action.

WML provides following four elements to handle four WML tasks called go task, pre task, refresh task and noop taks.

The <go> Task:

As the name suggests, the <go> task represents the action of going to a new card.

The <go> element supports the following attributes:

| Attribute | Value | Description |
|---|---|---|
| href | URL | Gives the URL of the new card. Relative URLs are resolved relative to the current card |
| method | • get<br>• post | Specifies the method that should be used to fetch the deck. This must be one of the values get or post, corresponding to the GET and POST methods of HTTP.<br>When using **method="get"**, the data is sent as an request with ? data appended to the url. The method has a disadvantage, that it can be used only for a limited amount of data, and if you send sensitive information it will be displayed on the screen and saved in the web server's logs. So do not use this method if you are sending password etc. |

| | | |
|---|---|---|
| | | With **method="post"**, the data is sent as an request with the data sent in the body of the request. This method has no limit, and sensitive information is not visible in the URL |
| sendreferer | • true<br>• false | If set to true, the browser sends the URL of the current deck along with the request. This URL is sent as a relative URL if possible. The purpose of this is to allow servers to perform simple access control on decks, based on which decks are linking to them. For example, using HTTP, this attribute is sent in the HTTP Referer header. |
| accept-charset | charset_list | Specifies a comma- or space-separated list of character sets that can encode data sent to the server in a POST request. The default value is "unknown". |
| class | class data | Sets a class name for the element. |
| id | element ID | A unique ID for the element. |

Following is the example showing usage of <go> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="GO Element">
<p>
  <anchor>
    Chapter 2 : <go href="chapter2.wml"/>
  </anchor>
</p>
</card>
</wml>
```

Another example showing how to upload data using Get Method

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="GO Element">
<p>
  <anchor>
```

```
    Using Get Method
    <go href="chapter2.wml?x=17&y=42" method="get"/>
  </anchor>
</p>
</card>
</wml>
```

Another example showing how to upload data using <setvar> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="GO Element">
<p>
  <anchor>
    Using setvar:
          <go href="chapter2.wml">
            <setvar name="x" value="17"/>
            <setvar name="y" value="42"/>
          </go>
  </anchor>
</p>
</card>
</wml>
```

Another example showing how to upload data using <postfiled> element

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="GO Element">
<p>
  <anchor>
    Using setvar:
          <go href="chapter2.wml" method="get">
        <postfield name="x" value="17"/>
        <postfield name="y" value="42"/>
          </go>
  </anchor>
</p>
</card>
</wml>
```

**The <prev> Task:**

The <prev> task represents the action of returning to the previously visited card on the history stack. When this action is performed, the top entry is removed from the history stack, and that card is displayed again, after any <setvar> variable assignments in the <prev> task have taken effect.

If no previous URL exists, specifying <prev> has no effect.

The <prev> element supports the following attributes:

| Attribute | Value | Description |
|---|---|---|
| Class | class data | Sets a class name for the element. |
| Id | element ID | A unique ID for the element. |

Following is the example showing usage of <prev> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Prev Element">
<p>
  <anchor>
     Previous Page :<prev/>
  </anchor>
</p>
</card>
</wml>
```

One situation where it can be useful to include variables in a <prev> task is a login page, which prompts for a username and password. In some situations, you may want to clear out the password field when returning to the login card, forcing the user to reenter it. This can be done with a construct such as:

```
<?xml version="1.0"?>
```

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Prev Element">
<p>
  <anchor>
      <prev>
        <setvar name="password" value=""/>
      </prev>
  </anchor>
</p>
</card>
</wml>
```

**The <refresh> Task:**

The <refresh> task is the simplest task that actually does something. Its effect is simply to perform the variable assignments specified by its <setvar> elements, then redisplay the current card with the new values. The <go> and <prev> tasks perform the same action just before displaying the new card.

The <refresh> task is most often used to perform some sort of "reset" action on the card.

The <refresh> element supports the following attributes:

| Attribute | Value | Description |
|---|---|---|
| class | class data | Sets a class name for the element. |
| id | element ID | A unique ID for the element. |

Following is the example showing usage of <refresh> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>
```

```
<card title="Referesh Element">
<p>
  <anchor>
       Refresh this page:
        <go href="test.wml"/>
        <refresh>
          <setvar name="x" value="100"/>
        </refresh>
  </anchor>
</p>
</card>
</wml>
```

**The <noop> Task:**

The purpose of the <noop> task is to do nothing (no operation).

The only real use for this task is in connection with templates

The <noop> element supports the following attributes:

| Attribute | Value | Description |
|---|---|---|
| Class | class data | Sets a class name for the element. |
| Id | element ID | A unique ID for the element. |

Following is the example showing usage of <noop> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Noop Element">
<p>
  <do type="prev" label="Back">
     <noop/>
  </do>
</p>
</card>
</wml>
```

## WML Events

Event in ordinary language can be defined as something happened. In programming, **event** is identical in meaning, but with one major difference. When something happens in a computer system, the system itself has to **(1)** detect that something has happened and **(2)** know what to do about it.

WML language also supports events and you can specify an action to be taken whenever an event occurs. This action could be in terms of WMLScript or simply in terms of WML.

WML supports following four event types:

- onenterbackward: This event occurs when the user hits a card by normal backward navigational means. That is, user presses the Back key on a later card and arrives back at this card in the history stack.
- onenterforward: This event occurs when the user hits a card by normal forward navigational means.
- onpick: This is more like an attribute but it is being used like an event. This event occurs when an item of a selection list is selected or deselected.
- ontimer: This event is used to trigger an event after a given time period.

These event names are case sensitive and they must be lowercase.

### WML <onevent> Element:

The <onevent>...</onevent> tags are used to create event handlers. Its usage takes the following form:

```
<onevent type="event_type">
  A task to be performed.
</onevent>
```

You can use either *go, prev* or *refresh* task inside <onevent>...</onevent> tags against an event.

The <onevent> element supports the following attributes:

| Attribute | Value | Description |
|---|---|---|
| Type | • onenterbackward<br>• onenterforward<br>• onpick<br>• ontimer | Defines a type of event occured. |
| Class | class data | Sets a class name for the element. |
| Id | element ID | A unique ID for the element. |

Following is the example showing usage of <onevent> element. In this example, whenever you try to go back from second card to first card then **onenterbackward** occurs which moves you to card number three. Copy and paste this program and try to play with it.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>
<onevent type="onenterbackward">
  <go href="#card3"/>
</onevent>

<card id="card1" title="Card 1">
<p>
  <anchor>
     <go href="#card2"/>
     Go to card 2
  </anchor>
</p>
</card>
<card id="card2" title="Card 2">
<p>
   <anchor>
   <prev/>
      Going backwards
```

```
   </anchor>
</p>
</card>
<card id="card3" title="Card 3">
<p>
Hello World!
</p>
</card>
</wml>
```

The <setvar> element is used as a result of the user executing some task. The >setvar> element can be used to set a variable's state within the following elements: <go>, <prev>, and <refresh>.

## Attributes:

This element supports the following attributes:

| Attribute | Value | Description |
|---|---|---|
| name | string | Sets the name of the variable |
| value | string | Sets the value of the variable |
| class | class data | Sets a class name for the element. |
| id | element ID | A unique ID for the element. |

## Example:

The following element would create a variable named *a* with a value of 1000:

```
<setvar name="a" value="1000"/>
```

Following is another example showing usage of this element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>
<card>
<p>
 <anchor>
  Go to next chapter
```

```
  <go href="#chapter2">
   <setvar name="x" value="30"/>
  </go>
 </anchor>
</p>
</card>

</wml>
```

Setting a value to a variable using the <setvar/> tag is simple. The <setvar/> tag has two attributes, name and value. The name attribute defines the variable name and the value attribute defines the variable value. The following WML code assigns the value WML Tutorial to a variable called var1:

<setvar name="var1" value="WML Tutorial"/>

The <setvar/> tag should be enclosed in the following tags: <go></go>, <prev></prev>, <refresh></refresh>. For example, let's say you want to assign a value to a variable when a user follows an anchor link to a certain card, you should write something like this:

```
<anchor>
 <go href="tutorial_2.wml">
  <setvar name="last_tutorial" value="WML Tutorial"/>
 </go>
 Go to next tutorial
</anchor>
```

If you want to assign a value to a variable when a user clicks an anchor link but you do not want the WAP browser to leave the current card, you need to use the <refresh> tag instead of the <go> or <prev> tag. Here is a WML example that demonstrates how to use the <refresh> tag:

```
<anchor>
 <refresh>
  <setvar name="last_tutorial" value="WML Tutorial"/>
 </refresh>
Refresh variable
</anchor>
```

If you click the above anchor link, the value of the variable last_tutorial will be set to WML Tutorial. If the variable last_tutorial is shown in the current card, the WAP browser will refresh the card so that the updated value of last_tutorial will be displayed on the screen.

The <postfield> tag is used to post variables values to the server.

## Attributes:

This element supports the following attributes:

| Attribute | Value | Description |
|---|---|---|
| Name | string | Sets the name of the variable |
| value | string | Sets the value of the variable |
| class | class data | Sets a class name for the element. |
| id | element ID | A unique ID for the element. |

## Example:

Following example shows how to submit three fields *name, age* and *sex* to the server.

```
<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="card1" title="WML Form">
<p>
   Name: <input name="name" size="12"/>

   Sex : <select name="sex">
      <option value="male">Male</option>
      <option value="female">Female</option>
      </select>

   Age :  <input name="age" size="12" format="*N"/>
   <anchor>
      <go method="get" href="process.php">
          <postfield name="name" value="$(name)"/>
          <postfield name="age" value="$(age)"/>
          <postfield name="sex" value="$(sex)"/>
      </go>
```

```
      Submit Data
    </anchor>
</p>
</card>

</wml>
```

When you download above code on your WAP device, it will provide you option to enter three fields *name, age* and *sex* and one link *Submit Data*. You will enter three fields and then finally you will select *Submit Data* link to send entered data to the server.

## UNIT I

1. WAP stands for

**a) Wireless Application Protocol** b) Wired Application Protocol

c) Wide Application Protocol d) Web Application Protocol

2. The WAP protocol was designed to show internet contents on --------- clients?

a) wired **b) wireless**

c) web d) all of the above

3. WAP is designed for?

a) Internet Explorer b) Web browser

**c) Micro browser** d) Macro browser

4. The WAP protocol was mainly designed for _____.

a) Computers b) Networks

c) Pagers **d) Mobile phones**

5. Which language is used to create programs which are supported by WAP?

a) WAP **b) WML**

c) WASP d) WHTML

6. The WAP protocol was designed to show _____ contents on wireless clients, like mobile phones.

**a) Internet** b) network

c) Computer d) Mobile

7. WAP is _____ from Internet standards.

a) Built b) Developed

**c) Inherited** d) Linked

8. WAP homepages are not too different from the HTML homepages.

a) False b) Not always

**c) True** d) Noe of these

9. In ..... Frequency Spectrum is divided into smaller spectra and is allocated to each user.

a) TDMA b) CDMA

**c) FDMA** d) FGMA

10. In ..... multiple access is achieved by allocating different time slots for the different users.

**a) TDMA** b) CDMA

c) FDMA d) FGMA

11. Wireless communication is started in

a) 1869 b) 1895

c) 1879 **d) 1885**

12. The basic GSM is based on _____ traffic channels.

**a) Connection oriented** b) Connectionless

c) Packet switching d) Circuit switching

13. Which of the following is/are the main part(s) of basic cellular system.

a) A mobile unit b) A cell Site

c) A mobile telephone switching office **d) All of above**

14. The WAP gateway compiles the WML into _____ form.

**a) Binary** b) Hexa

c) Object d) Decimal

15. ---------- has been designed by the WAP Forum to provide fast connection suspension and reconnection.

**a) Wireless Session Protocol** b) Wireless Transaction Protocol

c) Wireless transport layer security d) Wireless Datagram Protocol

16. WAP micro browsers are used for _____.

**a) small handheld devices** b) large computer systems

c) Laptop d) Printers

17. To make your website mobile friendly, you can make your mobile _____.

a) Reactive b) Fast loading

**c) Responsive** d) Light

18. WAE stands for _____.

a) Wired Application Environment **b) Wireless Application Environment**

c) Web Application Environment d) Wide Application Environment.

19. WTA stands for _____.

**a) Wireless Telephony Application** b) Wireless Telegram Application

c) Wired Telephony Application d) Wired Telegram Application

20. Which organization specifies Wireless Application Protocol?

a) WAP Consortium **b) WAP Forum**

c) CCITT d) IEEE

21. What programming model is the WAP programming model similar to?

**a) WWW** b) OSI

c) Electromagnetic spectrum d) Object oriented

22. Which layer protocol is WTP?

a) Application b) Presentation

**c) Session** d) Transport

23. What are the benefits of WAP?

a) Easy access to time sensitive information b) Portable

c) Worldwide communication d) **All of above**

24. The WAP protocol was designed to show internet-contents on ----- clients?

a) Wired b) **Wireless**

c) Both d) None

25. Which layer protocol is WAE?

a) Transport b) Session

c) Presentation **d) Application**

26. Second generation of cellular phone network was developed, to provide higher-quality mobile _____.

a) Video Communications b) Signal generation

c) Frame communication d**) Voice Communication**

27. The purpose of a WAP gateway is

a) convert analog voice to digital format

b) to enable a portable device to obtain an IP address

c) **to convert HTML code into WML code**

d) to send data using WAP protocols

28. _____ are tailored to suit the requirements of mobile devices and networks.

a) WAP standards b) WML standards

c) WSP standards **e) WAE standards**

## UNIT II

29. URL stands for

a) Unique Resource locator **b) Uniform Resource Locator**

c) Universal Resource Locator d) User Resource Locator

30. WML supports _____.

**a) Client-side scripting** b) Server-side scripting

c) Both d) None of the above

31. The tag <b> defines ---------.

a) Big text **b) Bold text**

c) Black text d) none of the above.

32. The <go> task represents the action of _____.

a) discarding new card b) replacing card

**c) switching to a new card** e) deleting a new card

33. The <refresh> task refreshes _____.

a) specified text b) specified image

**c) specified card variables** d) specified controls

34. The onenterforward event is triggered when a user goes to a card in the _____ direction

a) Backward **b) Forward**

c) Left d) Right

35. In the early days, WAP-enabled wireless devices only supported the _____ image format.

**a) Wireless Bitmap** b) JPEG

c) Bitmap d) PNG

36. _____ helps to enhance the functionality of WAP application.

a) WML client **b) WML script**

c) WML server d) WML gateway

37. _____ initiate a request for connection to a server.
a) Gateway b) Network Channel
c) Network Event **d) Client**

38. WML is used to create_____ that can be displayed in a WAP browser

a) Documents b) Tables

**c) Pages** d) Files

39. _____ is used to specify the destination of a hyperlink.
**a) href** b) <a>
c) src d) <do>

40. _____ is the data repository, which could flow through other servers and reside Notes within one or more databases.
a) Data file **b) Data source**
c) Page file d) Page source

41. Which of these are not task elements?

a) go b) noop

c) refresh **d) do**

42. A deck contains a set of ?

a) pages **b) cards**

c) decks d) files

43. WML stands for?

a) Wired Markup Language **b) Wireless Markup language**

c) Wireless Main language d) Web Main Language

44. Pages in WML are called as

a) CARDS **b) DECKS**

c) MART d) WITE

45. What is WML Script?

a) Programming language **b) Scripting language**

c) System software d) Operating system

46. WML Script is used in ---------

a) ASP pages b) XML pages

**c) WML pages** d) HTML pages

47. WML pages can be displayed in a ---------- browser.

a) Web b) Mobile

**c) WAP** d) Wired

48. WML is inherited from

a) ASP b) DHTML

**c) HTML** d) PHP

49. WML script is a ------- version of the Java script language.

a) high **b) light**

c) strong d) medium

50. WML pages only contains references to script ----------.

**a) URLs** b) forms

c) pages d) links

51. WML Script is compiled into ---------- on the server before it is sent to WAP browser.

a) object code b) binary code

**c) byte code** d) Hexa code

52. WML script is a part of the ---------- specification.

a) W3C **b) WAP**

c) Java d) HTTP

53. What is the WML script supported file extension?

**a) .wml** b) .wap

c) .wmlscript d) .wsp

54. Which of these are not WML script dialogs library functions.

a) alert() b) confirm()

**c) error()** d) prompt()

55. The Float library contains a set of ------- functions.

a) numeric handling **b) math**

c) string d) character

UNIT III

56. _____ are used to obtain alphanumeric data from users.

a) Event field **b) Input fields**

c) selection c) Attribute field

57. _____ can be used in the place of strings and are substituted at run-time with their current value

**a) WML variables** b) WML element

c) WML markup d) WML controls

## UNIT III VARIABLES

### Setting Variable Values in WML

In WML, variables do not have to be declared explicitly. You can choose a variable name you like and assign a value to it directly. If you read a variable without assigning a value to it earlier, you will obtain an empty string.

Variable names in WML are case-sensitive. The first character of a variable name must be a letter or an underscore. The rest of the characters can be letters, numbers or underscores. Other characters, such as punctuations, are not permitted.

All variables are stored as string. They have a global scope, which means once you have set the value of a variable, you can read it in any cards and decks.

A variable is used to store some data. You can modify or read the value of a variable during execution. The *var* keyword is used to declare WMLScript variables. It should be used in the following form (the part enclosed within brackets [] is optional):

var *variable_name* [= *value_to_be_initialized*];

Below is an example. The following line of code declares a variable called *wmlscript_variable* and initializes its value to "Welcome to our WMLScript tutorial".

var wmlscript_variable = "Welcome to our WMLScript tutorial";

Variable initialization is optional. If you do not initialize a variable, the WMLScript interpreter will assign an empty string to it automatically, i.e. the following line of script:

var wmlscript_variable;

is equivalent to:

var wmlscript_variable = "";

WMLBrowser.setVar("message", "Hello World. Welcome to our WMLScript tutorial.");

WMLScript's *setVar()* function is used to set the value of a WML variable. It should be used in the following form:

WMLBrowser.setVar(*variable_name*, *value*);

*variable_name* is the name of the WML variable you want to assign a value to, and *value* is the value you want to assign to *variable_name*.

Using WMLScript to Obtain the Value of a WML Variable

WMLScript's *getVar()* function helps you obtain the value of a WML variable. It is used in the following form:

WMLBrowser.getVar(*variable_name*);

*variable_name* is the name of a WML variable.

The following are some of the WAP WML Control Statements:

**WML Script Control Statements**

The sequence and the iteration in a program are controlled by the Control statements such as:

| Statement | Description |
|---|---|
| if-else | Conditional branching |
| for | Making self-incremented fixed iteration loop |
| while | Making variable iteration loop |
| break | Terminates a loop |
| continue | Quit the current iteration of a loop |

**WML Script if...else Statement:**

The following syntax is used by the WML Scripts if statement. The portion inside the brackets is optional. The syntax will be the same for C++, Java and JavaScript.

if (condition)

{

WMLScript statement(s)

}

[else

{

WMLScript statement(s)

}]

If condition is the Boolean value true, the statement(s) enclosed in the first curly brackets {} will be executed; if condition is false or invalid, the statement(s) enclosed in the second curly brackets {} will be executed.

**WML Script while Statement**

In order to repeat the execution of a block of statements when a condition is true, the while statement is used. The syntax is as follows:

while (condition)

{

WMLScript statement(s)

}

The statement(s) enclosed in the curly brackets {} will be executed again and again as long as condition is true. The loop stops when condition evaluates to false or invalid.

**WML Script for Statement**

The for statement is used for repeatedly execute the function as long a the condition is satisfied. In case if the number of repetition times is known, then for statement would be more convenient than the while statement. The syntax for the for statement in WML Script is as follows:

for ([expression1]; [expression2]; [expression3])

{

WMLScript statement(s)

}

expression1 is the initialization expression. It is executed before any WMLScript statements in the for loop are executed. expression1 is executed once only in the lifetime of the for statement. expression2 is the conditional expression that determines whether the for loop should continue or stop. expression3 is executed after each iteration.

**WML Script break Statement**

In order to quit a loop, the break statement is used. They are put inside the while loops or for loops. The lustration to demonstrate the WML Script for using the break statement is as follows:

```
var result = 0;
for (var counter=0; counter<10; counter++)
{
break;
result += 5;
}
```

After the execution the above WMLScript code, the value of result is 0. This is because the break statement exits the for loop. The statement "result += 5;" is never executed.

**WML Script continue Statement**

In order to quit the current iteration of the loop in the WML Script, the continue statement is used. If the conditional expression of the loop evaluates to be true, the next iteration will be started. This statement is put inside the while loops or for loop. An illustration demonstrating the use of continues statement is as follows:

```
var result1 = 0;
var result2 = 0;
for (var counter=0; counter<10; counter++)
```

```
{
result1 += 5;
continue;
result2 += 5;
}
```

On execution of the code, the value of result1 would by 20 and the value of result2 is 0. The current iteration ends when the WML Script interpreter encounters continue statement. Hence, the statement "result2 += 5;" is never executed.

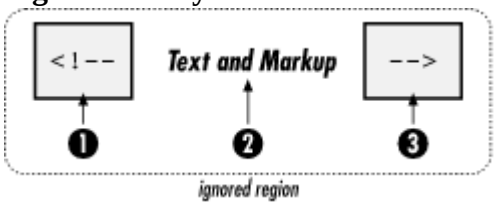
## MISCELLANEOUS MARKUP

Elements, attributes, namespaces, and entities are the most important markup objects, but they are not the end of the story. Other markup objects including comments, processing instructions, and CDATA sections shield content from the parser in various ways, allowing you to include specialized information.

### 2.6.1. Comments

Comments are notes in the document that are not interpreted by the parser. If you're working with other people on the same files, these messages can be invaluable. They can be used to identify the purpose of files and sections to help navigate a cluttered document, or simply to communicate with each other. So, in XML there is a special kind of markup called a *comment*. The syntax for comments is shown in Figure 2.17.

**Figure 2.17. Syntax for comments**




A comment starts with four characters: an open angle bracket, an exclamation point, and two dashes (1). It ends with two dashes and a closing angle bracket (3). In between these delimiters goes the content to be ignored (2). The comment can contain almost any kind of text you want, including spaces, newlines, and markup. However, since two dashes in a row (--) are used tell the parser when a comment begins and ends, they can't be placed anywhere inside the comment. This means that instead of using dashes to create an easily visible line, you should use another symbol like an equals sign (=) or an underscore (_):

Good: <!--==========-->

Good: <!--_____-->

Good: <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - -->

Bad: <!---------------------------------------------------------->

Bad: <!-- -- Don't do this! -- -->

Comments can go anywhere in your document except before the XML declaration and inside tags; an XML parser will ignore those completely. So this piece of XML:

The quick brown fox jumped<!-- test -->over the lazy dog.
The quick brown <!-- test --> fox jumped over the lazy dog. The<!--

test

-->quick brown fox
jumped over the lazy dog.

becomes this, after the parser has removed the comments:

The quick brown fox jumpedover the lazy dog.
The quick brown fox jumped over the lazy dog. Thequick brown fox
jumped over the lazy dog.

Since comments can contain markup, they can be used to "turn off" parts of a document. This is valuable when you want to remove a section temporarily, keeping it in the file for later use. In this example, a region of code is commented out:

Our store is located at:
<!--
<address>59 Sunspot Avenue</address>
-->
<address>210 Blather Street</address>

When using this technique, be careful not to comment out any comments, i.e., don't put comments inside comments. Since they contain double dashes in their delimiters, the parser will complain when it gets to the inner comment.

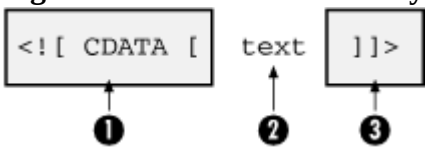
*2.6.2. CDATA Sections*

If you mark up characters frequently in your text, you may find it tedious to use the predefined entities <, >, &. They require typing and are generally hard to read in the markup. There's another way to type lots of forbidden characters, however: the CDATA section.

CDATA is an acronym for "character data," which just means "not markup." Essentially, you're telling the parser that this section of the document contains no markup and should be treated as regular text. The only thing that cannot go inside a CDATA section is the ending delimiter (]]>). For that, you have to resort to a predefined entity and write it as ]]>.

The CDATA section syntax is shown in Figure 2.18. A CDATA section begins with the nine-character delimiter <![CDATA[ (1), and it ends with the delimiter ]]> (3). The content of the section (2) may contain markup characters (<, >, and &) but they are ignored by the XML processor.

**Figure 2.18. CDATA section syntax**




Here's an example of a CDATA section in action:

<para>Then you can say <![CDATA[if (&x < &y)]]> and be done
with it.</para>

CDATA sections are most convenient when used over large areas, say the size of a small computer program. If you use it a lot for small pieces of text, your document will become hard to read, so you'd be better off using entity references.

*2.6.3. Processing Instructions*

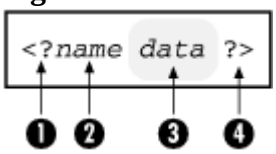
Presentational information should be kept out of a document whenever possible. Still, there may be times when you don't have any other option, for example, if you need to store page numbers in the document to facilitate generation of an index. This information applies only to a specific XML processor and may be irrelevant or misleading to others. The prescription for this kind of information is a *processing instruction*. It is a container for data that is targeted toward a specific XML processor.

Processing instructions (PIs) contain two pieces of information: a target keyword and some data. The parser passes processing instructions up to the next level of processing. If the processing instruction handler recognizes the target keyword, it may choose to use the data; otherwise, the data is discarded. How the data will help processing is up to the developer.

Figure 2.19 shows the PI syntax. A PI starts with a two-character delimiter (1) consisting of an open angle bracket and a question mark (<?), followed by a *target* (2), an optional string

of characters that is the data portion of the PI (3), and a closing delimiter (4), consisting of a question mark and closing angle bracket (?>).

**Figure 2.19. Processing instruction syntax**




"Funny," you say, "PIs look a lot like the XML declaration." You're right: the XML declaration can be thought of as a processing instruction for all XML processors[4] that broadcast general information about the document.

[4] This syntactic trick allows XML documents to be processed easily by older SGML systems; they simply treat the XML declaration as another processing instruction, ignored except by XML processors.

The target is a keyword that an XML processor uses to determine whether the data is meant for it or not. The keyword doesn't necessarily mean anything, such as the name of the software that will use it. More than one program can use a PI, and a single program can accept multiple PIs. It's sort of like posting a message on a wall saying, "The party has moved to the green house," and people interested in the party will follow the instructions, while those uninterested won't.

The PI can contain any data except the combination ?>, which would be interpreted as the closing delimiter. Here are some examples of valid PIs:

<?flubber pg=9 recto?>
<?thingie?>
<?xyz stop: the presses?>

If there is no data string, the target keyword itself can function as the data. A forced line break is a good example. Imagine that there is a long section heading that extends off the page. Rather than relying on an automatic formatter to break the title just anywhere, we want to force it to break in a specific place.

Here is what a forced line break would look like:

<title>The Confabulation of Branklefitzers <?lb?>in a Portlebunky
Frammins <?lb?>Without Denaculization of <?lb?>Crunky Grabblefooties
</title>

**SENDING INFORMATION**

In the earlier sections of this WML tutorial, we have mentioned about how to use input fields and selection lists to create forms and obtain data from a user. In many situations, you need to submit the form data to the server for further processing. To submit data to the server in WML, you need the <go></go> and <postfield/> tags. The <postfield/> tag should be enclosed in the <go></go> tag pair. Let's first have a look at the following WML example before we go into the details:

(sendDataEg1.wml)

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org/DTD/wml13.dtd">

<wml>
  <card id="card1" title="WML Form">
    <onevent type="onenterforward">
      <refresh>
        <setvar name="my_temp_id" value="123456"/>
      </refresh>
    </onevent>

    <p>
      Hello, welcome to our WML tutorial.<br/>

      What's your name?<br/>
      <input name="myName"/><br/>

      Are you a boy or a girl?<br/>
      <select name="myGender">
        <option value="Boy">I am a boy</option>
        <option value="Girl">I am a girl</option>
      </select><br/>

      Which part of our WML tutorial do you like?<br/>
      <select name="favorite_tutorial_part" multiple="true">
        <option value="Part 1">Part 1</option>
        <option value="Part 2">Part 2</option>
        <option value="Part 3">Part 3</option>
        <option value="Part 4">Part 4</option>
      </select><br/><br/>
```

```
    <anchor>
      <go                    method="get"                    href="processing.asp">
        <postfield              name="name"                  value="$(myName)"/>
        <postfield             name="gender"                value="$(myGender)"/>
        <postfield      name="tutorial_part"      value="$(favorite_tutorial_part)"/>
        <postfield              name="temp_id"               value="$(my_temp_id)"/>
      </go>
      Submit                                                                Data
    </anchor>
   </p>
  </card>
</wml>
```

Many times, you will want your users to submit some data to your server. Similar to *HTML Form* WML also provide a mechanism to submit user data to web server.

To submit data to the server in WML, you need the <go>...</go> along with <postfield/> tags. The <postfield/> tag should be enclosed in the <go>...</go> tag pair.

To submit data to a server, we collect all the set WML variables and use <postfield> elements to send them to the server. The <go>...</go> elements are used to set posting method to either POST or GET and to specify a server side script to handle uploaded data.

In previous chapters we have explained various ways of taking inputs form the users. These input elements sets WML variables to the entered values. We also know how to take values from WML variables. So now following example shows how to submit three fields *name*, *age* and *sex* to the server.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card id="card1" title="WML Form">
<p>
  Name: <input name="name" size="12"/>
  Sex : <select name="sex">
    <option value="male">Male</option>
```

```
    <option value="female">Female</option>
    </select>
  Age :  <input name="age" size="12" format="*N"/>
  <anchor>
    <go method="get" href="process.php">
      <postfield name="name" value="$(name)"/>
      <postfield name="age" value="$(age)"/>
      <postfield name="sex" value="$(sex)"/>
    </go>
    Submit Data
  </anchor>
</p>
</card>
</wml>
```

When you download above code on your WAP device, it will provide you option to enter three fields *name, age* and *sex* and one link *Submit Data*. You will enter three fields and then finally you will select *Submit Data* link to send entered data to the server.

The *method* attribute of the <go> tag specifies which HTTP method should be used to send the form data.

If the HTTP POST method is used, the form data to be sent will be placed in the message body of the request. If the HTTP GET method is used, the form data to be sent will be appended to the URL. Since a URL can only contain a limited number of characters, the GET method has the disadvantage that there is a size limit for the data to be sent. If the user data contains non-ASCII characters, you should make use of the POST method to avoid encoding problems.

There is one major difference between HTML and WML. In HTML, the name attribute of the <input> and <select> tags is used to specify the name of the parameter to be sent, while in WML the name attribute of the <postfield> tag is used to do the same thing. In WML, the

name attribute of <input> and <select> is used to specify the name of the variable for storing the form data.

The <postfield> tag contains information to be sent to the server along with a <go> tag.

**Syntax**

```
<postfield name="somename" value="somevalue"/>
```

**Attributes**

| Attribute | Value | Description |
|---|---|---|
| name | *cdata* | REQUIRED. The name of the field |
| Value | *cdata* | REQUIRED. The value of the field |
| Class | *class_name* | Sets a class name for the element. The class name is case sensitive. An element can be connected to multiple classes. Multiple class names within the class attribute are separated by white space |
| Id | *id_name* | Sets a unique name for the element |

**Other data, Meta data**

You can specify some generic metadata in your WML file using the <meta/> tag. Metadata is placed at the document head. The <head> and </head> tags mark the start and end of the document head respectively. A WAP browser will ignore the metadata if it does not understand the metadata's meaning. You can specify metadata of any sort in a WML file without affecting the cards' look. For example, you may want to state the author name in your WML file without displaying it on the screen. The following example illustrates how to do this:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org/DTD/wml13.dtd">

<wml>
  <head>
    <meta name="author" content="Andrew"/>
  </head>
```

```
<card id="card1" title="WML Tutorial">
  <p>Hello World</p>
</card>
```

**Errors and browser limitations**

Encoding Semantics

**Document Tokenisation**

The process of tokenising an XML document must convert all markup and XML syntax (ie, entities, tags, attributes, etc.) into their corresponding tokenised format. All comments must be removed. Processing directives intended for the tokeniser may be removed. Other meta-information, such as the document type definition and unnecessary conditional sections must be removed. All text and character entities must be converted to string (eg, STR_I) or entity (ENTITY) tokens. Character entities in the textual markup (eg, &) must be converted to string form when tokenised, if the target character encoding can represent the entity. Characters present in the textual form may be encoded using the ENTITY token when they cannot be represented in the target character encoding. XML parsed entities (both internal and external) must be resolved before tokenisation. XML notations and unparsed entities are resolved on an application basis (eg, using inline opaque data). Attribute names must be converted to an attribute start token or must be represented by a single LITERAL token. Attribute values may not be represented by a LITERAL token.

It is illegal to encode markup constructs as strings. The user agent must treat all text tokens (eg, STR_I and ENTITY) as CDATA, ie, text with no embedded markup.

The XML white space handling semantics must be preserved during the tokenisation process. Insignificant white space, as defined by [XML], may be altered or removed by the WBXML tokenisation process. Significant white space, as defined by [XML], must be preserved. This includes white space in elements and attribute values.

**Document Structure Conformance**

The tokenised XML document must accurately represent the logical structure, as defined by [XML], and semantics of the textual source document. This implies that the source document must be well-formed, as defined in [XML]. Document tokenisation may validate the document as specified in [XML], but this is not required. If the semantics of a particular DOCTYPE are well known, additional semantic checks may be applied during the tokenisation process.

**Encoding Default Attribute Values**

The tokenised representation of a XML document may omit any attributes that are implied in the DTD or are specified with their default value. This implies that a user agent implementation must be aware of the attribute defaults of a given version of the DTD. This information can be inferred from the version number in the tokenised data format.

Encoding Examples

The following example encodings are for demonstration purposes only, and do not necessarily represent an optimal WBXML encoding.

**A Simple XML Document**

The following is an example of a simple tokenised XML document. It demonstrates basic element, string and entity encoding. Source document:

```
<?xml version="1.0"?>
<!DOCTYPE XYZ [
<!ELEMENT XYZ (CARD)+>
<!ELEMENT CARD (#PCDATA | BR)*>
<!ELEMENT BR EMPTY>
<!ENTITY nbsp " ">
]>
<XYZ>
<CARD>
X &amp; Y<BR/>
X = 1
</CARD>
</XYZ>
```

The following tokens are defined for the tag code space:

**Tag Name Token**

BR 5

CARD 6

XYZ 7

Tokenised form (numbers in hexadecimal) follows. This example uses only inline strings and assumes that the character encoding uses a NULL terminated string format. It also assumes that the transport character encoding is US-ASCII. This encoding is incapable of supporting some of the characters in the deck (eg,  ), forcing the use of the ENTITY token.

```
01 01 03 00 47 46 03 ' ' 'X' ' ' '&' ' ' 'Y' 00 05 03 ' '
'X' 00 02 81 20 03 '=' 00 02 81 20 03 '1' ' ' 00 01 01
```

**Tag Tokens**

**The following token codes represent tags in code page zero (0). All numbers are in hexadecimal.**

**Table 5. Tag tokens**

| Tag Name | Token | | |
|---|---|---|---|
| **a** | **1C** | | |
| **anchor** | **22** | **go** | **2B** |
| **access** | **23** | **head** | **2C** |
| **b** | **24** | **i** | **2D** |
| **big** | **25** | **img** | **2E** |
| **br** | **26** | **input** | **2F** |
| **card** | **27** | **meta** | **30** |
| **do** | **28** | **noop** | **31** |
| **em** | **29** | **p** | **20** |
| **fieldset** | **2A** | **postfield** | **21** |

### 14.2.3 Document Validation

XML document validation (see [XML]) should occur during the process of tokenising a WML deck and must be based

on the DOCTYPE declared in the WML deck. When validating the source text, the tokenisation process must accept

any DOCTYPE or public identifier, if the document is identified as a WML media type (see section 13.1.2).

The tokenisation process should notify the user of any well-formedness or validity errors detected in the source deck.

#### 14.2.3.1 Validate %length;

The WML tokenisation process should validate that attribute values defined as %length; contain either a NMTOKEN

or a NMTOKEN followed by a percentage sign character. For example, the following attributes are legal:

vspace="100%"

hspace="123"

%length; data is encoded using normal attribute value encoding methods.

**14.2.3.2 Validate %vdata;**

The WML tokenisation process must validate the syntax of all variable references within attribute values defined as

%vdata; contain variables and that other CDATA attribute values do not. Attribute values not defined in the DTD

must be treated as %vdata; and validated accordingly.

**WML version negotiation**

The WAP standards have been through a number of revisions. WML 1.1 is part of WAP1.1 (released in June 1999) and is the language baseline. WML 1.2, part of WAP 1.2(released in November 1999), is mostly an information release providing corrections and clarifications to the version 1.1 specification, although it did introduce a few new features. Few implementations are likely to support WAP (and WML) version 1.2. Instead, most will support the subsequent release, tentatively titled WAP 1.2.1. However the WAP Forum is moving to a date-based identification scheme, so the release name will probably be similar to WAP June 2000. Regardless of how the suite is named, the corresponding version of WML is numbered 1.3. Where differences or ambiguities are known to exist between versions, we highlight them. Changes in WML 1.3 are described based on the proposed draft standard.

**History:**

The WAP standard was introduced by the WAP Forum, created in 1989 by the leading players in the mobile industry, with the main goal of creating a standardized protocol for data transmission using various wireless technologies. The first WAP site was launched in October 1999 by the Dutch mobile operator Telfort BV.

In 1998, the WAP Forum also published the WML 1.1 standard, based on Openwave's HDML, Nokia's Tagged Text Markup Language (TTML) and Ericsson's proprietary markup language for mobile content. WML 2.0 was specified in 2001, but was never widely accepted, as it was taken over by more advanced technologies.

The WAP Forum was consolidated into Open Mobile Alliance (OMA) in 2002, along with many other mobile industry forums. In the time of its introduction, WAP was marketed very aggressively, which led users to expect WAP performance to be on par

with non-mobile, fixed internet access. When the first WAP-enabled mobile devices appeared in 1999, they fell short of users' expectations, which was to be expected given their limited hardware resources.
By 2003, WAP made a stronger appearance in Europe, for the introduction of additional wireless services like T-Mobile or Vodafone Live! Operators generated revenue on the GPRS and UMTS data transfer, and WAP traffic in the UK doubled from 2003 to 2004.

WAP was widely adopted in Japan, while in the USA, WAP never really took off. WAP was never fully embraced by major markets such as the US, China, and a lot of other nations, particularly in Asia.

Since then, WAP has largely gone out of use in Europe, as modern mobile devices enabled full support for HTML and CSS. Multimedia Messaging Service (MMS) still remains in use, as a combination of WAP and SMS technologies.

## UNIT IV

### Making Wireless Application Easy to use:

The Wireless Application Protocol (WAP) is a worldwide standard for the delivery and presentation of wireless information to mobile phones and other wireless devices. The idea behind WAP is simple: simplify the delivery of Internet content to wireless devices by delivering a comprehensive, Internet-based, wireless specification. The WAP Forum released the first version of WAP in 1998. The WAP Forum was founded in 1997 by Ericsson, Motorola, Nokia, and Openwave Systems with the goal of making wireless Internet applications more mainstream by delivering a development specification and framework to accelerate the delivery of wireless applications.

The WAP specification is continually changing to meet the growing demands of wireless applications. The majority of wireless carriers and handset manufacturers support WAP and continue to invest in the new capabilities it offers. Over the years WAP has evolved from using proprietary protocols in WAP 1.x to using standard Internet protocols in WAP 2.x, making it more approachable for Web developers. The following are some of the key benefits that WAP provides:

- WAP supports legacy WAP 1.x protocols that encode and optimize content for low-bandwidth, high-latency networks while communicating with the enterprise servers using HTTP.
- WAP supports wireless profiles of Internet protocols for interoperability with Internet applications. This allows WAP clients to communicate with enterprise servers, without requiring a WAP gateway.
- WAP allows end users to access a broad range of content over multiple wireless networks using a common user interface, the WAP browser. Because the WAP specification defines the markup language and microbrowser, users can be assured that wireless content will be suitable for their WAP-enabled device.
- WAP uses XML as the base language for both WML and WML2 (which uses XHTML), making it easy for application developers to learn and build wireless Internet applications. It also makes content transformation easier by incorporating

support for XSL stylesheets to transform XML content. Once an application is developed using WML or WML2, any device that is WAP-compliant can access it.

- WAP has support for WTA. This allows applications to communicate with the device and network telephony functions. This permits the development of truly integrated voice and data applications.
- Using UAProf, the information delivered to each device can be highly customized. (Chapter 13 provides more details on how this information can be used to deliver user-specific content.)
- WAP works with all of the main wireless bearers, including CDPD, GSM, CDMA, TDMA, FLEX, and iDEN protocols. This interoperability allows developers to focus on creating their applications, without having to worry about the underlying network that will be used.

At present, all major wireless carriers support the WAP specification. This universal support is expected to continue as WAP evolves, providing a robust, intuitive way to extend Web content to wireless devices.

## WEBSITE DESIGN

If you are familiar to designing pages for the Web, you will need to adjust (and reduce) your thinking for handheld devices. Due to the limitations inscreen size, processing power, and connection speed, normal information-rich web pages are not feasible. WAP applications are typically made up of screenfuls of minimal text and lists of options. Information and interaction design is king as there is virtually no graphic design to be done.Another difference is that unlike the Web, a successful WAP application can be measured by how quickly the user can find information or make atransaction and get out.

## WAPBrowsers

Mobile devices use special browsing software, sometimes referred to as"microbrowsers" due to their size and capacity, to request and displayinformation from a network. As on the Web,

not all WAP browsers are created equal. Older devices withoutdated browsers are still in circulation, and what works on one device may not work on another at all.

Most WAP-enabled devices use the recently renamed Openwave MobileBrowser (formerly called UP.Browser) developed by Openwave (at one timecalled Unwired Planet, thus "UP").

Not surprisingly, Microsoft has thrown its hat into the wireless ring with itsbrowser, Microsoft Mobile Explorer for cell phones. MME is a dual-modebrowser, supporting both WML and HTML content.

***ScreenSize***

Screen size varies from one device to another. To give you a ballpark idea,many phone display areas are 95 to 120 pixels wide and 50 to 65 pixels high. Newer phones and PDAs may have larger screens (approximately 300 100pixels).

Screen resolution is difficult to pin down. To make matters more interesting, the Nokia 7110 phone has pixels that are taller than they are wide (by a ratio of 1.25:1), which can stretch out graphical images.

***Text***

Because WAP content is primarily text-based, it may be more meaningful to measure screen space in terms of number of characters displayed. In general, mobile browsers can display only 3 to 6 lines of text at a time with 12 to 20 characters per line.

Text may be displayed as either monospace, where all characters are the same width, or proportional, where characters are varying widths. This can make it difficult to anticipate how many characters will fall on a line.

***Colordepth***

The vast majority of mobile devices have 1-bit black and white LCD displays.

***Softkeys***

Handheld devices typically feature *softkeys*, buttons that can be programmed for application-specific functionality. How many keys are available, where they are placed, and how they get assigned to actions in the code varies from device to device. Some softkeys are just rendered

graphically in the display area. This makes it difficult to anticipate how users will access and navigate through an application.

**SizeLimitations**

Each *.wml* document (called a "deck") within a WAP application has a maximum permitted size of 1400 bytes, although most web developers aim to keep the file size below 500 bytes to improve performance. This size limit refers to *compiled* decks. If your deck is larger than 1400 bytes, you must split it logically into separate files.

***WAPEmulators***

WAP application developers generally use WAP emulators to test theirdocuments. An emulator (or "simulator") is a program that runs on yourcomputer that shows you how your document will appear and function on awireless device.

**Computer vs Mobile Terminals**

| Terminals | Computer | Mobile |
|---|---|---|
| Power | If you want high performance, the CPU will use a lot of electricity. | Smartphones and tablets run on batteries, which means they have to use power as efficiently as possible. |
| Cost | Prices for a computer can vary widely depending on its capabilities, storage, and features. | The price of a smartphone can be financed month-to-month along with a service plan from providers. |
| Screen sizes | Desktop computers can connect to a wide variety of monitors. Most modern computer monitors are LCDs. | Smartphone screens vary somewhat depending on the manufacturer and model. |
| Portability | While laptop computers are usually light and portable by design, they can't compete with smartphones in this regard. | Smartphones are lightweight and can fit in a purse or pocket. They are designed to be taken everywhere with you. |
| Storage | Consumer hard drive capacities already exceed many TB (terabytes) and they continue to increase in size. | Most smartphones and tablets come with between 16 GB and 128 GB of built-in storage. |

| | | |
|---|---|---|
| Operating System | Operating systems designed for computers and laptops are full-featured. They are designed to take advantage of fast CPUs, large amounts of disk space, and high amounts of RAM | Mobile operating systems (Android and iOS) are specialized for a specific set of devices. |
| Softwares | Desktop and laptop computers can run more powerful software than a smartphone or tablet due to their size, components, and less restrictive power requirements. | While smartphones and tablets are getting more powerful every year, they still can't match a traditional laptop or desktop computer in terms of performance. |
| Internet Connectivity | Desktop computers generally come with Ethernet capability, either on the motherboard or a discrete NIC. Most desktop computers require a peripheral device, such as a USB Wi-Fi adapter, to connect to wireless networks. | Smartphones and tablets can connect to Wi-Fi networks for Internet access. Also, smartphones can connect to a mobile data network; many tablets offer mobile data connectivity as well. |

Web / WAP protocol layer comparison

| | Web | WAP |
|---|---|---|
| Markup Language | HTML | WML |
| Scripting Languages | VBScript<br>JavaScript | WMLScript |
| Session and Transaction Layers | HTTP | WSP - Wireless Session Layer<br>WTP - Wireless Transaction Layer |
| Security | TLS-SSL | WTLS - Wireless Transport Layer Security |
| Delivery Service | TCP, UDP<br><br>IP | WDP - Wireless Datagram Protocol<br><br>Bearer |

**Designing a usable WAP site**

Due to the fact that WAP sites use primitive monochrome displays a thorough analysis of usability issues is required when considering developing a WAP site as mistakes in the design of the WAP site could make it totally unusable.Usability is concerned with determining the effects of interface design on the user with the goal of creating a usable design.

In developing the custom framework, background usability research into WAP design was carried out and a number of design methodologies were also looked at.

**1. Audience analysis**

The end users of WAP site is expected to have a good knowledge of how to access the WAP. In order to ensure that the WAP site is appropriate for its designated end users individuals matching the end user criteria will be used at the testing and feedback stage. It is important to find test candidates who are representative of your intended audience; the creation of user personas can assist with this.

**2. Mobile platform analysis**

The final product has been designed to run on WAP enabled mobile phones that conform to the WAP specification as defined by The WAP Forum. Platforms that will support our WAP site include:

• Nokia 7110 • Nokia 62210 • Motorola Timeport • Ericsson R380s • Ericsson R320 • Siemens c35 • Siemens m35 • Siemens s35 • Motorola a6188 • Motorola p73 89

**3.Development platform analysis**

In order to emulate accurately the process of an inexperienced user building a WAP, an off-the-shelf WAP development package is selected rather than coding the site by hand. There are a number of WAP sitedevelopment packages that could be used to develop the site. In order to decide which package would best meet the usability requirements, weighted rankings by levels table is used.

**Structured Usability Method**

To do this six areas were chosen that is felt as most important when using a development package. These six areas were:

1. Well structured GUI (Graphical User Interface). 2. Availability of a preview feature. 3. Design features of development package. 4. Extra features of development package. 5. Built-in WML Code validator. 6. Help system and availability of resources.

**1.Wellstructured GUI**

A well-structured development site GUI is important in the development of a WAP site. A poorly structured GUI would increase the time taken to use and learn the package and this in turn would increase the development time of the WAP site. Time management is essential when developing a piece of software thus poorly designed development packages must be avoided.

**2. Availability of a preview feature**

The availability of a preview feature is vital when developing a WAP site. A real time preview window allows the developer to see what they are building as they progress. Development packages that make you save changes and load a card up each and every time you wish to view your progress can cost valuable time.

**3.Design features of the development package**

The development package would invariably require a number of design features in order to build a high quality WAP site. Design features would ideally include:

1. Facility for creation of Wireless Bitmap Images (WBMPs). 2. Card title/Id display. 3. Facility to add an extra card to the WAP deck. 4. Facility to implement WBMP alt tag (will be shown when WAP device does not support WBMP images).

**4. Extra features of the development package**

Extra features of a WAP site development package that would make it more desirable would include:

1. Facility to publish WAP site directly from the development package to a web server (integrated FTP client). 2. WBMP image editor/converter.

**5. Built-in WML code validator**

A built-in WML code validator is of great importance when considering a WAP site development package. Using a built-in code validator allows you to check your code incrementally as you build the WAP site, this allows errors to be identified and corrected immediately, as opposed to coding the site 'blind' and then coming across errors when you attempt to access the WAP site.

**User Interface design Guidelines**

**1. Design according to the platform**

The platform is as important as other criteria for design and development of the product (Mobile app). Whether it is Android or iOS and cross-platform, designers must apply guidelines according to particular platform chosen by concern authority of the project development.

**2. Maintain the design flow as per the theme**

A theme from beginning to end of the mobile app flow must hold the charm of it. The line icons, fill icons, and color combinations must keep accurately aligned based on the selected theme. The user attention keeps continuing to move one to another page of the app.

**3. Less typing using auto-complete data**

Filling up a form using mobile is the boring process to do so. Hence, the best thing is to minimize the quantity of typing required to use a mobile app. Only required fields should be determined else remove the unnecessary fields from the form. Use auto-complete and personalized data where relevant so that users only have to bare least of information.

**4. Make interface elements clearly visible**

The elements depicted as front-end of the mobile app is just simply convincible when a user lands on the home screen. The app heading title must have 30pxl. If the font alignment is settled left, and center then the consistency of the aligned content must maintain up to the last screen.

Select basic colors for your app that support usability. You must assure that used color contrast between elements so that users with low vision can see and use your app. A content must set by the use of color and contrast to help users.

**5. Content must readable to user**

By comparing with desktops, the smartphones have relatively small screens. It means that one of the most challenges of mobile design is to fit a lot of information on a small User Interface.

**6. Product must highlight while representing a portfolio**

As a company wants to launch a product quickly with a product portfolio and suffering from limited time, resources then you have to make a fast decision. The designers must focus on design and optimization on the targeted product which in the portfolio.

**7. Define UI brand signatures**

The interaction with the app should connect with the story of the brand and must increase recognition, loyalty, and satisfaction. Features, visuals, wording, fonts, and animations could be considered as examples.

**8. Optimize UI flows and elements**

Users can better feel when they get an easier flow of the app that only depends on the designer who optimizes the app precisely. Optimizing each screen flow and UI elements will reduce waiting time for users.

**9. Make navigation self-apparent**

Mobile navigation must be unified. Designers should use proper visual metaphors so that the navigation doesn't require any explanation.

**Design Guidelines for Selected WML Elements**

• These design guidelines focus on the user interface, they depend heavily on the browsers • Design guidelines for selected WML elements and the user interface • As an application developer, you have a few options 1. Design your application with a full understanding of browser differences 2. You can base the majority of your application on common elements 3. You can tailor your application for each device or browser These guidelines are inspired by a document submitted to the WAP forum entitled "generic Content Authoring Guidelines for WML 1.1 The test cases were constructed and run across three simulators, five hand sets, and one PDA • Ericsson WAPPIDE SDK version 2.0B8 simulator • Nokia WAP Toolkit version 1.3 beta simulator • Phone.com UP.Simulator 3.2 simulator • Ericsson R380S handset • Ericsson R320S handset • Nokia 7110 handset • Two handsets whose identifies were withheld due to a nondisclosure agreement • Ericsson MC218 version 1.13 PDA

**1. Navigation tag**

(i). Use anchored elements to construct menus for navigating through an application's options.

(ii). Separate anchored elements from other anchored element with <br>or <p> tags

(iii). Avoid anchored elements that are longer than a single screen. If possible, keep them shorter than one line in length (12 character )Particularly if they are used as part of a list of anchors

(iv). If it is not feasible to keep anchored elements shorter than one line in length when part of a list of anchored elements specifying the text as now wrap

(v). Limit lists of anchored elements nine items.

(vi). Do not use the same term for the title attribute for an <anchor>element and for the label attribute for a <do type="accept"> event on the same card unless both are intended to invoke the same function.

(vii). Use a meaningful term for the title attribute, but do not make it critical that the user see this term.

**2. <do> Tag**

(i). When duplicating <do> element with anchored links, use the same term as the label for the <do> element and the title for the <a> or <anchor>element and maintain the same functionality.

(ii). When using <do> element use the <do type ="accept">event for the most frequently used task

(iii). Always define the <do type="prev">event. Map this event to the <prev> element for cards in which you want to enable backward navigation through the history stack the <go> element.

(iv). Give <do> events, such as <do type="accept"> and <do type="options">, a meaningful label attribute

(v). Use five or fewer characters for the label1 of a <do type="accept">or <do type="options">

(vi). Use standard label attributes for the <do type="accept"> and <do type ="options">functions and use consistent labels throughout the interface.

(vii) Be careful about defining more than one active <do type="options"> event within a single card.

(viii).Be careful about making necessary functions accessible only by <do type="delete"> or <do type="help">events

(ix). Define the <do type="accept">event for every card, even if the default behavior for a particular browser is desired

(x). Do not use the same label for two or more different <do>events

(xi). Ensure that when two labels are the same (within across<do> element, <select>elements or anchored lists) they produce the same action

(xii). Specify <do>events in the order in which you wish them appear.

**3. <input> Element**

Use the title attribute to give the <input>element a meaningful label

(i) Specify a value for the default attribute when that value is likely to be accepted by most of your users- to minimize text entry.

(ii) Use the format and maxlength attributes prevent user errors

(iii) Test to ensure your format attribute specifies text input in the manner you intended and perform a full input validation on the server as if no format had been specifies.

(iv) Ensure that introductory text prior to the <input> element states the number of characters to be input, if applicable, and the format required (5-digits)

(v) Within your application, be consistent in introducing text entry fields so that users clearly understand when they should enter text.

(vi) Avoid using literal characters such as parentheses or dashes, for imposing a known format
on user entered data without specifying the appropriate format(xxx-xxx-xxx).

(vii) Do not use the password type, even for entering passwords unless only numeric input is Required.

**4. <SELECT> Element**

(i) Avoid presenting more than nine items in a <select> list.
(ii) Use the title attribute to give a meaningful label to the <select>list
(iii) Do not rely on the title attribute being displayed.
(iv) Define a default value for the <select>list. This value should be the most common choice.
(v) Place descriptive introductory text immediately prior to the <select>list
(vi) Specify a name for the <option>title attribute
(vii) If the user does not have to select an option from the select list, provide an option item such as "none"

## USER INTERFACE DESIGN GUIDELINES

### <SELECT> ELEMENT

1. Avoid presenting more than nine items in a<select> list.
2. Use the title attribute to give a meaningful labelto the <select>list
3. Do not rely on the title attribute beingdisplayed.
4. Define a default value for the <select>list. Thisvalue should be the most common choice.
5. Place descriptive introductory text immediatelyprior to the <select>list
6. Specify a name for the <option>title attribute
7. If the user does not have to select anoption from the select list, provide anoption item such as “none”

### <IMG> ELEMENT

1. <img>Element-used to display graphics on thescreen. If the device does not support images, the altattribute is displayed in its place
2. Use the alt attribute to give a meaningful name tothe image
3. Use application specific images selectively
4. Avoid using images that are longer than the devicedisplay
5. Use wireless bitmap images instead of windowsbitmaps (WBMP), PNGs, GIFs or other types ofimages
6. Do not use the localsrc attribute unless the contentis targeted to a particular browser.

### <P> ELEMENT

1. Used to display text on the screen
2. Do not count on browsers to support soft hyphens.
3. Do not use hyphens if the sole purpose is to break aword.
4. When using different alignments within a card, specifythe alignment of all text elements.
5. Do not count on alignment being supported on allbrowsers.
6. If the nowrap mode is used , make sure that the text
will be understandable if shortened to ten characters.
6. Use left text alignment when the paragraph is innowrap mode
7. If possible, choose words that are shorter than 12characters, including format indentations by thebrowser

8. Make sure that anchored text does not containunderscores.

**<TABLE> ELEMENT**

1. Used to organize text in tables
2. Avoid using the <table>element whenpossible
3. Make sure that anchored text does notappear in tables
4. Do not use the <table> element as alayout tool
5. Do not include more than 10 charactersin a table row

**<CARD> ELEMENT**

1. Use the title attribute to give the card a meaningfulname
2. Do not rely on the title attribute being displayed
3. Ensure that the title makes sense when viewed alone
4. Ensure that the title make sense if shortened to 12characters
5. Use the ordered=true – when your card contains shortforms containing only required fields
6. Use the ordered=false option when your card containsshort forms that can be completed in any order
7. If the ordered attribute is false, make sure that all textimmediately preceding an <input> or select element

**<fieldset> element**

• Allows for grouping of related fields andtext

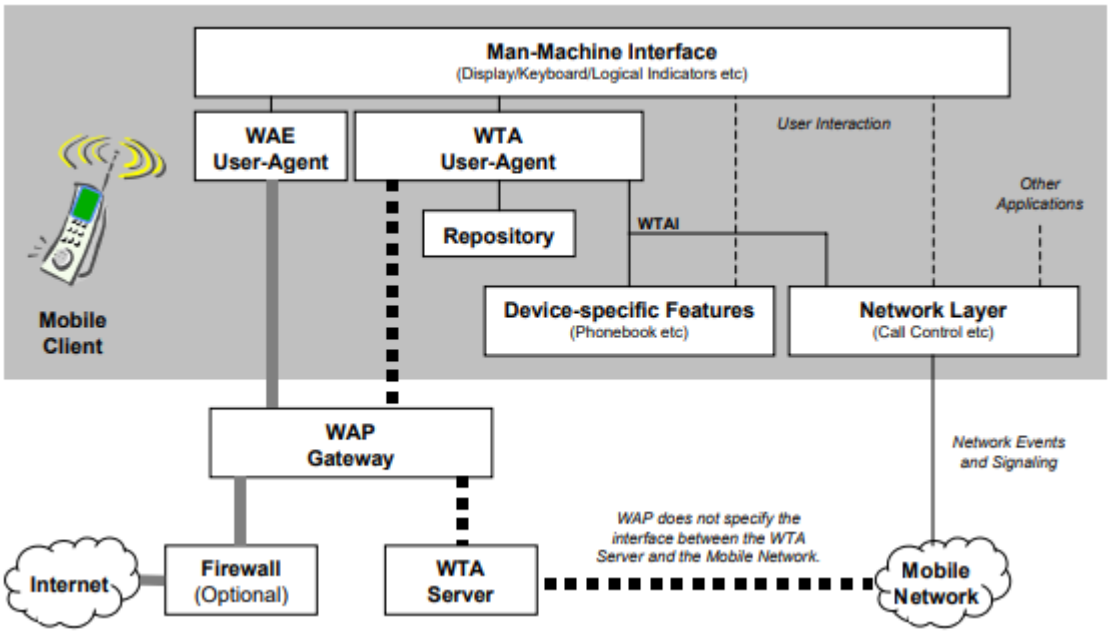
**<optgroup>element**

• Allows for groupings of related optionelements into a hierarchy

## UNIT V

### Overview of the WTA Architecture

WTA is an application framework for telephony services. The WTA user-agent essentially is a user-agent similar to the standard WML user-agent with the addition of capabilities for interfacing with mobile network services available to a mobile telephony device, e.g. setting up and receiving phone calls. The figure below describes one possible configuration of the WTA framework. However, this specification solely defines the components contained in the client.




As mentioned above, the WTA framework relies on a dedicated WTA user-agent in the client, The WTA server is not specified by WAP, but a brief overview is given below.

**1. The WTA User-Agent**

Figure 1 above illustrates how the WTA user-agent, the repository (persistent storage) and WTAI (telephony application interface) interact with each other and other entities in a WTA-capable mobile client.

The WAE user-agent only retrieves its content via the WAP gateway. In addition, the WTA user-agent is also able to retrieve content from the repository. Further, WTAI ensures that the WTA user-agent can interact with mobile network functions (e.g. setting up calls) and device specific features (e.g. manipulating the phonebook).

**2. WTA Server**

The WTA server can be thought of as a web server delivering content requested by a client. Like an Internet web browser, a WTA user-agent uses URLs to reference content on the WTA server. A URL can also be used to reference an application on a web server (e.g. a CGI script1) that is executed when it is referenced. Such applications can be programmed to perform a wide range of tasks, for example generate dynamic content and interact with external entities.

A WTA server may also make use of this concept. By referencing applications on a WTA server it is possible to create services that use URLs to interact with the mobile network (e.g. an IN-node) and other entities (e.g. a voice mail system). Thus, the concept of referencing applications on a WTA server provides a simple but yet powerful model for how to seamlessly integrate services in e.g. the mobile network with services executing locally in the WAP client.

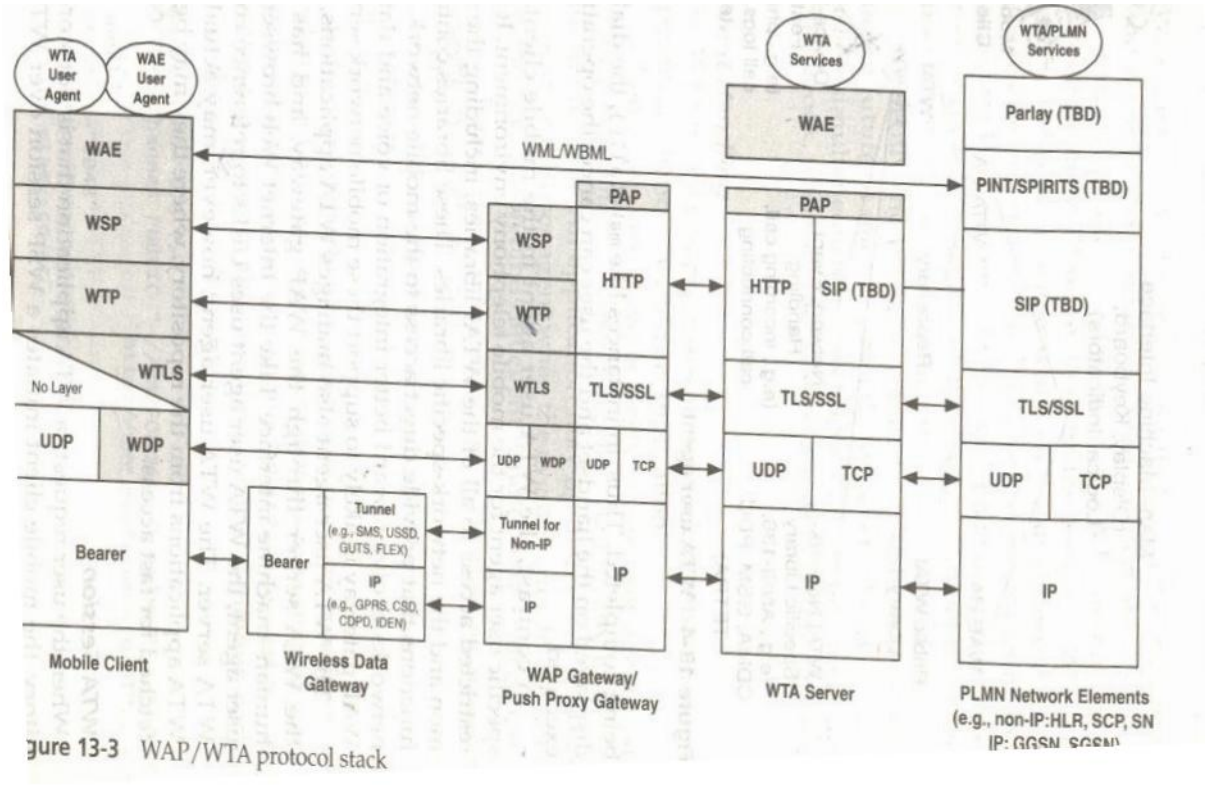
## WTA Client framework

• WTA client includes a user agent, the repository, and a WTA application programming interface.

Two user agents in the mobile client

1) Wireless Application Environment (WAE) user agent

2) WTA user agent – share the application environment (fig 1) and (fig 2)



gure 13-3 WAP/WTA protocol stack

**Figure 1: WAP/WTA protocol stack**

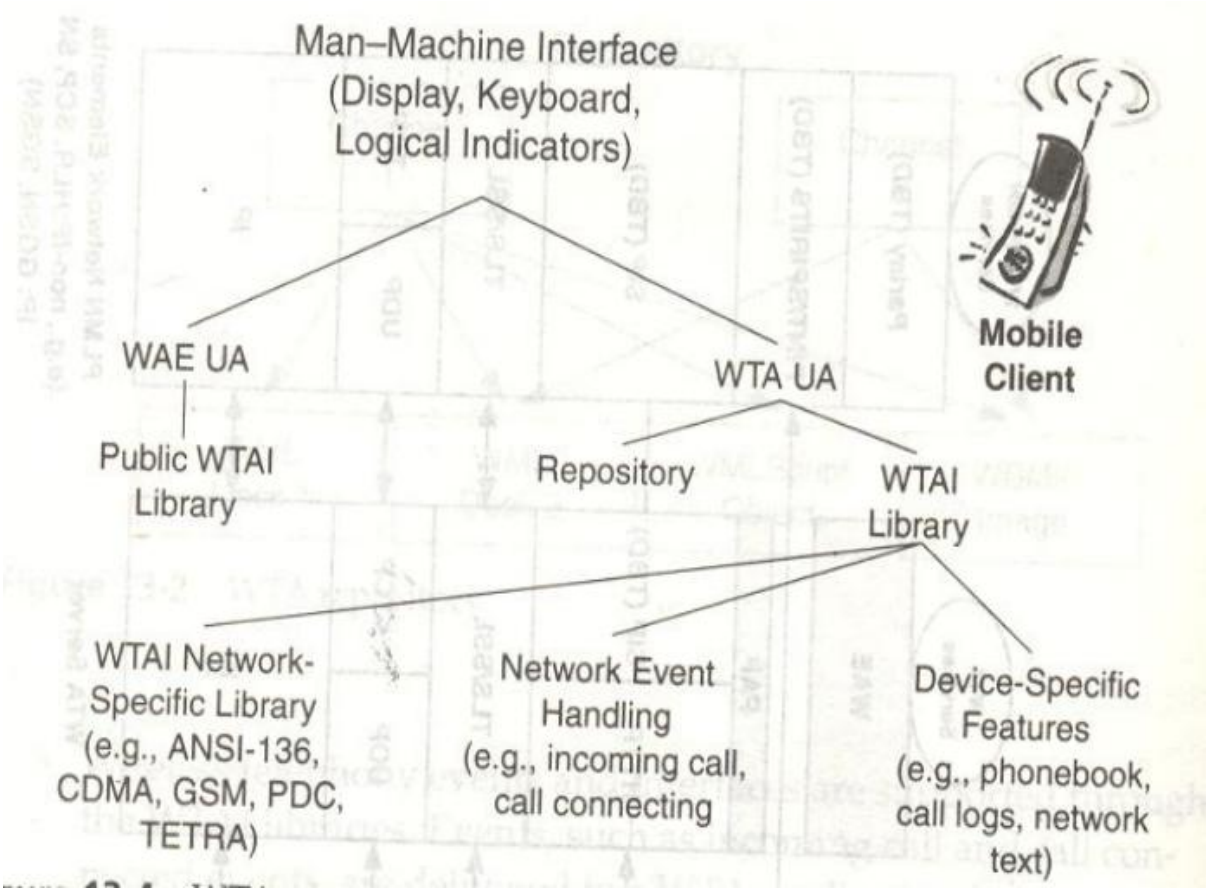



**Fig 2: WTA User Agent**

**Repository:**

A repository (storehouse or warehouse) provides persistent client side storage for WTA app. – that retrieved or pushed from the WTA server prior to use.

-WTA app. are stored as channels in the repository

-channels made up of entities-called resources includes WML decks, WML Script, WBMP (wireless Bit) map.

**Library** - WTA Public uses library "wp."

**Event Handling:**

WTA user agent processes all network events and directs them to an appropriate handler within an application

• When a n/w event is received, the user agent first attempts to perform local binding, dispatching the event locally to the event locally to the currently executing program.

• An app. may bind an event to a handler that is invoked by the user agent when the event occurs.

**WTA server and security:**

WTA server- WTA applications in the form of WML decks and WML script.

Wireless Telephony Applications (WTA) are those applications designed to interact with the telephony-related functions present in a phone. These applications can include:

i. Originating a call
ii. Adding, searching, and removing Contacts entries
iii. Sending and receiving Short Text messages
iv. Examining call logs, including calls made, received, or missed
v. Sending DTMF tones during an active voice call
vi. Pushing content to the Prompt Line

• Responding to requests- mobile client

In the WTA security model, any entity may become a WTA Service Provider by being approved for access to a trusted gateway. Access control of the trusted gateway by the WTA servers should be enforced using existing secure solutions.

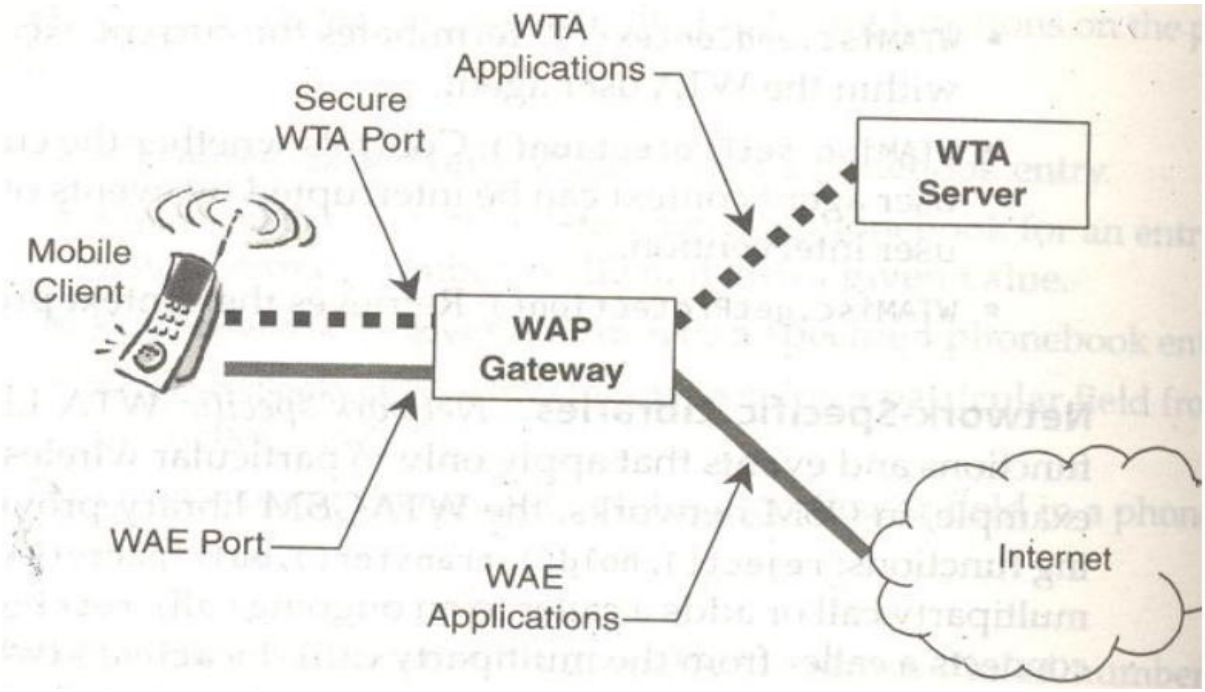



**Fig 3: WTA security model.**

In order to provide security to WTA, a WAP gateway may control the access between the WTA user agent and the WTA server. The WAP gateway should verify that the providers of WTA pull/push content are authorised.

**Design Considerations:**

Performing network mapping, service design, development, and testing

• Service concept- behavior and Interaction with the user

• Network mapping- service is mapped to a particular mobile network and the network architecture.

• Service design- service developer determines which pieces of app. Already resident in the mobile client- service developer whether those resources can be reused- which new pieces can be loaded.

**Network design considerations:**

• Real time loading Issues.

• Acceptable perception of latency- usability studies- a user will wait at most six seconds after making a request to see or hear

• Bearer Selection- used to retrieve the application, among other things- each network offers a different combination of latency and bandwidth

• Currently available bearer networks- SMS, CDPD, CSD and GPRS

• Memory Constraints- Small amount of memory in the mobile client – general purpose, low power microprocessor used in the client devices are clearly getting faster.

**Application design considerations:**

Limited over the air bandwidth, memory and computing resources are in the mobile

Client Developer must consider the following issues.

• Partitioning

• Repository

• Event Handling

• Network specific WTAI features

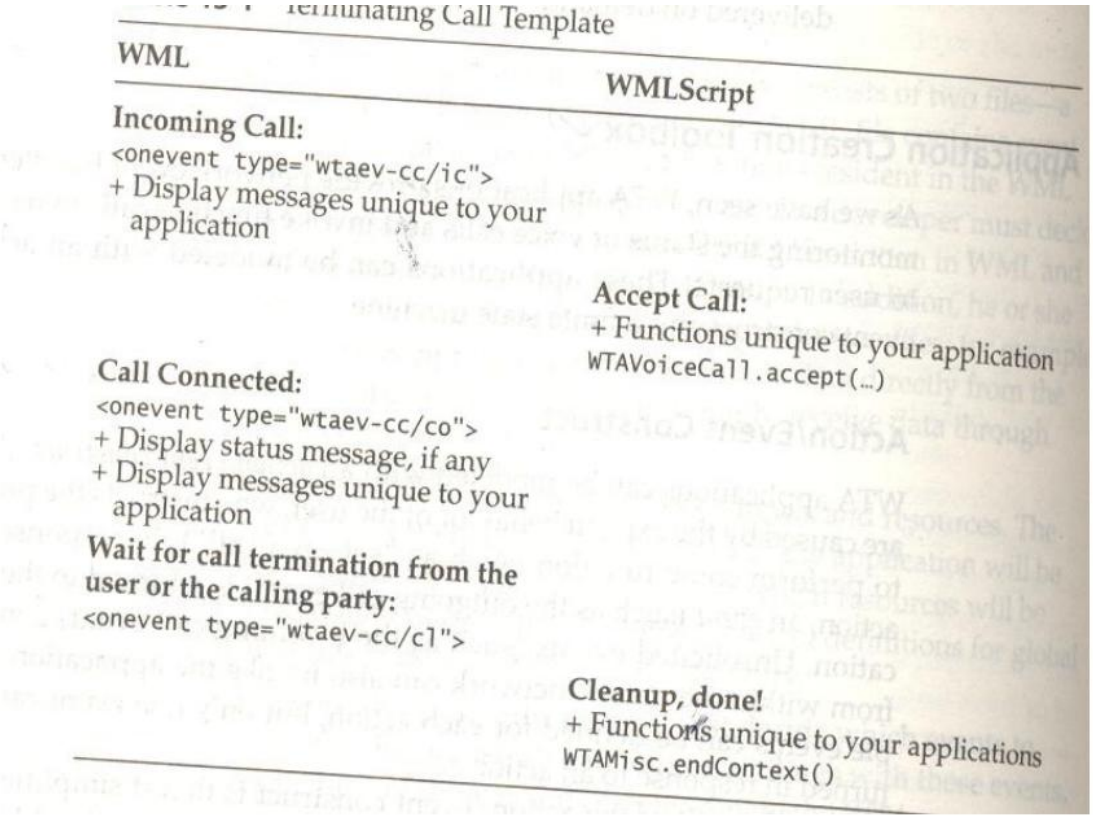
• Error handling

• Delivery to the mobile client

**Application creation tool box**

• Action/Event Construct

• Terminating call Template


Terminating Call Template

| WML | WMLScript |
| --- | --- |
| **Incoming Call:**<br>`<onevent type="wtaev-cc/ic">`<br>+ Display messages unique to your application | |
| | **Accept Call:**<br>+ Functions unique to your application<br>`WTAVoiceCall.accept(…)` |
| **Call Connected:**<br>`<onevent type="wtaev-cc/co">`<br>+ Display status message, if any<br>+ Display messages unique to your application | |
| **Wait for call termination from the user or the calling party:**<br>`<onevent type="wtaev-cc/cl">` | |
| | **Cleanup, done!**<br>+ Functions unique to your applications<br>`WTAMisc.endContext()` |

**Terminating Call Template**

**Future of WTA Enhancements**

In the next few years, mobile phones will start to benefit from very high bandwidth capabilities. The 2.5G/3G systems will allow much higher capacity and data rates, than can be offered by the restricted bandwidth currently available.

These wireless devices will be supported by a number of emerging technologies including GPRS, EDGE, HSCSD, and UMTS.

We consider several potential enhancements for future releases of the specification

• WTA server APIs-today- delivers WTA application to the mobile client.

• It may be extended to support integration with the call control features of mobile networks

• External Device Events- future-users will want to extend the capabilities of the mobile terminal by connecting it to external devices such as laptop PC or PDA

• Multiple context Support- future user agent ability to instantiate a new context for each WTA call and Distribute events for both incoming and outgoing calls

• Improved Event handling