

# **Wireless Internet Architecture**

**Course Designer and Acquisition Editor**

**Centre for Information Technology and Engineering**

**Manonmaniam Sundaranar University**

**Tirunelveli**

---

# **Wireless Internet Architecture**

---

---

## CONTENTS

Lecture 1	Introduction to WAP	1
	Introduction	
	Background of WAP	
	WAP Overview	
	Characteristics Features of WAP	
	Applications of WAP	
	Benefits of Adopting WAP	
	Limitations of WAP	
	Desired Characteristics of a WAP Solution	
	Future Developments in WAP	
Lecture 2	WAP Communication Model	15
	World Wide Web Architecture Overview	
	WAP Model	
	WAP Implemented Wireless Networks	
	WAP Communication Model	
	• WAP Client	
	• WAP Gateway	
	• WAP Application Server	
	WAP Portals	
Lecture 3	WAP Protocol Architecture	32
	Protocol and Layers	
	WAP stack and WEB stack	
	Wireless Application Environment (WAE)	
	WAE Useragent	
	WTA	
	Wireless Session Protocol (WSP)	
	Wireless Transport Protocol (WTP)	
	Wireless Transport Layer Security (WTLS)	
	Wireless Datagram Protocol (WDP)	
Lecture 4	Introduction to WML	45
	WML Structure	
	Characteristics of WML	
	A Basic WML Card	
	The Header	
	The Body	
Lecture 5	Style Element, Tables and Entities	52
	Text formatted Elements	
	<head> element	
	Tables	
	WML entities	

Lecture 6	WML Elements	62
	<ul style="list-style-type: none"> <li>• Navigational Elements</li> <li>• WML Task Elements</li> <li>• Template Elements</li> </ul>	
Lecture 7	Links and Images	71
	<ul style="list-style-type: none"> <li>• Links</li> <li>• Images</li> <li>• Interacting with Users</li> </ul>	
Lecture 8	WML Events	80
	<ul style="list-style-type: none"> <li>• Ontimer element</li> <li>• Onenterforward element</li> <li>• Onenterbackward element</li> <li>• Onpick element</li> <li>• Variables</li> </ul>	
Lecture 9	WML Script	91
	<ul style="list-style-type: none"> <li>• WML Script Language Syntax</li> <li>• How to call a WMLScript from a WML page</li> <li>• Data types</li> <li>• Operators</li> </ul>	
Lecture10	WML Script Controls and Functions	103
	<ul style="list-style-type: none"> <li>• Control Constructs</li> <li>• Functions</li> <li>• Break statement</li> <li>• Continue statement</li> <li>• Return statement</li> </ul>	
Lecture11	WML Script Library	112
	<ul style="list-style-type: none"> <li>• Libraries</li> <li>• Dialogs Library</li> <li>• Float Library</li> <li>• Lang Library</li> </ul>	
Lecture12	WML Script Library	118
	<ul style="list-style-type: none"> <li>• String Library</li> <li>• URL Library</li> <li>• WML Browser Library</li> </ul>	
Lecture13	ASP and Object Model	145
	<ul style="list-style-type: none"> <li>• ASP object Model</li> </ul>	
Lecture14	WAP MIME Types	154
	<ul style="list-style-type: none"> <li>• MIME types</li> <li>• Request and Response objects</li> </ul>	








Lecture15 Introduction to ADO	159
Cookies	
ADO Object Models	
Lecture16 Database Handling in WAP	168
Receiving Values from the WAP client and stored in a Database	
Retrieving from the Database and send to the client.	
Lecture17 Using Servlets in WAP	173
Power of Servlets	
Servlet API	
Basic Servlets Structure	
Life Cycle of Servlets	
Request and Response Headers	
Servlet and Cookies	
Lecture18 Using JSP in WAP	186
JSP Request Model	
JSP Architecture	
Using JDBC in JSP	
JSP Error Page	
Application using JSP.	
Lecture19 WAP Security	195
Security concepts	
Lecture20 Encryption Technologies	200
Cryptography	
Keys	
Ciphers	
Cipher Suites	
Certificates	
Security in WAP	
Role of SSL and WTLS in WAP communication.	
Lecture21 PUSH and WTA Technology	210
WAP push model	
Push framework	
Push Proxy Gateway	
Push Access Protocol	
Push Over The Air Protocol	
Lecture22 Introduction to WTA	220
WTA architecture	
WTA Services	

---

Lecture23 Wireless Technologies

226

---

-  I-mode Introduction
-  I-mode Internet components
-  GPRS
-  Bluetooth Wireless Solution Components
-  Bluetooth
-  Link management
-  Bluetooth Security.

---

Syllabus

235

---

EOGR

## Lecture 1

---

# Introduction to WAP

---

## Objectives

After completing this Lecture, you should be able to do the following

- ✔ Background of WAP
- ✔ Characteristics Features of WAP
- ✔ Limitation of WAP
- ✔ Future Development in WAP

---

## Coverage Plan

---

### Lecture 1

- 1.1 Snap Shot
- 1.2 Background Of WAP
- 1.3 WAP Overview
- 1.4 Characteristics Features Of WAP
- 1.5 Applications Of WAP
- 1.6 Benefits Of Adopting WAP
- 1.7 Limitations Of WAP
- 1.8 Desired Characteristics Of A WAP Solution
- 1.9 Future Developments In WAP
- 1.10 Short Summary
- 1.11 Brain Storm



## 1.1 Snap Shot

The Internet has emerged as one single technology, which has affected all walks of human life in the last decade of the last millennium. The Internet's penetration into fields such as education, research, medicine, finance, business etc., has initiated fresh ways for conducting business and providing services to the customers. This decade will see the expansion of the Internet reaching out to more millions across the globe and the evolution of new business models for reaching and communicating with the customer. The emerging technologies are projected to provide a new impetus to the Internet proliferation by providing wireless access to information anytime and anyplace.

**Wireless Application Protocol (WAP)** is one such technology that is heralding sweeping changes in the fields of telecommunications and information technology. In the near future, the proliferation of mobile wireless devices, such as cellular phones, is supposed to reach a figure of 1 billion. All these devices will have the capability to transmit data packets along with the voice data and open avenues for supporting Internet based applications. WAP provides a platform to reengineer the existing Internet applications and enhance them with personalization and broadcasting features and deliver the contents on the new wireless devices. For example, when a cheque comes for clearance to a bank, the bank can notify the account holder on his mobile phone and get his approval interactively.

## 1.2 Background of WAP

### WAP Forum

The WAP forum is the industry association dedicated to the goal of enabling sophisticated telephony and information services on hand-held wireless devices such as mobile telephones, pagers, personal digital assistants (PDA's) and other wireless terminals.

The WAP specification was initiated in June 1997. The WAP Forum was founded by Unwired Planet (now Phone.com), Nokia, Ericsson and Motorola in December 1997. Now the WAP Forum has grown to a membership of over 350, made up of carriers, handset manufacturers, infrastructure providers, software developers and other organisations. The WAP Forum's membership roster now includes computer industry heavyweights such as Microsoft, Oracle, IBM and Intel along with several hundred other companies. With 90% of the handset market now being represented at the WAP Forum, along with many software companies and network operators, WAP will be the primary way of accessing the Internet in the future.

Recognizing the value and utility of the World Wide Web architecture, the WAP forum has chosen to align certain components of its technology very tightly with the Internet and the WWW. The WAP specifications extend and leverage mobile networking technologies such as IP(Internet Protocol), HTTP(HyperText Transfer Protocol), XML(Extensible Markup Language) and URLs (Uniform Resource Locators), scripting and other content formats.

The objectives of the WAP Forum are:

- To bring Internet content and advanced data services to digital cellular phones and other wireless terminals.
- To create a global wireless protocol specification that will work across different wireless network technologies.
- To enable the creation of content and applications that scale across a very wide range of wireless bearer networks and wireless device types.
- To embrace and extend existing standards and technology wherever appropriate.

***What is WAP Forum?***

*WAP Forum, founded in 1997, is the industry association of carriers, handset manufacturers, infrastructure providers, software developers etc., dedicated to bring internet and other information services to handheld wireless devices.*

**Philosophy of WAP**

WAP takes a client sever approach. It incorporates a relatively simple microbrowser into a mobile phone, requiring only limited resources on the mobile phone. This makes WAP suitable for thin clients and early smart phones. WAP is aimed at turning a mass- market mobile phone into a “network based smart phone”.

The philosophy behind the WAP approach is to utilise as few resources as possible on the handheld device and compensate for the constraint of the device by enriching the functionality of the network.

**1.3 WAP – An Overview**

WAP is a hot topic that has been widely hyped in the mobile industry and outside of it. WAP is simply a protocol – a standardised way by which a wireless device talks to a server installed in a wireless network. WAP provides a method to communicate across wireless networks quickly, securely and efficiently. WAP provides the opportunity to integrate databases, dynamic content, e-commerce and secure information trafficking through an WAP-enabled device. Although the name itself refers to a single protocol, WAP can actually be thought of a compilation of protocols brought together to cover many aspects of wireless communications.

**Definition**

The **Wireless Application Protocol (WAP)** is an open, global specification, which gives mobile users with access to wireless devices the opportunity to easily access and interact with information and services. WAP was developed by *WAP forum*.

***What is WAP?***

*“WAP is an open global standard for communication between mobile devices and the Internet or other computer applications, defined by the WAP forum.”*

**WAP Components**

The deployment of WAP technology is dependent on four components:

- Clients
- Gateways
- Servers
- Applications

We will now briefly look at each of the components.

**Clients**

WAP client is what the end user will be using. WAP clients, like the Internet clients - the web browsers, will decide on how to present the information and what it can handle.

### Gateways

WAP gateways are a key component. They provide the link between the technologies used in the wireless domain and the Internet.

### Servers

The WAP servers host the WAP content and applications. They provide a framework for building robust and high-performance applications.

### Applications

Applications are the components that the end users really see. This is what the users will be using the wireless devices for.

### WAP Internal Structure

Before we look at how the WAP protocols are structured, let us briefly examine the concepts of a **protocol**, a **layer** and a **protocol stack**.

#### Protocol

A protocol defines the type and structure of messages that two devices have to use when they are communicating with each other.

#### Layers

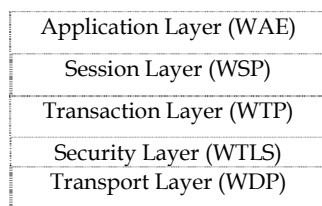
Since protocols are functionally and logically divided into different groups of functionality, they are also physically framed into layers. Each layer provides a specific service to the next layer. For example, one layer may provide methods to send bits down a physical cable; another may supply methods to establish a connection.

#### Protocol Stack

The protocol stack is the set of all the layers that comprise the set of protocols.

#### WAP Protocol Stack

We will now briefly look at how the WAP protocol is structured. The WAP stack, illustrated in the following diagram, has five different layers.



- **Application Layer:** Wireless Application Environment (WAE) provides an application environment intended for the development and execution of portable applications and services.
- **Session Layer:** Wireless Session Protocol (WSP) supplies methods for the organised exchange of content between client/server applications.

- **Transaction Layer:** Wireless Transaction Protocol (WTP) provides different methods for performing transactions, to a varying degree of reliability.
- **Security Layer:** Wireless Transport Layer Security (WTLS) is an optional layer that provides, when present, authentication, privacy and secure connections between applications.
- **Transport Layer:** Wireless Datagram Protocol (WDP) is the bottom layer of the WAP stack which shelters the upper layers from the bearer services offered by the operator.

## 1.4 Characteristics Features of WAP

WAP offers a number of useful features for communicating via the wireless medium. These features have a very close relationship with the Internet. Rather than re-inventing the wheel, WAP developers have based their design upon currently existing ideas and standards. This has given an even further reason for developers to implement services and products geared towards wireless data transfer. WAP developers are attempting to produce a cohesive environment with that of the existing infrastructure not only on the level of data transfer, but also in the way the developers can communicate this information through a common language. Many of the features described below show how existing implementations have been modified for use on this new medium. Few of the existing features have been described below:

### **Optimization of Data Transfer**

It is important to remember that wireless data transfer is (around 9.6 kbps). For this reason, the WAP developers concentrate upon optimizing existing protocols such as HTTP (HyperText Transport Protocol) and TCP (Transmission Control Protocol) to maximize data transfer while taking low bandwidth into account.

### **Operating System Independent**

WAP can be used on any operating system, including such popular handheld platforms like PalmOS, Windows CE, and JavaOS. It can also be compatible with Windows 95/98/NT, Linux, Solaris, etc. This is because the protocol is dependent upon communications standards rather than the platform itself. Any platform capable of adhering to the communications standards is WAP compatible.

### **Handheld Device Markup Language (HDML)**

HDML is an integral part of the WAP. HDML is the language, which provides a programmer with the tools to produce content for a handheld WAP Device.

### **Wireless Markup Language (WML)**

WML is also a markup language, which is used to interface with the WAP browser. It allows a programmer to simultaneously develop web-based applications that can be viewed both within a traditional web browser and within a hand held device.

### **Wireless Telephony Application (WTA)**

WTA is responsible for the actual interfacing of the device to the wireless network. This includes the handling of network events that allow the device to interact with a remote application. These events are similar to those that take place when the 'submit' button is pressed on an HTML-enabled page. WTA makes this same sort of events possible, put into action when the user depresses a key on the phone.

*What are the characteristic features of WAP?*

*The characteristic features of WAP have a close relationship with the internet:*

- *Handheld Device Markup Language (HDML) & Wireless Markup Language*

(WML)

- *Optimisation of data transfer*
- *Independent of operating system*
- *Application of wireless telephony*

## 1.5 Applications of WAP

WAP technology can be adopted for various applications. We shall now look at a few of the applications.

### **CORPORATE APPLICATIONS**

#### **Sales force Automation**

Sales people spend between twenty to eighty percentage of their time in a day away from a PC terminal or a wired computing device. The handset can now be used to provide instant, direct access to the latest pricing and competitive information anytime, anywhere. Important sales leads and contact information updated with the latest news from the office are always available. A salesman can give a call to the data sheet that the customer is looking for and fax it to the nearest fax machine while the customer is waiting.

#### **Dispatch**

Keeping delivery and service personnel aware of every changing orders and schedules becomes easy with a micro-browser-enabled handset. The phone alerts the technician every time a schedule change occurs and provides them with a one-button access to view the changes.

### **ONLINE SERVICES**

#### **Real Time Delivery of Content**

Subscribers can now get the current information about the weather, news and stocks whenever they need it, wherever they are. Real time information can also be delivered to a subscriber's handset, providing up-to-the minute stock level, traffic alerts etc.

#### **Banking**

Nowadays banks have introduced new ways such as ATM and mini branches in supermarkets to make banking tasks easier. The next logical step is to bring the ATM features to the display of a wireless handset. Of course, subscribers will not be able to withdraw cash or make a deposit but they can get their current balance, transfer funds between accounts and receive a fax of a mini statement. With the online banking extensions subscribers could also pay bills right from their phone.

#### **M-Commerce**

Subscribers can use their handset just like their PC to purchase products and services over the Web.

### **PERSONAL PRODUCTIVITY**

#### **Email**

Wireless subscribers can keep track of their mail right from their handset. In addition to receiving email, subscribers can compose and send reply to messages.

#### **Organizer**

WAP enables the subscriber to have a real time copy of their address book. Using notifications, subscribers can be alerted when a co-worker changes an appointment. They can also select a contact number from their address book and direct the application to dial the contact with the push of a button.

## **COMMUNITY AND GAMES**

### **Interactive Chat**

People around the world are adopting new technologies to stay in touch. Mobile phones with ring stones can be used to identify friends, send and receive emails and notifications while on the go. This makes staying in touch easy and fun.

### **Games**

Wireless subscribers can test their knowledge against other wireless phone users in a game or enjoy a round of cards.

#### ***What can WAP devices be used for?***

- *Ticket purchase*
- *Airway/Railway Time Schedule*
- *Traffic alerts*
- *Shopping*
- *Banking services*
- *Email*
- *Advertisements*
- *Games*
- *Chatting*
- *Stock exchange information*

## **1.6 Benefits of adopting WAP**

With the WAP technology likely to make rapid strides the benefits of adopting WAP are enormous. We shall now look at the benefits of adopting WAP from the point of view of the three key players viz., network operators, content providers and end users.

### **Network Operators**

For wireless network operators, WAP promises to decrease churn, cut cost and increase the subscriber base. WAP hopes to accomplish this both by improving existing services such as interfaces to voice mail and prepaid systems, and facilitating an unlimited range of new value-added services and applications, such as account management and billing inquires. New applications can be introduced quickly and easily without the need for additional infrastructure or modifications to the phone. This will allow operators to differentiate themselves from their competitors with new, customized information services. Since WAP is an interoperable framework enabling the provision of end to end turnkey solutions, it will create a lasting competitive advantage, build consumer loyalty, and increase revenues.

### **Content Providers**

WAP applications will be written in Wireless Markup Language (WML), which is a subset of extensible markup language (XML). Using the same model as the Internet, WAP will enable content and application developers to grasp the tag-based WML. This will pave the way for services to be written and deployed

within an operator's network quickly and easily. As WAP is a global and interoperable open standard, content providers have immediate access to a wealth of potential customers from their own existing and potential subscriber base who will seek such applications to enhance the services offered to them. Mobile consumers are becoming hungrier to receive increased functionality and value-additions from their mobile devices. WAP opens the door to this untapped market that is expected to grow rapidly. This presents developers with significant revenue opportunities.

### End Users

End users of WAP will benefit from easy and secure access to relevant Internet information and services such as unified messaging, banking and entertainment through their mobile devices. Intranet information such as corporate databases can also be accessed via WAP technology. Since a wide range of handset manufacturers already supports the WAP initiative, users will have significant freedom of choice while selecting mobile terminals and the applications they support. Users will be able to receive and request information in a controlled, fast, and low-cost environment, a fact that renders WAP services more attractive to consumers who demand more value and functionality from their mobile terminals.

The Internet has and will set many of the trends in advance of WAP implementation. It is expected that the ISPs will exploit the true potential of WAP. Web content developers will have greater knowledge and direct access to the people they attempt to reach. In addition, these developers will acknowledge the huge potential of the operators' customer bases; thus they will be willing and will be able to offer a competitive price for their content. WAP's push capability will enable weather and travel information providers to use WAP. This push mechanism affords a distinct advantage over the WWW and represents tremendous potential for both information providers and mobile operators.

#### *What are benefits of adopting WAP?*

- *Wireless network operators can decrease churn, cut costs and increase subscriber base.*
- *Content providers can have access to customers who seek enhanced services.*
- *End users can have more value and functionality from their mobile devices.*

### 1.7 Limitations of WAP

Most of the technologies developed for the Internet have been designed for desktop and larger computers that support medium to high bandwidth connectivity over generally reliable data networks. Hence, providing Internet and WWW services on a wireless data network that supports a lower bandwidth presents many challenges.

#### Device limitations

Mass-market, hand-held wireless devices present a more constrained computing environment compared to desktop computers. Because of fundamental limitations of power and form factor, mass-market devices tend to have:

- Less powerful CPUs
- Less Memory (ROM and RAM)
- Restricted power consumption
- Smaller Displays
- Different input devices (e.g. a phone keypad, voice input, etc.)

#### Bearer Limitations

Similarly, wireless data networks present a more constrained communication environment compared to wired networks. Because of fundamental limitations of power, available spectrum and mobility, wireless data networks tend to have:

- Lesser bandwidth than traditional networks
- More latency than traditional networks
- Lesser connection stability than other network technologies
- Lesser predictable availability.

#### **Use Case Limitations**

Most wireless devices are mass-market consumer devices. This places a severe restriction on the type of usage that can be envisaged for these devices. These devices should have:

- Simpler and easier user interfaces than personal computers.
- More specific set of use cases
- Non-hindering nature

#### *What are the limitations of WAP?*

- Device limitations that arise from limitations of power and form-factor.
- Bearer limitations that arise from limitations of bandwidth and mobility.
- Use case limitations that arise from the consumer nature of mobile devices.

We shall now look at the limitations in greater detail.

#### **Device Limitations**

Wireless devices operate under a set of physical limitations, imposed by their mobility and form factor. The physical limitations are as discussed below:

##### **Limited Power**

Any personal, or 'hand held' mobile device will have a very limited power reserve, due to existing battery technology. This reduces available computational resources, transmission bandwidth, etc.,

##### **Scalability**

Mobile wireless devices are characterized by a different set of user interface constraints than a personal computer, To enable a consistent application-programming model; a very wide range of content scalability is required. In the future this should include voice-only input/output. In practice, a significant amount of the current WWW content is unsuitable for use on hand-held wireless devices. Problems include:

##### **Output scalability**

Existing content is designed for viewing on PC screens, whereas mobile devices will have a wide range of visual display sizes, formatting and other characteristics.

##### **Input scalability**

Mobile devices have a wide range of input models, including numeric keypads, very few programmable soft keys, etc.



**Bearer Limitations**

Wireless network bearers operate under several fundamental constraints, which place restrictions on the type of protocols and applications offered over the network. A few of the constraints are given below.

**Cellular network economics**

Mobile networks are typically based on a cellular architecture. Cells are a resource shared by all mobile terminals in a geographic area, and typically have a fixed amount of bandwidth to be shared among all users. This characteristic demands efficient use of bandwidth, as a means of reducing the overall cost of the network infrastructure.

**Latency**

The mobile wireless environment is characterized by a very wide range of network latency, ranging from sub-second round-trip communication time up to many tens of seconds. In addition, network latency can be highly variable, depending on the current radio transmission characteristics and the network loading in a particular area. Routing, error correction and congestion-avoidance characteristics of a particular network further increase latency.

**Bandwidth**

The mobile wireless environment is characterized by a very wide range of network characteristics, and typically has far less bandwidth available than a wired environment. In addition, the economics of the wireless environment encourage the conservation of bandwidth to achieve greater density of subscribers.

**Power Consumption**

As bandwidth increases, power consumption increases. In a mobile device, this reduces battery life.

**Use Case Limitations**

Many wireless devices, for example cellular phones and pagers, are consumer devices. These devices are used in a wide variety of environments. This places a severe restriction on the type of usage for these devices. The limitations are as given below:

**Simple user interfaces**

Many mobile devices, in particular, cellular telephones, are mass-market consumer oriented devices. Their user interfaces must be extremely simple and easy to use.

**Single purpose devices**

The goal and purpose of most mobile devices is very focused. This is in contrast to the general-purpose tool-oriented nature of a personal computer. This motivates a very specific set of use cases, with very simple and focused behavior.

**Hands-free, Heads-up Operation**

Many mobile devices are used in environments where the user should not be unnecessarily distracted (e.g. driving and talking). This places a restriction on the type of interfaces possible on the mobile devices.

## 1.8 Desired Characteristics of a WAP Solution

Mobile networks are growing in complexity and the cost of providing new value-added services to wireless users is increasing. In order to meet the requirements of mobile network operators, the WAP solutions must be:

- **Interoperable**

The solutions developed should enable terminals from different manufacturers to communicate with services on the mobile network.

*Scalable*

Mobile network operators should be able to scale services to customer needs.

*Efficient*

The solutions should provide quality of service suited to the behaviour and characteristics of the mobile network; provide for maximum users for a given network configuration.

*Reliable*

Mobile network operators should be able to provide a consistent and predictable platform for deploying services.

*Secure*

Mobile network operators should be able to extend their services over potentially unprotected mobile networks while still preserving the integrity of user data; protect the devices and services from security problems such as detail of service.

*How can a good WAP solution be developed? The WAP solutions should be developed such that*

- *The terminals of different mobile devices are interoperable.*
- *Services are scalable.*
- *Mobile networks are made efficient.*
- *Mobile networks are able to provide a consistent and predictable platform.*
- *Mobile networks are able to provide protection for the devices and the services.*

## 1.9 Future Development in WAP

WAP is focused on enabling the interconnection of the web and wireless terminals. Significant focus has been given to mobile telephones and pagers, but the technology has been developed with broader applicability in mind. The goal of WAP is to enable an extremely wide range of powerful devices, to enjoy the benefits of web technology and intercommunication.

The WAP forum's exclusive focus is on mobile wireless technologies. Its goal is to create recommendations and specifications that support the creation of advanced services on wireless devices, with particular emphasis on the mobile telephones. The WAP forum is creating recommendations and technologies, which would enable these services on all devices and on all networks.

The WAP forum has undertaken a variety of technical specification works relevant to the W3C/WAP forum's collaborative efforts. All these efforts relate to the use of World Wide Web technology on mobile devices, and ensure that the quality of the services provided is sufficient for mass deployment. Mobile

devices have a unique set of features, which must be exposed to the web. The ultimate aim of the WAP forum is to achieve the **Mobile Wireless Web**.

Intelligent network functionality includes integration of voice into the network. The WAP forum is actively exploring the specifications in these areas. Future technical work will address the integration of the ongoing evolution of wireless networks and mobile communication devices. A lot of work is being done in this direction which includes:

#### **Increase the Bandwidth efficiency**

The WAP forum is working towards increasing the bandwidth efficiency of web technology, to make it more applicable to the wireless environment. The work in this direction includes:

#### **Developing Smart Web Proxies**

Proxies capable of performing intelligent information of protocols and content and adapting to device and network characteristics, which enable more efficient use of the network, are being developed.

#### **Efficient Content Encoding**

Bandwidth efficient encoding of standard web data formats such as XML are being developed.

#### **Efficient Protocols**

Bandwidth efficient adaptations of standard web protocols, such as HTTP are in the pipeline.

#### **Facing Latency Constraints**

The WAP Forum is working towards improving the behavior of web technology in the face of high network latencies, and in particular the focus is to address the problems of:

- Tuning network protocols to be adaptive and efficient given wide ranging latencies.
- Creating Web Applications, which are resilient to either high latency environments or highly variable latency situations.

#### **Scalability of Content**

Mobile wireless devices are characterized by a different set of user interface constraints than a personal computer. The WAP forum's work in this area includes:

#### **Content adaptation**

Mechanisms allowing a web application to adapt gracefully to the characteristics of the device are to be developed.

#### **User interface Scalability**

WAP forum is working on the content formats, e.g., markup and display languages, that are suitable for impoverished devices, but which scale well to more sophisticated devices.

*What are the future plans of the WAP Forum?*

- *To achieve the mobile wireless web.*
- *To integrate the wireless network and mobile devices.*
- *To increase the bandwidth efficiency.*

- *To develop smart Web proxies.*
- *To develop efficient content encoding and protocols.*
- *To face latency constraints.*
- *To face scalability constraints.*

### 1.10 Short Summary

- ◆ The future has never looked as high-tech as today. The revolution that personal computers have created gives a clue as to what the handheld devices (like WAP phones, PDA (Personal Digital Assistants) and emerging broadband technologies) can do to everyday life. Therefore, WAP is an important development in the wireless industry because of its attempt to develop an open standard for wireless protocols, independent of vendor and airlink.
- ◆ In this chapter we have given an overview of WAP, its characteristic features, its benefits, applications, limitations, methods to overcome these limitations and the future trends in WAP.
- ◆ WAP is very much at its infancy. It is more or less at the evolutionary stage that the web was at about five years ago. But this is **today** and many companies are working hard on exciting new technologies to make WAP a success.

### 1.11 Brain Storm

1. What is WAP? Why we do need WAP?
2. What is WAP forum? What are the goals of WAP forum?
3. What are the characteristics and features of WAP?
4. What are the applications of WAP?
5. What are the limitations of WAP?
6. What are the future plans of the WAP form?

END

## Lecture 2

---

# WAP Communication Model

---

## Objectives

After completing this Lecture,  
you should be able to do the  
following

- ✔ WAP Communication Model
- ✔ WAP Client
- ✔ WAP Application Server

---

## Coverage Plan

---

### **Lecture 2**

- 2.1 Snap Shot
- 2.2 World Wide Web Architecture Overview
- 2.3 WAP Model
- 2.4 WAP Implemented Wireless Networks
- 2.5 WAP Communication Model
- 2.6 WAP Portals
- 2.7 Short Summary
- 2.8 Brain Storm.

## 2.1 Snap Shot

The objective is to give an overview of the WAP communication model. It also deals with the WWW architecture on which the WAP model is based. The components of the WAP communication model viz., WAP client, WAP Gateway and WAP server are discussed. It also compares the WAP model with that of the web.

### Introduction

In the previous chapter we saw an overview of WAP. We will now embark on a slightly more in depth description of its technical details. In this chapter we will get a bird's eye view of the WAP communication architecture.

WAP, as we have seen earlier, defines a communication protocol as well as an application environment. In essence, it is a standardized technology for cross-platform, distributed computing. The WAP protocols have been designed with web protocols in mind. WAP uses the underlying web structure. The goal of WAP is to render communication between content providers and mobile devices more efficient and less time consuming than what would have been possible if the web protocols themselves were used.

Before we describe the WAP communication architecture, we shall briefly look at how the World Wide Web (WWW) functions.

## 2.2 World Wide Web Architecture Overview

The WWW architecture provides a very flexible and powerful programming model. Applications and content are presented in standard data formats, and are browsed by applications known as *web browsers*. The web browser is a network application; i.e. it sends requests for named data objects to a network server and the network server responds with the encoded data using the standard formats.

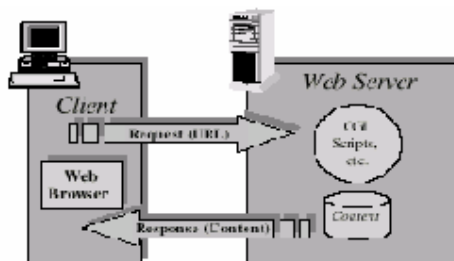


Fig 2.1 World Wide Web Programming Model

### Standard Components of WWW

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. This allows 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), JavaScript scripting language, 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 Transfer Protocol (HTTP).

This infrastructure allows users to easily reach a large number of third party applications and content services. It allows application developers to easily create applications and content services for a large community of clients.

### WWW Servers

The WWW protocols define three classes of servers:

#### Origin Server

The origin server is a server on which a given resource (content) resides or is to be created.

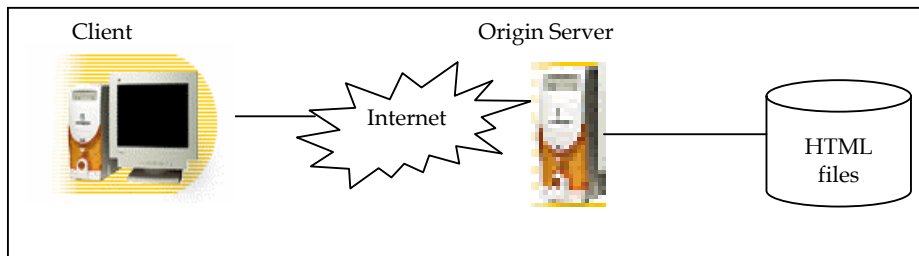


Fig 2.2 Origin Server

#### Proxy

Proxy is an intermediary program that acts both as a server and a client for the purpose of making a request on behalf of other clients. The proxy typically resides between clients and servers that have no means of direct communication e.g. across a firewall. Requests are either serviced by the proxy or passed on, with possible translation, to other servers. A proxy must implement both the client and server requirements of the WWW specifications.

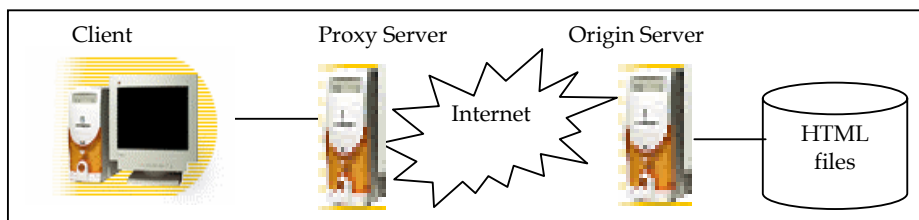


Fig 2.3 Proxy Server

#### Gateway



A gateway is a server that acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource. The requesting client may not be aware that it is communicating with a gateway.

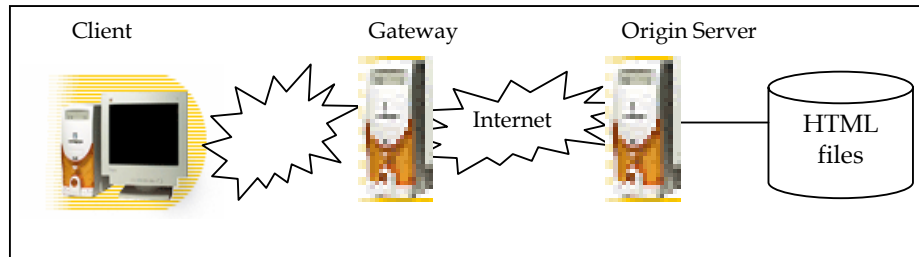


Fig 2.4 Gateway

We shall now look at how the WAP architecture has been developed on the well-known web structure. Since the WAP architecture has been designed to closely follow that of the web, the client/server paradigm used by the Internet has been inherited by WAP. The main difference, however, in the presence of the WAP gateway for translating between HTTP and WAP.

## 2.3 WAP Model

The WAP programming model is similar to the WWW programming model. This provides several benefits to the community of application developers, including a familiar programming model, a proven architecture, and the ability to leverage existing tools (e.g. web servers, XML, tools, etc.). Optimizations 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 content and applications are specified in a set of well-known content formats based on the familiar WWW content formats. Content is transported using a set of standard communication protocols based on the WWW communication protocols.

### Standard Components of WAP

WAP defines a set of standard components, similar to that of the web, 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 URLs are used to identify local resources in a device, e.g. call control functions.

#### Content typing

All WAP content is given a specific type consistent with WWW typing. This allows WAP server 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.

Before we look at the WAP communication model let us see how the WAP implemented wireless networks function. This would enable us to have a better understanding of the WAP communication model.

### 2.4 WAP Implemented Wireless Networks

In the wireless network world, the geographic region is divided into sections called **cells**. This is the reason why wireless networks are often called **cellular networks**. Every cell contains an antenna, also called a **Base Station**, which communicates with the mobile phones.

Base stations are grouped and controlled by a Base Station Controller that is attached to a Mobile Switching Center. The Base Station Controller has access to the fixed network as well as the entire wireless network; in this way, subscribers are able to communicate both with normal telephones and with mobile telephones.

While a mobile phone is switched on, it is communicating with a Base Station. Every time there is an incoming call, the Base Station attempts to contact the phone and, if the operation succeeds, the mobile user is informed of the incoming call. Every time the mobile user exits the area covered by a Base Station, during driving on the highway for example, a Hand Over procedure is initiated to pass the connection to the Base Station that covers the area you are moving into.

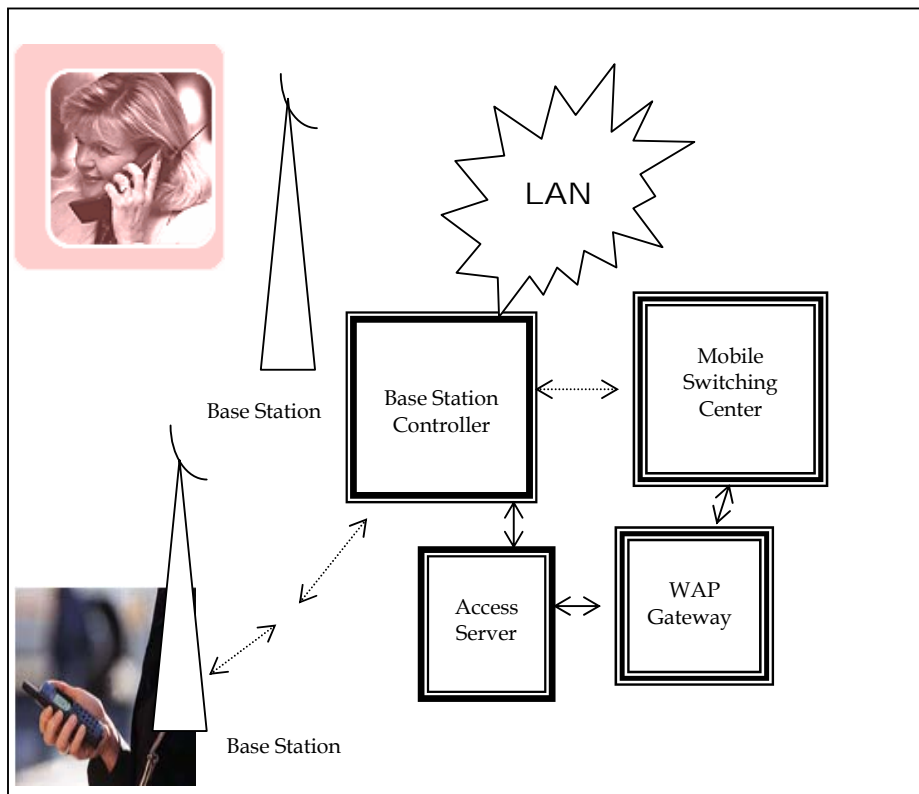


Fig.2.5 WAP Implemented Wireless Network

In the wireless networks where WAP is implemented, a WAP gateway is installed and connected to the wireless network LAN. A new phone number is defined and assigned to an Access Server. When the subscriber initiates a browsing session, a call is placed to that number.

The only purpose of the Access Server is to authenticate the subscriber willing to contact the gateway. It is connected to a database that stores the valid subscribers' numbers. Once the subscribers have been validated, they will be connected to the internal LAN, and allowed to communicate directly with the WAP gateway. We shall now look at how the WAP communication model works.

## 2.5 WAP Communication Model

In the introductory chapter we have already seen that the deployment of the WAP communication model depends on four components:

- ◆ Clients
- ◆ Gateways
- ◆ Servers
- ◆ Application

The function of the components in the WAP communication model can be explained in simple terms as follows:

The WAP client makes a request. The WAP request is routed through a WAP gateway which acts as an intermediary between the "bearer" used by the client and the computing network that the WAP gateway resides on (TCP/IP in most cases). The gateway then processes the request to the web server. Thereafter, it retrieves contents or calls CGI scripts, Java servlets, or some other dynamic mechanism from the server. The data is then formatted for return to the client. This data is formatted as Wireless Markup Language (WML), a markup language based directly on XML. Once the WML has been prepared (known as a deck), the gateway then sends the completed request back (in binary form due to bandwidth restrictions) to the client for display and/or processing. The client retrieves the first card off of the deck and displays it on the monitor.

The *deck of cards* metaphor is designed specifically to take advantage of small display areas on handheld devices. Instead of continually requesting and retrieving cards (the WAP equivalent of HTML pages), each client request results in the retrieval of a deck of one or more cards. The client device can employ logic via embedded WMLScript (the WAP equivalent of client-side JavaScript) for intelligently processing these cards and the resultant user inputs.

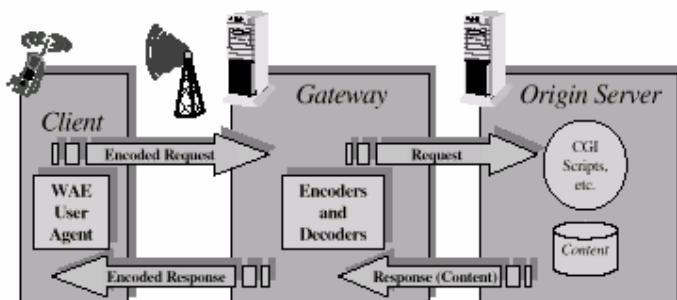


Fig 2.6 WAP Communication Model

We shall now look at the components of the WAP communication model one by one.

### 2.5.1 WAP Client

In a network environment, a client is typically the logical entity that is operated by the user and communicates with the server. In the WAP world, the client is the entity that receives content from the Internet via a WAP gateway. This is usually a WAP browser. The terms, 'WAP Client' and 'WAP Browser' are often used interchangeably. The WAP browser in the wireless terminal co-ordinates the user interfaces. It is an analogue of a standard web browser.



Fig 2.7 WAP Client/Browser

The WAP browser is software that runs on the WAP device. It displays the WAP content that it receives and decides on how to display it on the screen of the WAP device. It also provides the front-end through which the user can navigate the WAP application. The browser may be built into the phone or mobile device, or into the SIM card that the device contains. WAP browsers also are referred to as micro-browsers.

WAP browsers that run on PCs directly or through a web browser by means of a plug-in are also available. Often a PC WAP browser is included in the downloadable SDK (Software Development Kit) available from various companies and it can be easier to use a PC WAP browser than a real device. Moreover the testing of applications throughout their development can be made easier by PC WAP browsers.

Shown below is a list of some the WAP browsers that are currently available:

Browser	Description	Available from
Nokia	Several browsers are available with the Nokia toolkit.	<a href="http://forum.nokia.com/">http://forum.nokia.com/</a>
UP.Browser	Available with the Phone.com toolkit.	<a href="http://updev.phone.com/">http://updev.phone.com/</a>
Ericsson	There are different browsers available with Ericsson Development toolkit	<a href="http://ericsson.com/">http://ericsson.com/</a>
Motorola	Motorola have an offering similar to the other companies with their toolkit.	<a href="http://motorola.com/">http://motorola.com/</a>
Gelon.net	At this site, a WAP browser is run on-line with in the web browser on the PC.	<a href="http://www.gelon.net/">http://www.gelon.net/</a>

Table 2.1 List of WAP Browsers

The WAP client comprises of:

- Wireless Application Environment User Agent (WAE User Agent)
- Wireless Telephony Application User Agent (WTA User Agent )
- WAP Stack

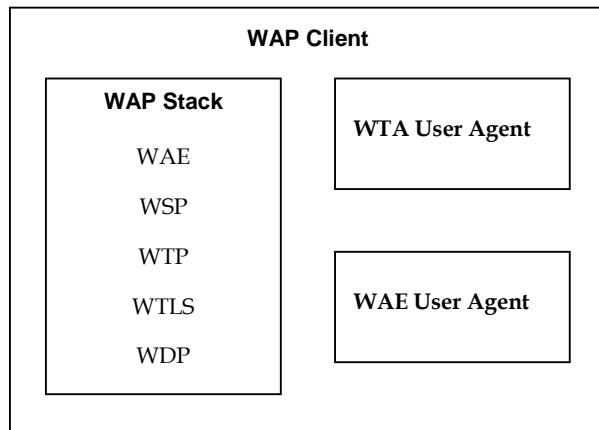


Fig 2.8 Components of the WAP Client

Let us now see how these components of the WAP client function.

#### WAE User Agent

The WAE User Agent (Wireless Application Environment User Agent) is the micro-browser that renders the content for display. It receives the compiled WML, WMLScript, and any images from the WAP gateway, and executes or displays them on the screen. Even if the implementation details are left to the vendor, the browser must implement all the functionality provided by WML and WMLScript. It must have interaction with the user, through text inputs, and error or warning messages.

#### WTA User Agent

The WTA User Agent (Wireless Telephony Applications User Agent) receives compiled WTA files from the WTA server and executes them. The WTA User Agent includes access to the interface of the phone, and network functionality such as number dialing, call answering, phonebook organization, message management and location indication services.

#### WAP Stack

The WAP stack implementation allows the phone to connect to the WAP gateway using the WAP protocols. Let us reiterate three important concepts viz., Content/Application server, Proxy and Gateway, before we embark on an in-depth description of the WAP gateway and its functionality.

- **Content/Origin/Application Server:** This is the element in the network where the information or web/ WAP applications reside. (Web servers belong to this category.)
- **Proxy:** This is an intermediary element, acting both as a client and a server in the network. It is located between the client and the origin servers. The clients send requests to it and it retrieves and caches the information needed by contacting the origin servers.

- **Gateway:** This is an intermediary element that is usually used to connect two different types of network. It receives requests directly from the clients as if it actually were the origin server from which the information is to be retrieved. The clients are usually unaware that they are communicating with the gateway.

### 2.5.2 WAP Gateway

WAP gateway is the element that sits (logically) between the WAP device and the origin server. This is an intermediary element that is usually used to connect two different types of networks. It receives requests directly from the clients as if it actually were the origin server that the clients want to retrieve the information from. The clients are unaware that they are communicating with the gateway. It usually resides within the operator's network.

WAP gateway is responsible for translating WSP requests into appropriate HTTP requests for delivery over an intranet or the Internet to the designated origin server. The WAP gateway is also responsible for translating HTTP responses to appropriate WSP responses for delivery over the wireless network to the requesting client device.

#### Is WAP Gateway a Proxy or a Server?

The three terms WAP gateway, proxy and server are often used interchangeably and wrongly so. On the contrary, in the world of networks these three elements are quite different logically and they have different functionalities as well:

The element used in the WAP architecture, which we earlier defined as a WAP gateway, is actually a **proxy**. It is used to connect the wireless domain with the Internet. However, it contains *protocol gateway* functionality plus *encoder/decoder* functionality.

The products at present on the market create the confusion of these terms. What is typically offered today is a mixture of all of the servers described above. It logically belongs to the proxy category but has gateway functionality. In addition to this it is equipped with server functionality. In other words, it can run server-side scripts, Java servlets and do all the things that a standard web server can do.

The rule to overcome this confusion is to generally consider a WAP gateway and a WAP proxy as the same thing. It is a good idea to avoid the term 'WAP server'. WAP server is usually a WAP gateway with server functionality added. It is probably better to refer to such an element as a "combined application server and gateway".

In the diagram below we illustrate the use of a WAP proxy/gateway. We will move on to consider exactly what the gateway does in the next section.

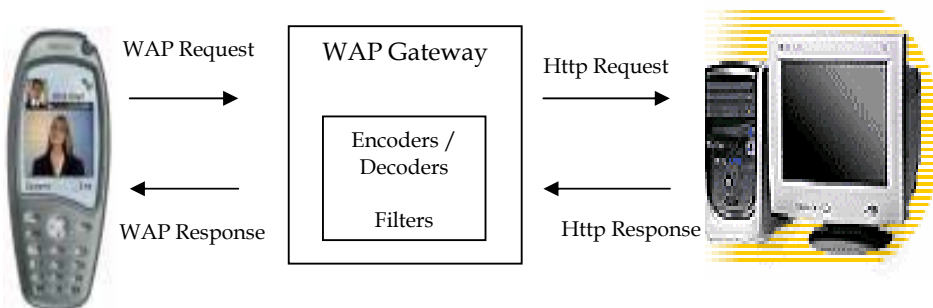


Fig 2.9 WAP Proxy Gateway

### WAP Gateway Functionality

The WAP content types and protocols have been optimized for mass market, hand-held wireless devices. WAP utilizes proxy technology to connect between the wireless domain and the WWW. The WAP proxy is comprised of the following functionality:

Whenever a WAP session starts, the following steps are executed.

- A connection is created via WSP (Wireless Session Protocol) between the mobile device and the WAP gateway, which we assume, is present in the operator network.
- As the user enters the address of a WAP site, the gateway is sent a request from the device's microbrowser using WSP. WSP is the WAP protocol in charge of starting and ending the connections from the mobile devices to the WAP gateway.
- The gateway translates the WSP request into an HTTP request and sends it to the appropriate origin server.
- The origin server sends back the requested information to the gateway via HTTP.
- The gateway translates and compresses the information and sends it back to the microbrowser in the mobile device.

We shall now look at the components of the WAP gateway and their functions.

The WAP proxy is comprised of the following:

What are the functionalities of a WAP Gateway?

- Implementation of WAP protocol stack layers
- Access Control
- Protocol conversion : WSP                      HTTP
- Domain Name Resolution     $\longleftrightarrow$
- HTML to WML conversion
- Encoding of WML content
- WMLScript compilation
- Security
- Provide caching for frequently accessed content

**Proxy gateway**

The protocol gateway translates request from the WAP protocol stack (WSP, WTP, WTLS, and WDP) to the WWW protocol stack (HTTP and TCP/IP).

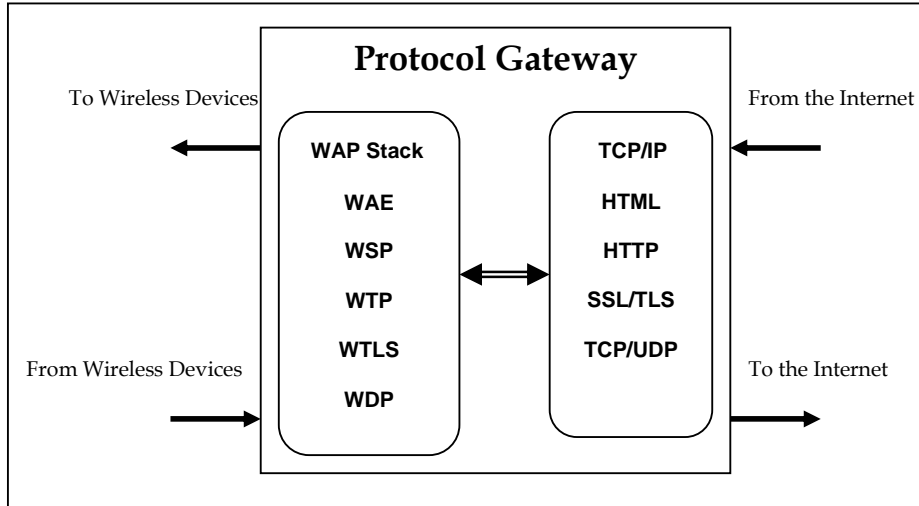


Fig. 2.10 WAP Protocol Gateway

**Content Encoders and Decoders**

The content encoders translate WAP content into compact encoded formats to reduce the size of data to be sent over the network. This infrastructure ensures that mobile terminal users can browse a wide variety of WAP contents and applications, and that the application author is able to build content services and applications that run on a large base of mobile terminals. The WAP proxy allows content and applications to be hosted on standard WWW servers. It also allows content and applications to be developed using proven WWW ethnologies such as CGI Scripting.

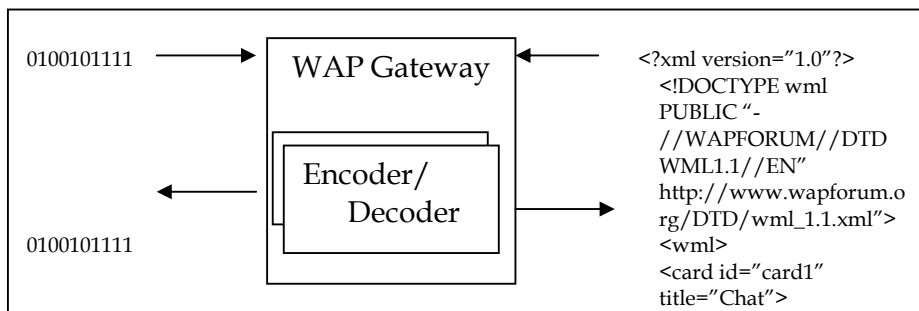


Fig. 2.11 WAP Gateway Encoder/Decoder functionality

The gateway part of the WAP proxy takes care of translating all the requests that are sent and received to the protocol that the origin server is using (HTTP). The content provider sends its content using HTTP to the gateway. It then forwards all the content received to the WAP devices, using the WAP protocols. This is illustrated in the diagram below.

The coder/decoder functionality within the gateway is used to convert the WML and WMLScript content going to and coming from the client into a form that is optimized for low-bandwidth networks.



Translation of encrypted data takes place in the memory of the gateway. No unencrypted data is ever stored on a secondary storage medium since this would create crucial security problems.

Content belonging to a non-secure session is cached on the storage media of the gateway, reducing the processing time and the resources required when someone else requests the same content.

Another service that the coder/decoder functionality can provide is the translation of HTML or text to WML. The HTML to WML translator, when present among the gateway features, is there mainly for giving us compatibility and to reassure that information important call be retrieved.

The WAP gateway is also connected to the WTA server, present in the operator network, that provides the interface for accessing some of the network services the operator wants to provide.

### **Who Provides the WAP Gateway?**

The wireless network operator always provides the gateway. Furthermore, WAP gateways are designed for installation and use in an operator's network and their use in another environment generates some difficulties, for example the adjustment of the gateway for the different bearers and handsets.

Instead, many companies wish to install their own gateway, ensuring that their content can be sent securely to the mobile phones authorized to access it. This avoids completely the Internet side of WAP. But the installation of a full-fledged WAP gateway will take time, effort, and usually cost a lot of money. Another problem is that the customers who wish to use it will need to change the way their phone is configured completely, and then change it back, if they are to use the original gateway provided by their mobile operators their phone. This will mean changing the IP address of the gateway, phone number and possibly the user name and password. However, this is an exception rather than a rule.

What are the currently available gateways?

*Currently available gateways are:*

- *Nokia WAP server - for General Solutions for Mobile Phones (GSM) network*
- *Ericsson WAP gateway - that will fit the GSM network.*
- *Ericsson Jambala - product aimed at the TDMA network.*
- *Motorola Exchange (MIX) - will adopt to different types of networks.*
- *Phone.com UP.Link - Gateway package that fits diverse types of network*

### **2.5.3 WAP Application Server**

WAP Application/Content Server denotes the element that hosts the Internet content that is sent to clients when they make a request to it. A WAP server is usually just a WAP application server with gateway functionality added. It will provide all the services that a normal origin server provides, but it will also act as a WAP gateway. The WAP application server stores WML, WMLScript and WBMP (Wireless Bitmap) image files.

While the nominal use of WAP will include a WAP client, WAP proxy, and web server, the WAP architecture can quite easily support other configurations. It is possible to create an origin server that includes the WAP proxy functionality. Such a server might be used to facilitate end-to-end security solutions, or applications that require better access control or a guarantee of responsiveness, e.g., WTA.

Now that we have seen how the various components of the WAP communication model work, let us see how these components enable a wireless device user to browse WAP sites.

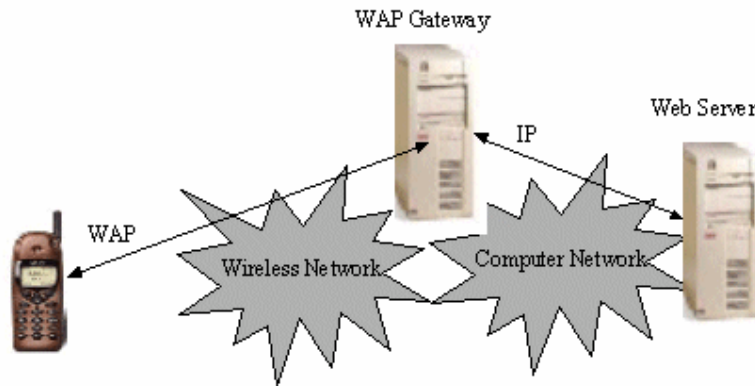


Fig 2.12 WAP Application Server

### Browsing WAP sites and the Latency constraint

Once a request is made from a mobile device, the browser contained in the phone will automatically send the subscriber's details to the gateway together with a user name and password. The gateway checks them against a database as would a dialup connection to the Internet in a traditional PC session.

#### How Do WAP Phones Connect To A WAP site?

Each WAP phone needs to connect to a WAP gateway. Usually, the mobile phone company will have a gateway, but other companies are coming with their own ones. Some mobile phone companies will only let you visit sites that they approve of, which is why independent gateways are a good idea. But back to technicalities... when you visit a WAP site, your phone sends a request to the WAP gateway. That then sends a request to the WAP site for the page you want to view, when the WAP gateway has a copy of the page, it compresses it and sends it back to your phone, where it is expanded and displayed. As with the normal Internet, the speed is often down to how busy the site is, and how busy the Internet is in general. The gateways also vary in how good they are at compressing a page. As some phones might reject a WAP page that is getting close to the size limit, you can sometimes end up with the odd situation that a page will work on one gateway or phone, but not on another.

The login procedure that takes place at the gateway will cause the first, sometimes long, wait. However, during the login procedure the gateway doesn't access an external application server, since the WAP portal is stored on the gateway itself or on an application server located in the internal operator network. Once we are logged in, we can start the WAP browsing.

A WAP browser also has an associated 'home page' deck, determined by the service provider, which is loaded into the microbrowser after the user has been authenticated.

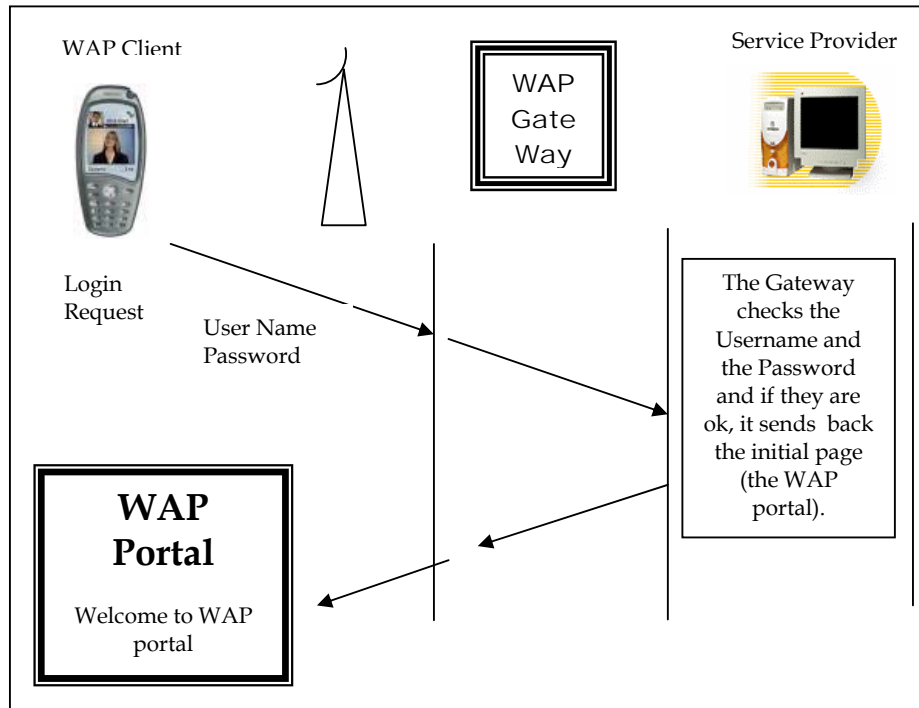


Fig 2.13 Browsing of WAP Sites

## 2.6 WAP Portals

A portal provides lists of links, making it easier to navigate to different sites. The portal interface is provided to users, so that they can define, organize and personalise the services to be accessed on the mobile device. By using a desktop PC browser, users can log on to the portal and add their favorite links to be accessed via the mobile device.

WAP portals supply us with two useful advantages:

- Users do not have to browse and search for the sites that provide them with a certain service. Users would find search already grouped by the service they are provided. This, in case of WAP, saves a lot of time, since WAP browsing can be rather time consuming, since there is no proper keyboard or mouse available and the speed of the connection is pretty slow.
- Users do not have to remember or to write down the name of the site that they want to visit, since the user can be automatically redirected to the site by clicking on one of the links on the portal.

WAP portals are offered by the network operators, but also sometimes by content providers. The address of the WAP portal owned by the network operator is normally specified in the settings of the WAP enabled device.

But while browsing using a mobile device the latency constraint should be taken into account. The latency associated with browsing via a mobile device is strongly dependent on the type of bearer used. The more advanced the bearer is, the faster the connection with the WAP gateway, and consequently the less time required to transfer data to and from the gateway.

### **Bearer Service**

Mobile phones can use different ways to communicate with the wireless network. A bearer service is the method that is used to accomplish this. Let us now consider the General Solution for Mobile (GSM) phones network. General solution for mobile phones (GSM) is the terminology that is used globally to denote the methods adopted to accomplish communication between a mobile and the wireless network. GSM gives the user the choice from a diverse selection of bearers, for example Short Message System (SMS), Circuit Switched Data (CSD) or the latest General Packet Radio Switching (GPRS).

When we run WAP over an SMS bearer, the WAP Gateway has to divide the content addressed to the WAP device into packets, each one containing at most 160 bytes. The device then has to reassemble the messages it receives to decipher the content. This procedure is very time consuming and accounts for the fact that SMS is the slowest bearer amongst the possible choices.

With CSD, a data connection is established between the WAP device and the WAP gateway. The speed of the connection is 9600 bps, providing a faster medium for data exchange compared to SMS. CSD is similar to how a modem in a computer communicates with the Internet Service Provider (ISP).

GPRS is a new technology for data transfer within the GSM wireless network. It has been designed to be an upgrade of GSM, supplying more bandwidth for wireless communication. The main slogan related to GPRS is "always connected, always online". With GPRS there will be a minimal connection setup procedure that will occur when the mobile phone is switched on. After that the mobile user will always be online, ready to receive and send data in less than one second. In the GPRS Technology, the volume of data that they send and receive, and not, as it is now with the GSM data connection, by the length of time for which they are connected will charge the subscribers. In the future, GPRS will provide speeds of up to 171.2 kbps, which will be a step forward for WAP and m-commerce in general.

Moreover, while dealing with voice or data communication in a wireless network, factors such as obstacles that disturb the transmission have to be taken into consideration. This limits the bandwidth and hence increases the already high latency associated with wireless networks.

Although the WAP communication model has been developed on the lines of the WWW model, there are major differences between the two. We shall now look at the differences.

### **WAP Vs WEB**

The first major difference between the Web and WAP is with respect to the services offered. While there are common services such as mail access and reading the latest news, there are many web services that may not be possible to port to mobile devices. Instead there are opportunities or many new types of services, such as location based information, which are available only to mobile devices.

Another major difference between WAP and the Internet at the moment lies in the manner of browsing. When surfing on the Internet we are most of time not concerned about the length of time for which we are connected, since nowadays the cost of an Internet call is quite cheap. On the other hand, a WAP user, for whom the connection costs are higher, will be more concerned with the issue of cost. Inputting data on a phone is relatively difficult and so the connection time increases. Therefore, WAP users will want faster access to services guaranteed to be useful to them than their web counterparts.

The typical WAP user will not necessarily be a computer or a web user, so applications developed for the WAP devices should not depend on the computer literacy of the user. WAP applications must therefore be as simple and as user-friendly as possible.

The user interface influences the usability of any given web or WAP service, though this is of higher importance in the case of WAP. With such small screens, information needs to be displayed in a neat and clear way in order to stimulate people into using the service.

## 2.7 Short Summary

- ◆ In this chapter we have discussed the WAP communication model. The end-to-end components of the WAP communication model can be summarized as follows:
  - A client device capable of requesting and rendering WAP content.
  - A wireless network employing WAP.
  - A WAP – capable gateway capable of translating WAP protocol requests into corresponding requests over the Internet and translating responses from the Internet into corresponding responses over the WAP protocols.
  - The Intranet or an Internet using TCP/IP-based protocols and possibly having one or more protocol gateways and web/HTTP proxies.
  - An origin (Web) server that can generate requested content.
- ◆ With these components the functioning of the model can be summed up as follows: The client makes a request. This request is received by a WAP gateway that then processes the request and formulates a reply using WML. When ready, the WML is sent back to the client for display. As mentioned earlier, this is very similar in concept to the standard stateless HTTP transaction involving client Web browsers.
- ◆ We have also seen how WAP sites can be browsed using the handheld WAP devices.
- ◆ The major differences between the WAP model and the web model have also been discussed.

## 2.8 Brain Storm

1. Compare the WAP architecture with the Web architecture.
2. What are the components of WAP? Discuss the various components.
3. Explain the WAP communication model.
4. Why is a WAP gateway a proxy?
5. What are the functionalities of the WAP gateway?
6. Mention some WAP browsers that are currently available.
7. Mention some WAP gateways that are currently available.
8. Discuss about the WAP protocol and the WAP application server.
9. What are the bearer services that are available currently?
10. Write the expansions of the following:
  - a) WAP
  - b) GPRS
  - c) GSM
  - d) SMS
  - e) CSD

## Lecture 3

---

# WAP Protocol Architecture

---

## Objectives

After completing this Lecture, you should be able to do the following

- ✔ Wireless Application Environment (WAE)
- ✔ Wireless Session Protocol (WSP)
- ✔ Wireless Transport Protocol (WTP)
- ✔ Wireless Transport Layer Security (WTLS)
- ✔ Wireless Datagram Protocol (WDP)

---

## Coverage Plan

---

### Lecture 3

- 3.1 Snap Shot
- 3.2 Protocol And Layers
- 3.3 WAP Stack And Web Stack
- 3.4 Wireless Application Environment (WAE)
- 3.5 WAE Useragent
- 3.6 WTA
- 3.7 Wireless Session Protocol (WSP)
- 3.8 Wireless Transport Protocol (WTP)
- 3.9 Wireless Transport Layer Security (WTLS)
- 3.10 Wireless Datagram Protocol (WDP)
- 3.11 Short Summary
- 3.12 Brain Storm

### 3.1. Snap Shot

WAP is an emerging industry standard which aims at providing wireless internet services and information access, using handheld devices having limiting display capabilities and over limited bandwidth wireless channels. The acceptability of this effort can be gauged by the fact today 95% of the global handset makers, are members of the WAP forum. The WAP forum defined the WAP protocol. The WAP protocol mimics that of the World Wide Web.

In this chapter we shall discuss about the constituent layers of the WAP protocol and their functions

Before we look at the details of how the WAP protocols are structured, let us first briefly examine the definitions of **protocol**, **layer** and **protocol stack**.

### 3.2 Protocol and Layers

#### Protocol

It is important to speak a common language that allows others to understand what you are saying. The problem of a common language also arises in telecommunication networks. There are many different devices, and networks, and to allow them to communicate with each other, a common language must be provided. Protocols are the answer to this problem.

A protocol defines the type and the structure of the messages that two devices have to use when they are communicating with each other.

There are a lot of different kinds of protocols, from very simple ones, to very elaborate ones, but they all have the same property in common: they allow computers to communicate with each other.

#### Layers

Since the protocols are functionally and logically divided into different groups of functionality, they are also physically framed into layers, each one providing a specific service to the next layer. One layer may provide methods to send bits down a physical cable; another may supply methods to establish a connection.

#### Protocol Stack

The protocol stack is the set of all the layers that compose the set of protocols.

#### WAP Protocol Stack

WAP stack has been derived from, and has inherited most of the characteristics of the ISO OSI reference model [ISO7498]. It is maybe useful to compare the WAP layers with the web protocol stack used in the web model. There are strong similarities between the two. But there are differences too.

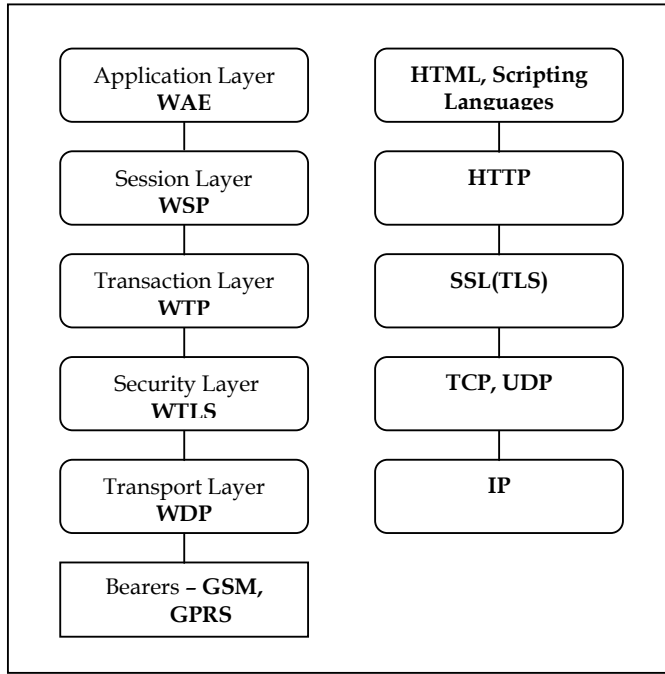
### 3.3 WAP Stack Vs Web Stack

The main difference between the two is in the number of layers. WAP has just five layers, while the OSI model has seven layers.

The other major difference is the compactness and lightness of the WAP protocol. It must be borne in mind that the Internet protocols introduce high overheads and are not effective when used in the low bandwidth and high latency networks such as the ones used with mobile phones.



The application layers of both WAE and that of the web stack provide a markup language and a scripting language for the development of applications. The figure given below shows the WAP stack viz -a - viz the web stack. The figure is only illustrative and it is not intended to imply a direct one-to-one correspondence between the protocols.



Although the WAP protocol mirrors Internet standards to a great degree, it obstinately manages to be almost completely incompatible with them. Because of this incompatibility, WAP devices cannot communicate directly with WWW servers. The WAP protocols must first be translated from their WAP formats to the protocols used by the WWW. This is why every WAP device needs to communicate with a WAP gateway in order to request WML pages (or “decks” as they are referred to in WAP) on a web server. The WAP gateway’s job is to translate the WAP binary protocols into the HTTP text protocol that the World Wide Web servers speak.

<b>Web Model</b>	<b>WAP Model</b>
TCP & IP are the transport and network layer protocols providing end-to-end communication.	A lightweight protocol stack WDP - WTP - WSP suited for wireless networks is used between the device and the WAP gateway. TCP/IP is used between the gateway and the origin server.
HTTP is the application layer protocol used.	WSP is the application layer protocol used, which can be considered as a binary form of HTTP.
The protocol headers and the content markup language tags are in a textual format. Typically these are human readable.	The protocol headers and the content markup language tags are enclosed in a compact binary form between the gateway and the device.
An HTTP proxy can be used as an intermediary.	A WAP gateway, which is comparable to an HTTP proxy, is always used.
Client-side scripting embedded in the content travels in the source code before being interpreted	Client-side scripting (WMLS) is in a separate file on the origin server. When there is a request for a script resource, it is delivered to the WAP gateway where it is compiled and byte code is generated, before it is transmitted to the user agent in the WAP device. The user agent,

by the browser.	therefore, needs to have a bytecode interpreter and a virtual machine environment.
Browsers support large number of image formats and other multimedia formats either directly or indirectly through plugging.	Mobile user agents currently support a much smaller number of multimedia formats because of the limited capabilities of the devices themselves and because of the available bandwidth.. In fact the black & White bitmap format known as WBMP is the only image format that is currently supported.

The WAP protocol suite is built upon hierarchical service of layers, some of which are optional. In the next few pages, we will look at how the WAP protocol is structured.

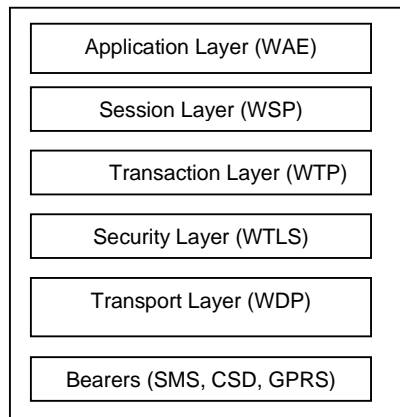


Fig. WAP Protocol Stack

The WAP Stack, illustrated in the above diagram, has five different layers;

### Application Layer

Wireless Application Environment (WAE) provides an application environment intended for the development and execution of portable applications and services.

### Session Layer

Wireless Session Protocol (WSP) supplies methods for the organized exchange of content between client/server applications.

### Transaction Layer

Wireless Transaction Protocol (WTP) provides different methods for performing transactions, to a varying degree of reliability.

### Security Layer

Wireless Transport Layer Security (WTLS) is an optional layer that provides, when present, authentication, privacy and secure connections between applications.

### Transport Layer

Wireless Datagram Protocol (WDP) is the bottom layer of the WAP stack, which shelters the upper layers from the bearer services offered by the operator.

We will now consider each of the WAP stack layers in detail.

### 3.4 Wireless Application Environment (WAE)

The application layer of WAP provides an environment that includes all the elements related to the development and execution of WAP. WAE provides industry-wide standards and specifications for the development of applications and services that operate over wireless communication networks. The WAE is based on the World Wide Web (WWW) technologies and philosophies and provides a framework for application development on a wide variety of wireless devices. WAE enhances some of the WWW standards in way that reflects the device and network characteristics. Careful attention has been paid to the memory and CPU processing constraints that are found in mobile devices. Support for low bandwidth and latency networks is included in the WAE architecture as well.

The main objectives of WAE are to facilitate application development on smaller screens, support for interactive applications, support for security and access control while accessing third party content. WAE consists of two different user agents located on the client side:

- ◆ WAE User Agent
- ◆ WTA User Agent

### 3.5 WAE User Agent

In the WWW model, the user agent, also referred to as client, makes requests for a set of named data object (content) to an origin server. The origin server responds with the requested data in a standard format called HyperText Markup Language (HTML), known to the user agent. The named data objects include text in HTML, scripts in scripting languages such as JavaScript, bitmap images and a large number of other formats. These standard content formats are known and supported by the standard user agents or clients. The WWW also specifies a standard protocol named Hyper Text Transport Protocol (HTTP), using which the clients communicate with the server.

WAE user agents, like their WWW counterparts, are designed to understand text in encoded Wireless Markup Language (WML) and compiled Wireless Markup Language Script (WMLScript).

WAE User Agent adopts a model, which is very close to that of the WWW model. The main building blocks of the WAE User Agent are the following:

- A lightweight markup language - WML
- A lightweight markup language scripting - WML script

#### Wireless Markup Language (WML)

WML like HTML is a tag-based document language. It is based on HTML and is influenced by Handheld Markup Language (HDML), supported by phone.com. WML is optimized for specifying presentation and user interaction on wireless devices with limited capabilities, such as screen sizes, bandwidth and keyboard inputs.

WML incorporates support for text, formatted text with emphasized elements, line-break models, tab elements, and bitmap images. WML, however, does not support style sheets, frames and colors like HTML. WML delivers the application to the client as a deck of cards. A deck has one or more cards and each card can be labeled. Each card can display text, images, hyper-links and input fields. It essentially

contains structured content and navigation specification. The requests for user inputs in each card can be of text type, multiple-choice selection or buttons. The client, depending on the display capabilities of the client renders the WML content. The tags of WML content are encoded during the communication between the client and server. The encoding helps to reduce the content transferred on the low bandwidth wireless channels and also helps in convenient navigation of the Web content.

### **WML Script**

WML Scripts is a lightweight procedural scripting language and is loosely based on a subset of JavaScript, the WWW scripting language. WML Script is a weakly typed language and supports the basic data types: boolean, integer, floating-point, string and invalid. It supports assignment operations, logical operations and comparison operations. WML Script supports different types of functions: those defined in the local script, those defined in the external script and those defined in the standard library. The scripts are compiled into machine independent, compressed code, to make the transmission of script more efficient. An operating environment executes the compiled byte-code, which is similar to Java Virtual Machine (JVM), supported by the client. WAE specifies a set of standard library functions, thus providing for device-independent application development environment.

## **3.6 Wireless Telephony Application (WTA)**

In addition, WAE specifies Wireless Telephony Application (WTA) specifications, which provides a set of functions that allow control over the client device, assuming it is a telephone device. WTA merges the features and services of data networks with the services of the voice network. WTA framework aims at providing advanced telephony services, which are well integrated into the client and provides a consistent user interface. WTA allows access to and interaction with mobile telephone features (e.g. call control) as well as other calendar applications. Hence, WTA allows real-time processing of events important to the end-user while browsing.

## **3.7 Wireless Session Protocol (WSP)**

The Wireless Session Protocol enables services to be exchange data between applications in an organized way. Session services are those functionality that help in setting up a connection between a client and a server. A service is delivered through the use of the primitives it provides. *Primitives* are defined messages that a clients sends to the server to request a service facility. WSP supports two different services:

- ◆ Connection oriented Service
- ◆ Connectionless Service

### **Connection Oriented Service**

The connection-oriented session service provides facilities used to manage a session and to transmit reliable data between a client and a server. This service operates over the Wireless Transaction Protocol (WTP). Most of the facilities provided by the connection-oriented session service are confirmed, meaning that the client can send *Request primitives* and receive *Confirm primitives* and the server can send *Response primitives* and receive *Indication primitives*.

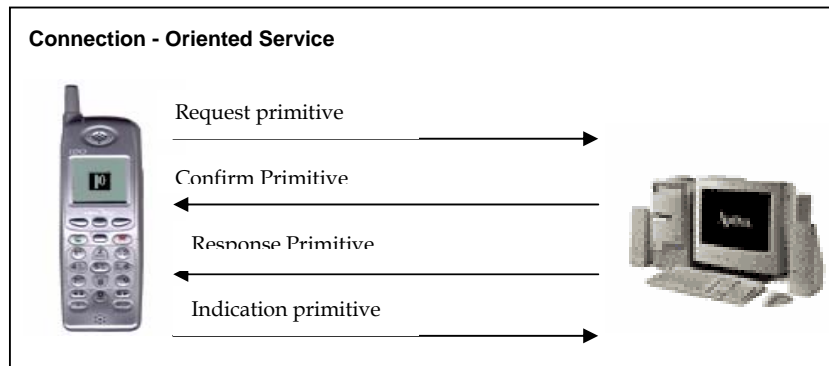


Fig 3.1 Connection - Oriented Service

### Connectionless Service

This operates directly over the Wireless Transport layer (WDP). The connectionless session services provide only non-confirmed services. Here when the client sends request primitives confirm primitives are not received. Similarly when the server sends response primitives it does not receive an indication of primitive.

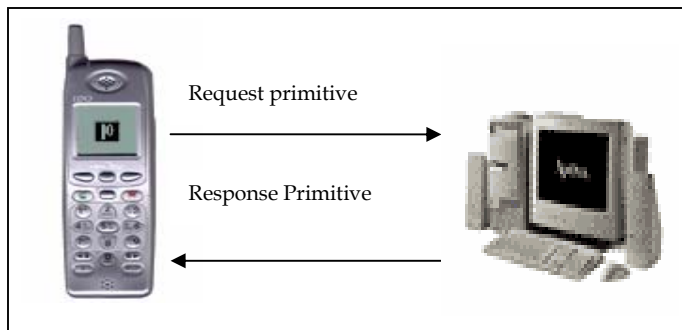


Fig 3.2 Connectionless Service

The core design of WSP is based on HTTP1.1 and it supports all the methods defined in HTTP along with extended requests for capability negotiations. WSP is based on request-reply model, each consisting of a header and a body. The header is metadata, i.e. data about data, and contains information about the particular request or response as name-value pairs. The content on the other hand contains the actual WML text in tokenized form, compiled Scripts, or images WSP optionally supports asynchronous requests from the client. i.e., a client can send multiple requests to the server simultaneously without waiting for the response. WSP similarly provides for the delivery of composite objects i.e. multi part data containing several header-data pairs, as one deck. This eliminates the need for succeeding request-replies, thus making the rendering at client faster by reducing the round trip delays. In some ways WSP is basically a binary form of HTTP. The binary transmission of data is a necessary adaptation made for the narrow bandwidth of mobile networks.

WSP extends the functionality of HTTP by specifying three types of “data push” mechanisms:

- confirmed push within the context of an existing session
- non-confirmed data push within the context of an existing session
- non-confirmed data push without an existing session.

The “data push” capabilities supported by WSP provide mechanisms to broadcast data such as stock quotes, weather information asynchronously on to the client devices. The push mechanisms have been discussed in Chapter 9.

### 3.8 Wireless Transport Protocol (WTP)

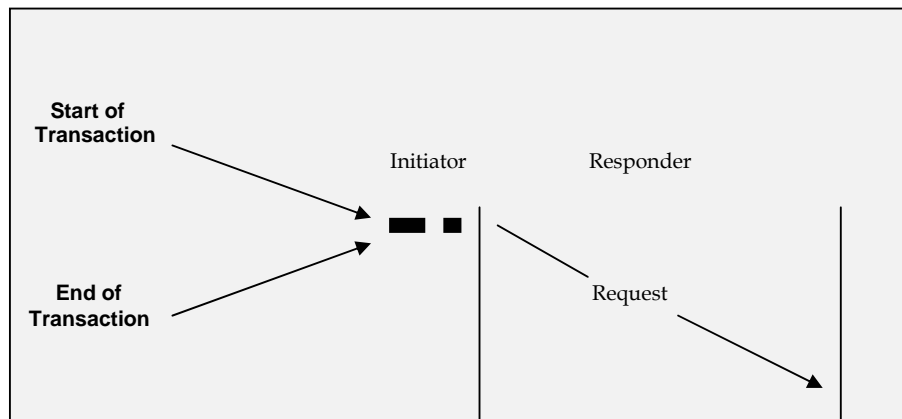
WTP is responsible for providing reliable transmission of WSP data packets between the client and server over a wireless medium. WTA draws from the concept of TCP or UDP. The current specification of WTP allows for both reliable (TCP - like) and an unreliable (UDP-like) communication. When the connection is unreliable however, WSP is responsible for retransmission to make the connection reliable.

In particular, three different classes of transaction services are supplied to the upper layers:

- Unreliable requests
- Reliable requests
- Reliable requests with one result message

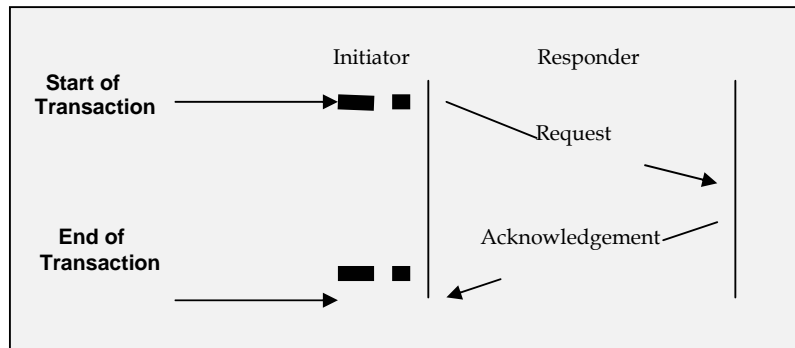
#### Unreliable Request

The initiator (in this case a content server) sends a request to the responder (the user agent) who does not reply with an acknowledgment. The transaction has no state and terminates once the invoked message is sent:



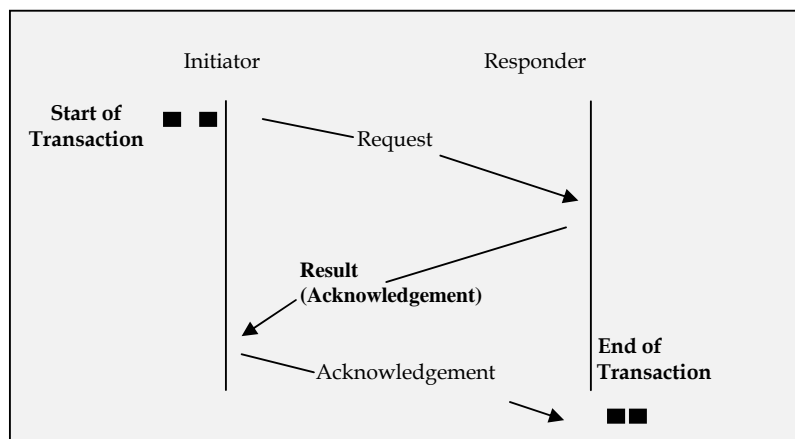
#### Reliable Request

The initiator sends a request to the responder who acknowledges it. The responder stores the transaction state information for some time, so that it can re-transmit the acknowledgement message if the server requests it again. The transaction ends at the initiator when the initiator receives the acknowledgement message:



### Reliable Request with One Result Message

The initiator sends a request to the responder who implicitly acknowledges it with a result message. The initiator then acknowledges the result message, maintaining the transaction state information for some time after the acknowledgement has been sent, in case it fails to arrive. The transaction ends at the responder when it receives the acknowledgement message.



WTP is responsible for packet segmenting and reassembling. It is also responsible for acknowledgement of packets received and retransmission of ones lost, unacknowledged, or corrupted. Packets are numbered in a way that complies with an at-most-once policy. This helps to ensure that a retransmitted packet is not handled as a new packet; thereby, eliminating duplication and other related problems.

WTP like the other layers in WAP, is optimised to adapt to the small bandwidth. It attempts to reduce the total number of replayed transactions between the client and server.

### 3.9 Wireless Transaction Layer Security (WTLS)

WTLS is the solution to the security issue, provided by the WAP Forum. WTLS is an optional layer and is based on TLS (Transport Layer Security)v1.0, which in turn is based on SSL (Secure Socket Layer)v3.0, which are Internet protocols. WTLS operates over the WDP.

Since, WTLS is an optional layer in the WAP stack. This means that security in WAP is only available on demand and is not a built in feature of the WAP architecture. Hence, the information travelling to and

from the WAP gateway is normally not encrypted, unless we use SSL connections to communicate between the origin servers and the gateway

WTLS, when present, provides the same grade of security that is supplied by SSL 3.0. It provides services that ensure *privacy, server authentication, client authentication and data integrity*. These terms are briefly described below.

**Privacy**

Guarantees that the data sent between the server and the client is not accessible to anyone else. No one can read the unencrypted message, although they can see the encrypted message.

**Server Authentication**

Ensures that the server really is who it claims to be, and that it is not an imposter.

**Client Authentication**

Provides a way for the origin server to limit the access to the content it provides. Only subscribers who are recognized as trusted ones can gain access to the site.

**Data Integrity**

Takes care that no one can alter the content of a message that is being transmitted between server and client.

In the above diagram, a standard SSL session is opened between the web server and the WAP gateway and a WTLS session is initialized between the gateway and the mobile device. The encrypted content is sent through this connection from the server to the gateway, which translates it and sends it to the mobile phone.

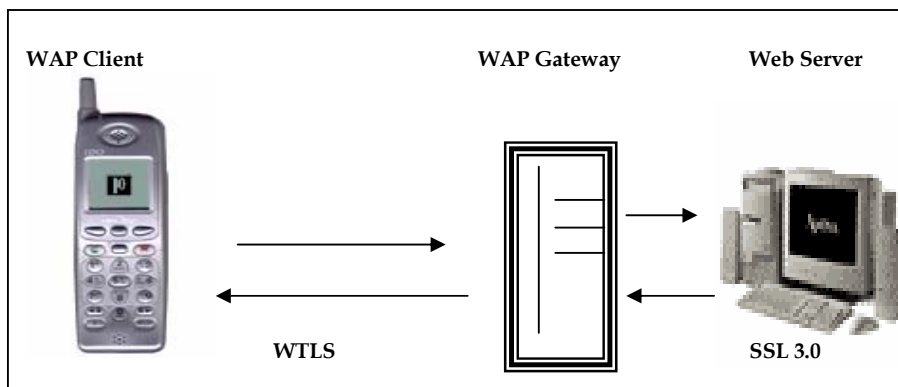


Fig 3.2 Data transmitted between Server and Client

The translation between SSL and WTLS takes place in the memory of the WAP gateway. It is important that unencrypted information is not stored anywhere in the gateway, since this defeats all the security measures that are used to protect the stored data from being seen by unauthorized people.

To use a secure connection the origin server has to be installed as if a secure connection over the Internet is being setup; the gateway will take care of matching the SSL connection to a WTLS one.



WTLS empowers the SSL protocol by adding effective features such as datagram support, optimized handshake and dynamic key refreshing.

Even though WAP gateways are provided with many features to supply the maximum level of security, there is still a lot of concern surrounding the WAP security solution. Banks and other companies that really have to protect their data, still prefer to host and install their own gateway, giving them the ability to send encrypted data right to the mobile phones, with no need for translation. Time will show whether WTLS will be gradually adopted as the standard or if it will just be ignored.

### 3.10 Wireless Datagram Protocol (WDP)

WDP is the bottom most layer of the WAP stack. It is one of the elements that make WAP an extremely portable protocol. The WAP Datagram Protocol (WDP) is a datagram oriented, network layer protocol modeled after User Datagram Protocol (UDP) used on the Internet. UDP is a member of the TCP/IP protocol suite and is the simple, "best effort" data delivery protocol. On those networks where Internet protocols are present, WDP and UDP are identical. On networks where UDP is not available, WAP defines an UDP equivalent.

Bearer services such as SMS, CMD etc., are the nitty gritty of communication between the mobile phone and the Base Stations. WDP shields the upper layers from the bearer services provided by the network, so allowing the applications a transparent transmission of data over the different bearers. Since WDP has to interface with the idiosyncrasies of various bearer networks, it must have a bearer specific implementation. Therefore, if a WAP protocol stack is developed, WDP is the only layer that must be rewritten to support the different bearer networks.

### 3.11 Short Summary

The WAP protocol stack, is implemented is a layered approach (similar to the OSI mode). These layers consists (from top to bottom) of:

- Wireless Application Environment (WAE)
- Wireless Session Protocol (WSP)
- Wireless Transaction Protocol (WTP)
- Wireless Transport Layer Security (WTLS)
- Wireless Datagram Protocol (WDP)
- Bearers

In this chapter we have seen the functions of the different layers of the WAP stack.

### 3.12 Brain Storm

1. Define Protocol, layer and Protocol stack.
2. Mention the layer of the WAP Protocol Stack.
3. How does the WAP stack differ from the web stack?
4. What are the main building blocks of WAE?
5. Discuss the WAE Agent and WTA User Agent
6. What is the difference between a connection oriented service and a connectionless service?
7. Discuss the three classes of transaction services.
8. How does WTLS empower the SSL protocol?
9. Which layer of the WAP stack should have a bearer specific implementation and why?

END

## Lecture 4

---

# Introduction to WML

---

## Objectives

After completing this Lecture,  
you should be able to do the  
following

- ✎ WML Structure
- ✎ Basic WML Card
- ✎ Characteristics Of WML

---

## Coverage Plan

---

### Lecture 4

- 4.1 Snap Shot
- 4.2 WML Structure
- 4.3 Characteristics of WML
- 4.4 A Basic WML Card
- 4.5 The Header
- 4.6 The Body
- 4.7 Short Summary
- 4.8 Brain Storm

## 4.1 Snap Shot

In the early days of the Internet, HTML was created with the intention of specifying the contents to be displayed, leaving decisions as to how to display the contents to the browsers. However nowadays what is encapsulated in an HTML page is much more than the content; Layers, pictures, and special effects leave very little creativity to the browser.

With WML, WAP takes a step backwards in time to the old system. Wireless Markup Language (WML), formerly called Handheld Devices Markup Language (HDML). It a language that allows the text portions of Web pages to be presented on cellular telephones and personal digital assistants (PDAS) via wireless access. Application designers, we do not know to whom they are talking to, how big the screen is, or how many keys the keyboard has. They simply know that the screen is available and that it is tiny. The beautiful images provided by web sites, the astonishing dynamic effects, the sounds and all the other fancy things that can be found on the web cannot be made available on mobile devices.

WML and its supporting environment were designed with certain small narrow-band device constraints in mind including small displays, limited user-input facilities, narrow band network connections, limited memory resources and limited computational resources. Given the wide and varying range of terminals targeted by WAP, considerable effort was put into the proper distribution of presentation responsibility between the author and the browser implementation.

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

In this chapter we will be introducing the basics of WML, such as formatting text, adding links for implementing user navigation and using tables and images to add a professional touch to applications. We will also be seeing how we can allow the user to interact more with the application, by making selection, entering text, and so on.

*WML is an open language offered royalty-free. Specifications are available at Phone.com's Web site. A filter program can be written or may be available from a vendor that will translate HTML pages into WML pages.*

Before we start with WML, let us first take some time to look at the roots of WML and examine the language from which it is derived - XML

### **Xtensible Markup Language (XML)**

WML is a tag-based markup language and has been designed to display mainly text-based pages. It is specified as an XML document type. Knowing XML is not vital to writing Wireless Markup Language applications, but having a little background XML knowledge can help to make learning WML a faster process.

XML is a technology for creating structured documents that can be exchanged between systems. In XML the way the data is to be displayed is not described; rather, it is the structure and organization of the data that is defined.

XML is the product of the World Wide Web Consortium's effort to create a language for describing documents in a system independent way. It is subset of SGML (Standard Generalized Markup Language). In addition to document content, XML contains metadata – data that describes the content. In XML this was designed to be both operator-readable and machine-readable. This makes it a powerful technology that is available for both human and automatic agents' consumption.

## 4.2 WML Structure

WML is a tag-based markup language and has been designed to display mainly text-based pages. It is specified as an XML document type.

*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 through links.*

WML is based on a subset of HDML version 2.0. WML changes some elements adopted from HDML and introduces new elements, some of which have been modeled on similar elements in HTML. The resulting WML implements a card and deck metaphor.

### WML Components

Developing in WML is slightly different from developing for the web. As far as the development for the web is concerned, each HTML file constitutes one HTML page. In WML, since each page or screen is very small, it does not make much sense for each page to constitute a separate file. WML pages – content viewed on separate screens – are *cards* and the cards are all placed within a *deck* of related pages that constitute one single file.

Logically, a user navigates through a set of cards. The user navigates to a card, reviews its contents, may enter requested information, may make choices, and then moves on to another card. Instructions imbedded within cards may invoke services on origin servers as needed by particular interaction. Decks are fetched from origin server as and when needed. WML decks can be stored in 'static' files on an origin server, or the content generator that is running on an origin server can dynamically generate them. Each card, in a deck, contains a specification for a particular user interaction.

## 4.3 Characteristic features of WML

Although WML has limited capabilities compared to HTML, it nevertheless has a wide variety of features, including:

### Support for Text and Images

WML provides the authors with means to specify text and images to be presented to the user. This may include layout and presentation hints.

### Support for User Input

WML supports several elements to solicit user input. The elements can be combined into one or more cards. All requests for user input are made in abstract terms, allowing the user agent the freedom to optimize features for the particular device. WML includes a small set of input controls. For example, WML includes a text entry control that supports text and password entry. Text entry fields can be masked preventing the end user from entering incorrect character types. WML also supports client-side validation by allowing the author to invoke scripts at appropriate times to check the user's input. WML includes an option selection control that allows the author to present the user with a list of options that can set data, navigate among cards, or invoke scripts. WML supports both single and multiple option selections. WML also includes task invocation controls. When activated, these controls initiate navigation or a history management task such as traversing a link to another card (or script) or popping the current card off of the history stack. The user agent is free to choose how to present these controls. It may for example, bind them to physical keys on the device, render button controls in a particular region of the screen (or inline within the text), bind them to voice commands, etc.

### **Navigation and History Stack**

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

### **International Support**

The document character set for WML is the Universal Character set of ISO/IEC [10646]. Currently, this character set is identical to Unicode 2.0[UNICODE]. There is no requirement that WML decks be encoded using the full Unicode encoding. Any character encoding ("charset") that contains a proper subset of the logical characters in Unicode may be used.

### **Narrow – band optimization**

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

### **State and Context Management**

WML exposes a flat context (i.e., a linear non-nested context) to the author. Each WML input control can introduce variables. The state of the variables can be used to modify the contents of parameterized card without having to communicate with the server. Furthermore, the lifetime of a variable state can last longer than a single deck and can be shared across multiple decks without having to use a server to save to save intermediate state between deck invocations.

*When a WML page is accessed from a mobile phone, all the cards in the page are downloaded from the WAP server. Navigation between the cards is done by the phone computer - inside the phone - without any extra access trips to the server.*

#### 4.4 A Basic WML Card

As with all new technologies, the best place to start is at the very beginning. Shown below is an example of a basic WML card.

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

<!-- This is the First WML Program -->
<wml>
    <card id="main" title="Radiant">
        <p align = "center">
            Join with us to make a Wireless World...
        </p>
    </card>
</wml>
```

The result might look like this on your mobile phone display:



**Fig 4.1** Program to display "Join with us to make a Wireless World"

Let's look at this example, piece by piece, to see how it has been put together and what each section does.

#### 4.5 The Header

Every WML deck starts with the same XML header:

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

The first line states that what follows is an XML document and the version number used. The next line selects the document type and gives the URL of the Document Type Definition (DTD). This DTD gives the full XML definition of WML.

#### 4.6 The Body

The WML code of a deck is enclosed in the <wml></wml> tag pair. This is the body of the document, and the cards are defined within it using the <card></card> tag pair.

```
<wml>
    <card id="no1" title="Card 1">
```



```
<p> Join with us to make a wireless world!</p>
</card>
</wml>
```

The id attribute gives the card an identifier that can be used to refer to it in other parts of the WML. It can also be used to refer to a particular section of a deck from an external deck. The title attribute gives the name of the card to be presented to the user.

### Comments

WML comments follow the XML commenting style and have the following syntax:

```
<!-- comments -->
```

Comments are used for further reference to the WML author and are not displayed to the user.

## 4.7 Short Summary

- 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 through links
- When a WML page is accessed from a mobile phone, all the cards in the page are downloaded from the WAP server. Navigation between the cards is done by the phone computer - inside the phone - without any extra access trips to the server

## 4.8 Brain Storm

1. Short Note on WML Component?
2. What are the Characteristic Features of WML?
3. Discuss about the HEADER in WML?



## Lecture 5

---

# Style Elements, Tables and Entities

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ WAP Model Text Formatted Elements
- ✔ Tables
- ✔ WML Entities

---

## Coverage Plan

---

### **Lecture 5**

- 5.1 Snap Shot
- 5.2 Text Formatted Elements
- 5.3 THE HEAD Element
- 5.4 Tables
- 5.5 WML Entities
- 5.6 Short Summary
- 5.7 Brain Storm

## 5.1 Snap Shot

In this lecture you are going to learn about Style Formatting Elements, Tables and Entities.

## 5.2 Text Formatting Elements

In this section, we're going to see the tags defined for formatting text. In particular, we will be taking a look at how to handle paragraphs, line wrapping, and how to add different styles to plain text. How all these features are displayed is very device dependent.

### Paragraphs and the <p> Element

In the first example given above, the phrase "Join with us to make a Wireless world" was enclosed in <p></p>tags. These tags mark a section of text as a paragraph.

#### Syntax

```
<p align = "left | center | right" mode = "nowrap | wrap">
```

The align attribute is used to control the paragraph alignment of the text on the browser screen. The mode attribute is used to specify whether the browser should automatically wrap the lines when it gets to the end of the screen. In the nowrap modes wrapping is turned off. Default for the first card in a deck is wrap.

In WML, all text to be displayed on the main part of the screen must be inside a paragraph element.

Of course, there can be more than one paragraph inside a card, but remember not to fit too much of text onto one card. The typically small displays on a majority of WAP devices makes large quantities of data cumbersome for a user to absorb. In addition, large quantities of data will have an effect on the fetch speed and may decrease user satisfaction.

The card shown below is illegal and would generate an error in the browser because the WML does not conform to the definition given in the DTD:

```
<card id = "main" title =First "Example">  
  Join with us to make a Wireless World!  
</card>
```

The missing <p></p>tags will generate an error in this case.

#### Note

- WML is a case sensitive language. The Card and card are different things.

### Text Style Elements

Picking out information can be quite difficult with too much text on the small screen of a mobile device. What is needed is a way to mark certain parts of the text as noteworthy. WML provides several tags for this purpose.

#### The Bold <b> Element

This tag is used to make something **stand** out.

**Syntax**

`<b> text contents </b>`

Eg., `<b> Wireless Application Protocol </b>`

Result:

*Wireless Application Protocol*

**The Italic `<i>` Element**

This tag is used to display the text in a *slanted* form.

**Syntax**

`<i> text contents </i>`

Eg.,

`<i> Wireless Application Protocol at Radiant</i>`

Result

*Wireless Application Protocol at Radiant*

**The Underline `<u>` Element**

This tag is used to display the underlined text.

**Syntax**

`<u> text content </u>`

Eg.,

`<u> Wireless Application Protocol </u>`

Result

Wireless Application Protocol

**The Emphasis `<em>` Element**

This tag emphasis the text in some way; The choice is left to the device in this case.

Syntax:

`<em> text content </em>`

Eg.

`<em> Wireless Application Protocol </em>`

Result

*Wireless Application Protocol*

### The Strong <strong> Element

This tag is similar to emphasis, the device may apply a style to indicate that the text is signified.

Syntax	<code>&lt;strong&gt; Text content &lt;/strong&gt;</code>
Eg.,	<code>&lt;strong&gt; Wireless Application Protocol &lt;/strong&gt;</code>
Result:	

### The Big <big> element

This is used to increase the size of the text . It also makes the text to **stand** out.

Syntax	<code>&lt;big&gt; text content &lt;/big&gt;</code>
Eg.,	<code>&lt;big&gt; Wireless Application Protocol &lt;/big&gt;</code>
<u>Result</u>	Wireless Application Protocol

### The small <small> Element

This is reduces the size of the text.

Syntax	<code>&lt;small&gt; text content &lt;/small&gt;</code>
Eg	<code>&lt;small&gt; Wireless Application Protocol &lt;/small&gt;</code>
<u>Result</u>	Wireless Application Protocol

### The Line breaking <br/> Element

This is used for inserting a line break into the text. This means that a new line is to be started for all content immediately following the <br/> tag.

Syntax	<code>&lt;br/&gt;</code>
Eg.,	Wireless   Application   Protocol
<u>Result</u>	Wireless Application Protocol

Style	Tag	Description
-------	-----	-------------

Bold	<code>&lt;b&gt;&lt;/b&gt;</code>	Makes something stand out more.
Italic	<code>&lt;i&gt;&lt;/i&gt;</code>	Use slanted text.
Underline	<code>&lt;u&gt;&lt;/u&gt;</code>	Draw a line under the text.
Emphasis	<code>&lt;em&gt;&lt;/em&gt;</code>	Emphases the text in some way, the choice is left to the device
Strong	<code>&lt;strong&gt;&lt;/strong&gt;</code>	Similar to emphasis, device may apply a style to indicate the text is significant.
Big	<code>&lt;big&gt;&lt;/big&gt;</code>	Increases the size of the text.
Small	<code>&lt;small&gt;&lt;/small&gt;</code>	Reduces the size of the text.

Table : Text Style Elements

The example given below shows how 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>
```

Example 5.1 To show the usage of WML formatting styles.

The result might look like this on your mobile phone display (don't take it for granted that all formatting tags will render as expected):



### 5.3 The Head Element

The head element contains information relating to the deck as a whole, including meta-data and access control elements. At least one of the following elements should be included:

- ◆ access
- ◆ meta

#### The access element

The access element specifies access control information for the entire deck. Every deck can contain only one access element. If a deck does not include an access element, access control is disabled. When access control is disabled, cards from any deck can access this deck.

#### Syntax:

```
<access domain = "domain-name" path = "entire path">
```

Domain and path attributes specify the other decks that can access the current deck. As the user agent navigates from one deck to another, it performs access control to determine whether the destination decks allows access from the current deck.

#### Example

```
<head>
  <access domain="mycompany.com" path="/mypage">
</head>
```

#### The meta Element

The meta element contains generic meta information relating to the WML deck. Meta information is specified with property names and values.

#### Example

```
<head>
  <access domain="mycompany.com" path="/mypage">
  <meta content="charset" user agent = "character-set=UTF-8"/>
  <meta name = "author" content = "Karthik" />
</head>
```

### 5.4 Tables

Often, information is to be extracted from the database and displayed to the user in a useful and intuitive way. The information retrieved from the databases is well suited to being displayed in a tabular manner.



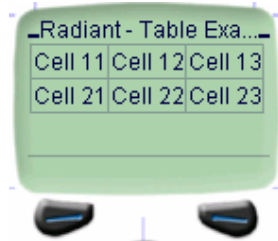
The following example shows how a WML card can be set up to display the table functions of WML as shown in the example below:

```
<?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="main" title="Radiant - Table Example">
    <p>
      <table columns="3">
        <tr>
          <td>Cell 11</td>
          <td>Cell 12</td>
          <td>Cell 13</td>
        </tr>
        <tr>
          <td>Cell 21</td>
          <td>Cell 22</td>
          <td>Cell 23</td>
        </tr>
      </table>
    </p>
  </card>
</wml>
```

*Example 5.2 An application to show how WML table tag works.*

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



## 5.5 WML Entities

There are some characters that have a special meaning in WML. We have already seen that all tags are encased in < and > characters. If one of these characters needs to be displayed on the screen of a WAP device, it is possible to enter the character using its ASCII number, in the form &#number. In addition, there are shortcut forms, so for example > can be represented as &#62; and < as &#60;. Another character that has a special meaning in WML is the ampersand (&) character; if we wish to output it to the screen we can use &#38;. This is the same encoding as is used in HTML.

The following example exemplifies this. It displays shows this in use, to display the less than, and greater than characters that have ASCII codes 60 and 62:

```
<?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="main" title="Maths facts">
    <p align="center">
```

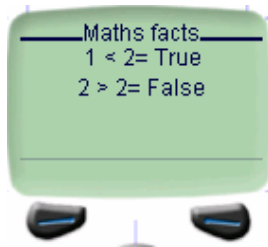
```

1 &#60; 2= True <br/>
2 &#62; 2= False <br/>
</p>
</card>
</wml>

```

Example 5.3 This shows how WML entities work.

This would display as



The following table shows some of the most commonly used characters, and the names by which they are referred to within the WML code.

Result	Description	Entity Name	Entity Number
&	Ampersand	&amp;	&#38;
'	Apostrophe	&apos;	&#39;
>	greater-than	&gt;	&#62;
<	less-than	&lt;	&#60;
	non-breaking space	&nbsp;	&#160;
"	quotation mark	&quot;	&#34;
--	soft hyphen	&shy;	&#173;

Table : Commonly used characters

```

<?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" title="First card">
    <p>
      &quot;
      &nbsp; &nbsp; &nbsp;
      &apos;
      &nbsp; &nbsp; &nbsp;
      &amp;
      <br/>
      &gt;
    </p>
  </card>
</wml>

```

*Example 5.4 Program to output commonly used characters.*



## 5.6 Short Summary

- ❖ In WML, all text to be displayed on the main part of the screen must be inside a paragraph element

## 5.7 Brain Storm

1. Discuss about Text Formatted Document?
2. Explain about WML Entities?
3. Discuss about Tables in WML?

END

## Lecture 6

---

# WML Elements

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✧ Navigational Elements
- ✧ WML Task Elements
- ✧ Template Elements

---

## Coverage Plan

---

### Lecture 6

- 6.1 Snap Shot
- 6.2 Navigational Elements
- 6.3 WML Task Elements
- 6.4 Template Elements
- 6.5 Short Summary
- 6.6 Brain Storm

## 6.1 Snap Shot

In this lecture you are going to learn about some of the WML elements such as Navigational element, Task elements and template elements.

## 6.2 Navigational Elements

Implementing good navigation around a WAP application is important since very little can be displayed on the small screen that the device has. In WML the elements that are used for navigation between WML cards are <do>, <a>, <anchor>and <go>. These are discussed below.

### The do Element

The do element gives the user a general mechanism for performing navigation between cards. The do element establishes a soft key. Soft keys are instrumental to navigation on WAP devices. These buttons give the user the ability to alter the path or jump to a different card without having to navigate a full list of options.

The do element may appear at both the card-level and deck level.

### Card-Level

The do element may appear inside a card and may be located anywhere in the text flow. If the user agent intends to render the do element inline, it should use the element's anchor point as the rendering point.

### Deck-Level

The do element may appear inside a template, indicating a deck level do element. A deck level do element applies to all the cards in the deck. It is equivalent to specifying the do element with in each card.

A card level do element overrides a deck-level do element if they have the same name. For a single card, the active do element are defined as the do elements specified in the card, plus any do elements specified in the deck's template and not overridden in the card.

Syntax

```
<do type="type" label="label" name="name">
```

### Attributes of the do Element

#### ❖ Type

This attribute provides an indication to the user agent about how the intended the element is to be used and how it should be mapped to a physical user interface construction. This attribute is required.

The values that this type attribute can taken are:

- accept - Positive acknowledgement
- prev - Navigates backwards through history
- help - Request for help

#### ❖ Name

This attribute specifies the name of the do element. This attribute is optional.

#### ❖ Label

This attribute specifies a textual string for labeling the user interface widget. Labels should have no more than 6 characters. This attribute is optional.

### 6.3 WML Task Elements

WML allows the user to specify tasks that can be performed when a certain event occurs, such as navigating to a specified card or deck.

WML includes four task elements to achieve this

- ◆ go task
- ◆ prev task
- ◆ noop task
- ◆ refresh task
- ◆ go task

The go element declares a go task, indicating navigation to an URL. A go executes a push operation on the history stack.

#### Contained Elements

- ◆ setvar
- ◆ postfield

#### Syntax

```
<go href="href" sendreferer = "true | false" method = "post | get">
```

The attribute of the go task element are href, sendreferer method.

- href specifies the destination URI.
- sendreferer - if it is true the user agent includes the URL of the deck containing this task in the URI request. The default value is false.
- method - this attribute specifies the HTTP submission method. The default value is get.

#### Prev Task

The prev element declares a prev task, indicating navigation to the previous URI in the history stack.

#### Syntax

```
<prev/>
```

#### Noop Task

The noop element specifies that nothing should be done, that is “no operation”. This attribute is useful for overriding deck-level do elements.

#### Syntax

```
<noop/>
```

#### Refresh Task

This indicating an update of the specified card variables.

#### Syntax

```
</refresh>
```

#### Setvar Element

The setvar element specifies the variable to be set in the current browser context as a side effect of executing a task. The element is ignored if the name attribute does not evaluate to a legal variable name at runtime.

#### Syntax

```
<setvar name="vname" value="value">
```

Name specifies the variable name and Value specifies the value to be assigned to the variable. Both the attributes are required.

#### Example

A Program to show the use of navigational elements and the task elements.

```
<?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="first" title="Radiant-First Card">
    <p align="center">
      Welcome to Radiant <br/>
      This is the First card
    <do type="accept" label="Next">
      <go href="#second" />
    </do>
  </p>
</card>
  <card id="second" title="Radiant-second Card">
    <p align = "center">
      Join with us to make a Wireless World <br/>
      This is the second card
    <do type="accept" label="Next">
      <go href="#third" />
    </do>
    <do type="prev" label="Prev">
      <go href="#first" />
    </do>
  </p>
</card>
  <card id="third" title="Radiant-Third Card">
    <p align = "center">
      Radiant - The Wireless World <br/>
      WAP - The Ultimate Technology <br/>
      This is the Third card
    <do type="prev" label="Prev">
```

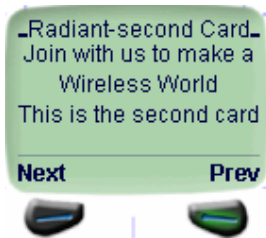


```

        <go href="#second" />
    </do>
</p>
</card>
</wml>

```

The result might look like this when we navigate from the first card to the second card.

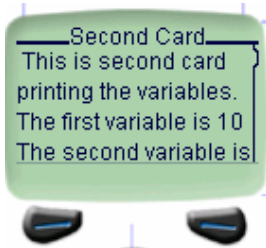


```

<?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" title="Radianr-First Card">
    <p>
      Here we are setting the Variables
      <do type="accept" label="Ok">
        <go href="#card2">
          <setvar name="v1" value="10"/>
            <setvar name="v2" value="20"/>
        </go>
      </do>
    </p>
  </card>
  <card id="card2" title="Second Card">
    <p>
      This is second card printing the variables. <br/><br/>
      The first variable is $v1 <br/>
      The second variable is $v2 <br/>
    </p>
  </card>
</wml>

```

The second card might look like this



Note :

The symbol \$ used to access the variable.

## 6.4 Template Elements

The template element enables you to specify that certain task elements which can be executed whenever any card in the deck is entered. A template is given at the start of deck, directly following the <wml> tag, and is specified using the <template> </template> tag pair. Everything that is between the <template></template> tags is automatically inserted into every card within the deck.

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 id="card1" title="Card1">
    <p align="center">
      This is First Card
    <do type="accept" label="Next">
      <go href="#card2"/>
    </do>
    </p>
  </card>
  <card id="card2" title="Card2">
    <p align="center">
      This is Second Card
    <do type="accept" label="Next">
      <go href="#card3"/>
    </do>
    <do type="prev" label="Prev">
      <go href="#card1"/>
    </do>
    </p>
  </card>
  <card id="card3" title="Card3">
    <p align="center">
      This is Third Card
    <do type="prev" label="Prev">
      <go href="#card2"/>
    </do>
    </p>
  </card>
</wml>
```

The second card would look like this:



Same Program with template

```

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

<wml>
  <template>
    <do type="prev" label="Prev">
      <prev/></do>
    </template>
  <card id="card1" title="Card1">
    <p align="center">
      This is First Card
    <do type="accept" label="Next">
      <go href="#card2"/>
    </do>
    </p>
  </card>
  <card id="card2" title="Card2">
    <p align="center">
      This is Second Card
    <do type="accept" label="Next">
      <go href="#card3"/>
    </do>
    </p>
  </card>
  <card id="card3" title="Card3">
    <p align="center">
      This is Third Card
    </p>
  </card>
</wml>

```

The output would be as shown below.



## 6.5 Short Summary

- The template element enables you to specify that certain task elements which can be executed whenever any card in the deck is entered.
- The setvar element specifies the variable to be set in the current browser context as a side effect of executing a task.

## 6.6 Brain Storm

1. Discuss about Navigational element in WML?
2. What is Template Element?
3. Short Note on Task Element in WML?

୧୦୩

## Lecture 7

---

# Links and Images

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ Links
- ✔ Images
- ✔ Interacting with Users

---

## Coverage Plan

---

### Lecture 7

7.1 Snap Shot

7.2 Links

7.3 Images

7.4 Interacting With Users

7.5 Short Summary

7.6 Brain Storm

## 7.1 Snap Shot

In this lecture you are going to learn about how to get link with other files and how to insert your images.

## 7.2 Links and Images

Implementing good navigation around a WAP application is important since very little can be displayed on the small screen that the device has. This section examines how to write a WML code to allow a user to move around.

<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 first.wml":

### Example

Program to show the use of might anchor tag. The output look like this on the mobile device.

```
<?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="main" title="Anchor Tag">
    <p>Click here for next page...<br/>
      <anchor>Next page
        <go href="first.wml"/>
      </anchor>
    </p>
  </card>
</wml>
```

An example to show the usage to anchor tag.



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

```
?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>
    Enter to Next page...
    <a href="first.wml">Next page</a>
  </p>
</card>
</wml>

```

An example to show the usage of <a> tag.

The output might look like this on your mobile device.

When the link is clicked the output

would be as shown below



### 7.3 Images

'A picture is worth a thousand words'. With such small screens, small images instead of text can be very useful, allowing us to get more information across to the user in the small space available. Images can be particularly useful when used as links. For example an application could be navigated using small icons instead of lots of text.

```

<?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="main" title="Image">
    <p>
      This is an image <br/>
       <br/>
      in a paragraph
    </p>
  </card>
</wml>

```

To show how img tag works. The result might look like this in your mobile phone display:





**Note**

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

**7.4 Interacting with Users**

WML can be used for more than displaying pre-formatted documents – we can use it to a much greater extent by enabling the user to interact with the WAP device. Given the limitations of memory and processing power of current devices, the real power of the wireless Internet lies not in the WAP device, but with the server.

With the breadth of options available on the web, it will not be too long before an application will require the user to make a decision as to what to do next. WML provides us with several ways to allow a user to make a selection on a WAP device. We will look at the options menu, and select lists, two mechanisms provided to create a list of options.

**7.4.1 Input Element**

The input element specifies a text entry object. We can specify the format of the user with the optional format attribute. If a valid input mask is bound to an input object, the user agent must ensure that any value collected by the entry object conforms to the bound input mask.

**Syntax**

```
<input type = "text | password" name = "name" value = "default value"
format=cdata" size = "number" maxlength = "number" emptyok = "boolean"
title="vdata">
```

Here indicates the type of the input element. It has two values, text and password. If text is the input element then it gets the data from the input formatted by format property. If the input element is a password then it echoes\* characters.

*name* attribute gives the unique name of the input element.

*value* attribute sets the default value of the text box.

*format* attribute specifies an input mask for user input entries. The string consists of mask control characters and static text that is displayed in the input area. An input mask is only valid when it contains legal format codes only. User agent must ignore invalid masks.

Values of format attribute

- A - Allows any non-numeric uppercase alphabetic or punctuation character non-numeric.
- A - Allows any lowercase character or punctuation character.
- N - Allows any numeric character.
- X - Allows any uppercase character.
- x - Allows any lowercase character.
- M - Allows any character. The user agent may choose to assume that the character is uppercase for the purposes of simple data entry, but must allow entry of any character.
- m - Allows any character. The user agent may choose to assume that the character is lowercase for the purposes of simple data entry, but must allow entry of any character.

- \*f - Allows any number of characters; f is one of the above format codes and specifies what kind of characters can be entered.
- nf - Allows n characters where n is a number from 1 to 9; f is one of the above format codes and specifies what kind of characters can be entered.
- \c - Displays the next character, c in the entry field.
- emptyok- If you set this attribute to true, the input element accepts empty input. If it is false, user must enter the values.
- size - Specifies the width of the text input area in characters.
- maxlength- Specifies the maximum number of characters that the user can enter in the text entry area. The default value for this attribute is an unlimited number of characters.

**Example**

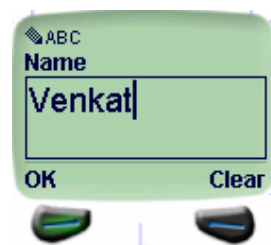
To show the use of the input element. There might look like this on your mobile device.

```
<?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" title="First Card">
    <p>
      Enter Your name <input type="text" name="nam"
maxlength="10" title="Name"/>
      Enter Your Age <input type="text" name="age"
format="N*N" maxlength="2" title="age" />
    </p>
  </card>
</wml>
```

On confirming the name that is entered, in the following would be displayed

If the user chooses to enter the name then the display would look like the following.



If the user now chooses to enter the age then the displayed might be as shown below.



On clicking "OK" the screen of the mobile device would appear as follows.



### 7.4.2 Select Element

Select list is an input element that allows the user to choose from a list of options.



WML supports both single-choice and multiple choice lists.

The select element lets the users pick from a list of options. Each option is specified by an option element having one line of formatted text. You can organize option elements into hierarchical groups using the optgroup element.

You must include one of the following elements at least once inside a select ELEMENT:

- ◆ optgroup
- ◆ option

#### Syntax

- name - Name of the Variable
- multiple - If it is true, the select list accepts multiple selections. If it is false, select list accepts only a single selected option. The default value is false.
- value - The default value of the variable.
- title - Title for the Selection option.

#### Example Program

To show the usage of select element.

```
<?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" title="First Card">
  <p>
    Select your Country
    <select name="sel" value="n">
      <option value="India">India </option>
      <option value="USA">USA </option>
      <option value="Singapore">Singapore </option>
    </select>
  </p>
</card>
</wml>

```

To output would look like this on the screen of the mobile device



On selecting, say Singapore, the output would look as follows



### 7.4.3 Fieldset

A WML card, can be set up to display the fieldset function of WML:

```

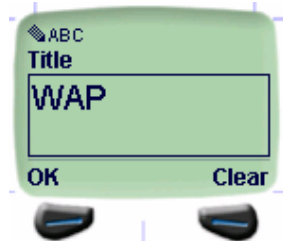
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card title="Fieldset example">
    <p> Enter... <br/>
    <fieldset title="CD Info">
      Title: <input name="title" type="text" title="Title"/><br/>
      Prize: <input name="prize" type="text" title="Prize"/>
    </fieldset>
  </p>
</card>
</wml>

```

An example for fieldset.

On executing the output would look as shown below.

If the user wishes to enter the title, the title, the screen would look as follows.



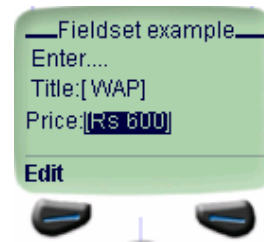
On confirmation the output on the screen would appear as follows



If the user chooses enter the price details, the screen would appear as follows



On confirming the price, the result might look like this in your mobile phone display.



## 7.5 Short Summary

- **Format** attribute specifies an input mask for user input entries. The string consists of mask control characters and static text that is displayed in the input area. An input mask is only valid when it contains legal format codes only.
- **.WBMP** is the only image type that can be displayed in a WAP browser.

## 7.6 Brain Storm

1. Explain how Images are Linked in WML?
2. What are Input Elements? Explain briefly.
3. Discuss about Select Element?
4. Define Fieldset?

☺☺☺

## Lecture 8

---

# WML Events

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ Onenterforward element
- ✔ Onenterbackward element
- ✔ Ontimer element
- ✔ Onpick element-variables

---

## Coverage Plan

---

Lecture 8
8.1 Snap Shot
8.2 Ontimer Event
8.3 Onenterforward Event
8.4 Onenterbackward Event
8.5 Ontimer Event
8.6 Onpick Event
8.7 Variables
8.8 Short Summary
8.8 Brain Storm

## 8.1 Snap Shot

Events can be used to make WAP applications dynamic. As the name suggests, an event is simply the happening of something notable. Events are a very common concept in windows-based programs and web programming. For example, when a PC user clicks on an icon in a window, it generates an event.

The various events that you can respond to in WML are summarized below:

Event Name	Description
Oneventforward	Occurs when the user navigates directly to the card, whether using the do element or by entering a URL directly.
Oneventbackward	Occurs when the user navigates to the deck, using a URL retrieved from the history stack.
Onpick	Occurs when the user selects, or deselects, an item.
Ontimer	Occurs when a timer expires.

Table : WML EVENTS

When one of these events occurs, a piece of code associated with it – called an event handler – is executed. There are many types of events. They can be generated by a window opening, an item being clicked on with the mouse, a set amount of time expiring or a menu being opened to name but a few. In WML, events are generated when a card is made visible, whether by navigating through to it (the onenter forward event) or by returning to it (onenter backwards), when a set amount of time has expired (ontimer event) or when a selection is made (onpick event). Many of these events are handled using the <onevent> element, with the exception of the onpick event, which is covered later in this chapter.

The use of this element is:

```
<onevent type = "eventname" >
    action
</onevent>
```

## 8.2 Ontimer Event

The ontimer event can be specified inside the following elements:

- ◆ card
- ◆ template

This event occurs when a timer expires.

### Syntax

```
(ontimer = href)
```

Specifies an intrinsic event that instructs the user agent to go to the specified URI after the timer has expired.

### Example Program

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
```



```

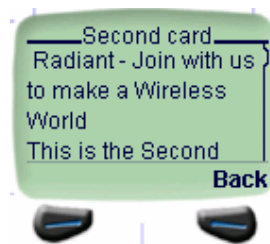
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="card1" title="First card"
ontimer="#card2">
    <timer value="10"/>
    <p>
      Welcome to Radiant...<br/>
      This is First Card
    <do type = "accept" label="Next">
      <go href="#card2"/>
    </do>
    </p>
  </card>
  <card id="card2" title="Second card"
ontimer="#card1">
    <timer value = "20"/>
    <p>
      Radiant - Join with us to make a Wireless World <br/>
      This is the Second Card
    <do type = "prev" label="Back">
      <prev/>
    </do>
    </p>
  </card>
</wml>

```

The following would be visible on the screen for 10 seconds. Automatically, the second card comes on view as soon as the timer expires.



The second card, shown below, would be displayed for 20 seconds. At the expiry of the timer, the user is taken back to the first card.



**onenterforward event**

The onenterforward event occurs when the user enters a card using a go task or any method with identical semantics.

It may be specified inside the following elements:

- ◆ card
- ◆ template

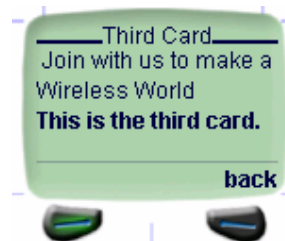
**Syntax**

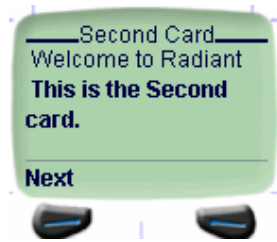
```
<onenterforward=href>
```

href indicates the destination url.

**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 id="card1" title="First Card">
    <p>
      WAP - The Ultimate Technology <br/>
      <b> This is first card. </b>
      <do type="accept" label="Next">
        <go href="#card2" />
      </do>
    </p>
  </card>
  <card id="card2" title="Second Card" onenterforward="#card3">
    <p>
      Welcome to Radiant <br/>
      <b> This is the Second card. </b>
    </p>
    <do type="accept" label="Next">
      <go href="#card3" />
    </do>
  </card>
  <card id="card3" title="Third Card">
    <p>
      Join with us to make a Wireless World <br/>
      <b>This is the third card. </b>
    </p>
    <do type="prev" label="back">
      <prev/>
    </do>
  </card>
</wml>
```





### 8.3 Onenterbackward

The onenterbackward event occurs when the user navigates into a card using a prev task or any method with identical semantics. This event may be specified inside the following elements:

- ◆ card
- ◆ template

Event bindings specified in the template apply to all cards in the deck and may be overridden.

#### Syntax

```
<onenterbackward = href>
```

href specifies an intrinsic event that instructs the user agent to go to the specified URI when the user navigates backwards to this card using prev task.

Example program

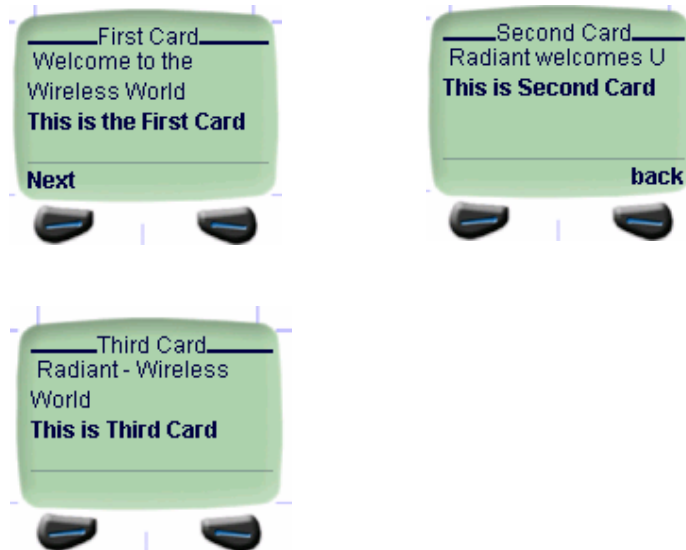
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card id="card1" title="First Card"
        onenterbackward="#card3">
    <p>
      Welcome to the Wireless World <br/>
      <b>This is the First Card</b>
    </p>
    <do type="accept" label="Next">
      <go href="#card2"/>
    </do>
  </card>
  <card id="card2" title="Second Card">
    <p>
      Radiant welcomes U <br/>
      <b>This is Second Card</b>
    </p>
    <do type="prev" label="back">
      <prev/>
    </do>
  </card>
  <card id="card3" title="Third Card">
```

```

<p>
  Radiant - Wireless World <br/>
  <b>This is Third Card</b>
</p>
</card>
</wml>

```



## 8.5 Onpick event

In a multile selection option list, when the user selects or deselects the option having the onpick attribute, the user agent navigates to the URL specified by the onpick attribute.

### Syntax

```

<option onpick="url">

<?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="first" title="Onpick Example">
    <p align="center">
      Select Any one <br/>
      <select name="sel" value="0">
        <option value="0"> Select </option>
        <option value="1" onpick="#second"> Java </option>
        <option value="2" onpick="#third"> Cadre </option>
        <option value="3" onpick="#fourth"> Oracle </option>
        <option value="4" onpick="#fifth"> WAP </option>
      </select>
    </p>
  </card>

  <card id="second" title="Radiant-Java">
    <p align="center">

```

```

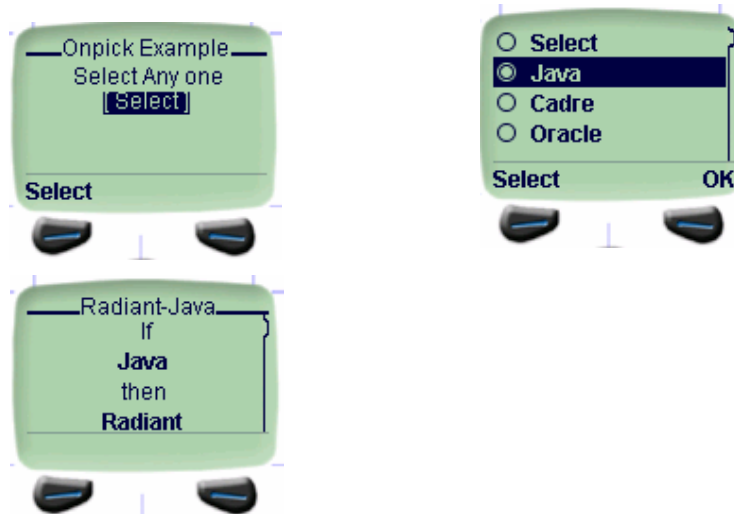
If <br/> <b>Java</b> <br/> then <br/> <b>Radiant</b>
  </p>
</card>

<card id="third" title="Radiant-Cadre">
  <p align="center">
    <b> Fly with CADRe....</b>
  </p>
</card>

<card id="fourth" title="Radiant-Oracle">
  <p align="center">
    <b>Oracle at its best</b>
  </p>
</card>

<card id="fifth" title="Radiant-WAP">
  <p align="center">
<b>Radiant </b><br/> <i><b>Join with us to make a Wireless World
</b></i>
  </p>
</card>
</wml>

```



## 8.6 Variables

When a user switch from card to card in a deck, we need to store data in variables. WML variables are case sensitive.

### Naming the Variable

#### Specify a Variable with the Setvar Command

When someone executes a task (like go, prev, and refresh), the setvar element can be used to set a variable with a specified value.

The following example will create a variable named i with a value of 500:

```
<setvar name="i" value = "500" />
```

The name and value attributes are required.

### Specify a Variable through an Input Element

Variables can also be set through an input element (like input, select, option, etc.).

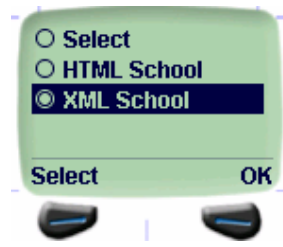
```
<card id="card1">
<select name="schoolname">
<option value="HTML">HTML School</option>
<option value="XML">XML School</option>
</select>
</card>
```

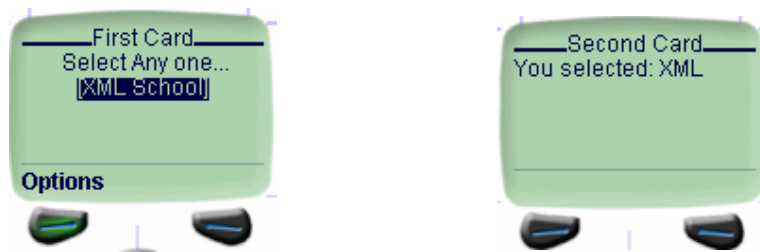
To use the variable we created in the example above:

```
<card id="card2">
<p>You selected: ${schoolname}</p>
</card>
```

### 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 id="first" title="First Card">
    <p align="center">
      Select Any one... <br/>
      <select name="schoolname">
        <option value="no">Select </option>
        <option value="HTML">HTML School</option>
        <option value="XML">XML School</option>
      </select>
      <do type="access" label="next">
        <go href="#card2"/>
      </do>
    </p>
  </card>
  <card id="card2" title="Second Card">
    <p>You selected: ${schoolname}</p>
  </card>
</wml>
```





### WML Examples

A WML deck with two cards - one for user input and one for displaying the result - can be set up, like demonstrated in this 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 id="card1" title="School">
<do type="accept" label="Answer">
  <go href="#card2"/>
</do>
<p>
<select name="name">
  <option value="HTML">HTML School</option>
  <option value="XML">XML School</option>
  <option value="WAP">WAP School</option>
</select>
</p>
</card>
<card id="card2" title="Answer">
<p>
You selected: $(name)
</p>
</card>
</wml>
```

The first card might look like this in your mobile phone display:

*//output to be added*

THE SECOND CARD MIGHT LOOK LIKE THIS:

*//output to be added*

Example Explained

### The Prolog

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

The first lines in the WML document is called the prolog: The prolog defines that this is an XML document. It defines the XML version, and the DTD to be referenced.

### The Deck

```
<wml> . . . . . </wml>
```

The deck is the WML document itself. It is embedded within <wml> tags

### The Cards

```
<card> . . . . . </card>
```

Card are always displayed one at the time. The two cards named "card1" and "card2" are defined between <card> tags

### The <do> element

```
<do> . . . </do>
```

The first card has a <do> element that defines an event to be triggered. The **type="accept"** attribute of the do element causes the **label="Answer"** to be displayed in the lower left corner of the display.

### The Event

The go element triggers when the user clicks the do label. The **href="#card2"** attribute of the go element causes card2 to be displayed on the screen.

### The Variable

Card2 displays the **\$(name)** variable from card1, because variables are valid across cards.

## 8.7 Short Summary

- The onenterforward event occurs when the user enters a card using a go task or any method with identical semantics.
- The onenterbackward event occurs when the user navigates into a card using a prev task or any method with identical semantics.
- The first lines in the WML document is called the prolog: The prolog defines that this is an XML document. It defines the XML version, and the DTD to be referenced.

## 8.8 Brain Storm

1. Discuss briefly about WML Event?
2. What is Prolog?
3. What is the use of DO Element?





## Lecture 9

---

# WML Script

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ Language Syntax
- ✔ Data types
- ✔ Operators

---

## Coverage Plan

---

Lecture 9
9.1 Snap Shot
9.2 WML Script Language Syntax
9.3 How To Call A WMLscript From A WML Page
9.4 Data Types
9.5 Operators
9.6 Short Summary
9.7 Brain Storm

## 9.1 Snap Shot

The WMLScript language is part of the industry-wide standard for wireless applications called Wireless Application Protocol (WAP). It is the scripting language used in WML pages.

WMLScript is based on JavaScript, but it has been modified to provide better support for low bandwidth communication and thin clients. Several changes to the language have been made in order accomplish this. First, WMLScript can be compiled to a bytecode, speeding up the interpretation by the user agent. Secondly, it is a procedural language and it supports locally-installed standard libraries of WMLScript procedures. Finally, it does not support some of the more advanced features of JavaScript that are heavy on computational and bandwidth needs.

WMLScript is a client-only scripting platform used in combination with WML to provide client side procedural logic. Like WML, WMLScript is compiled via a WAP gateway into binary form to provide intelligence to mobile clients.

WMLScript is not embedded in the WML pages. Instead a separate file is created that contains the code. This file is then called from the card on view. The WML Script is called via hyperlink. The browser then initializes an additional trip to the server to request the script file. It then loads it into the memory and runs the code. WMLScript is compiled into bytecode on the server before it sent to the WAP browser.

WMLScript is used to validate user inputs. It is also used to generate message boxes and dialog boxes locally to view error messages and confirmations faster. The user agent facilities can be accessed using WMLScript.

Many of the services that can be used with hin mobile clients can be implemented with WML. Scripting enhances standard browsing and presentation facilites of WML with behavioural capabilities.

## 9.2 WMLScript Language Syntax

WMLScript syntax is based on the ECMAScript programming language. Unlike ECMAScript, however, the WMLScript specification also defines a bytecode and interpreter reference architecture for optimal utilization of current narrowband communications channels and handheld device memory requirements. The following points help summarize some basic syntactical features of the language:

- The smallest unit of execution in WMLScript is a statement and each statement must end with a semicolon (;).
- WMLScript is case-sensitive like JavaScript. All language keywords and function names must use the proper capitalisation of letters.
- Comments can either be single-line (beginning with //) or multi-line (bracketed by /\* and \*/). This syntax is identical to both C++ and Java.
- A literal character string is defined as any sequence of zeros or more characters enclosed within double (") or single (') quotes.
- Boolean literal values correspond to true and false.
- New variables are declared using the var keyword (i.e. var x;).

- Spaces, tabs, newlines etc that appear between tokens in program, except that occur in string constants are inored.

### 9.3 How to call a WMLScript from a WML page

In the example given below there is a card where on entering URL selecting and the go label, the specified URL, can be accessed.

#### Example 5.1

```
<?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('url')"/>
</do>
<p>
Enter a URL:
<input type="text" name="url"/>
</p>
</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**.

The **go\_url** function sends you to the URL you entered. The function also checks if you omitted the "http://" part in the URL, if you did; this part will be added.

#### Program 5.1

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

```
extern function go_url(url)
{
if (String.find(url,"http://") < 0)
{
url="http://"+url;
}
WMLBrowser.go(url);
}
```

#### Note

- 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.

## 9.4 Data Types

WMLScript is a weakly typed language. This means that no type-checking is done at compile- or run-time and no variable types are explicitly declared. The variables and literals can take on one of the following data types:

- ◆ Boolean
- ◆ Integer
- ◆ Floating-point
- ◆ String
- ◆ Invalid

The programmer need not specify the type of any variable; WMLScript will automatically attempt to convert between the different types as needed. One other point to note is that WMLScript is not object-oriented (as Java or C++). Therefore, it is impossible to create user-defined data types programmatically.

## 9.5 Operators

**WMLScript** supports a variety of operators that support assignment operations, arithmetic operations, logical operations, string operations, comparison operations, and array operations.

### 9.5.1 Arithmetic Operators

The arithmetic operators perform common arithmetic calculations on variables and literals. The table given below lists the arithmetic operators and the operations that they perform.

Operation	Operator	Example	Result
Addition	+	10+20	30 ( Sum of 10 and 20)
Subtraction	-	20-10	10 ( Subtraction 10 from 20)
Multiplication	*	10*20	200(Multiplication of 10 and 20)
Division	/	5/2	2.5 ( Division of 5 by 2)
Integer Division	div	5 div 2	2 Integer part of Division. ( Ignoring the remainder)
Modulo Division	%	5 % 2	1 (Integer remainder of dividing 5 by 2)
Prefix Increment	++	++5*2	(5+1)*2= 12
Profix Increment	++	5++*2	(5*2)+1=11
Prefix Decrement	--	--5*2	* 2 = 8
Profix Decrement	--	5--*2	(5*2) -1 = 9

*Table : The program given below illustrates the use of arithmetic operators.*

#### 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 id="main" title="First Card">
```

```

    <p align="center">
        This is an example of Arithmetic Operators
        Enter the First Number
    <input name="fn" type="text" format="N*N" title="First Number"/>
        Enter the Second Number
    <input name="sn" type="text" format="N*N" title="Second Number"/>
        <do type="accept" label="Submit">
            <go href="arithmetic.wmls#arith($(fn),$(sn))"/>
        </do>
    </p>
</card>
</wml>

```

```

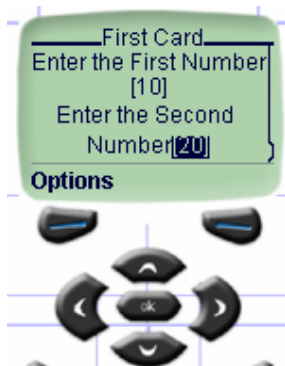
extern function arith(a,b)
{
    Dialogs.alert ("The First Number is" + a);
    Dialogs.alert ("The Secoind Number is" + b);

        var c,d,e,f,g,h,i,j;
        c = a+b;
    d = a-b;
    e = a*b;
    f = a/b;
    g = a div b;
    h = a % b;
    i = ++a * b;
    j = a-- * b;

    Dialogs.alert ("Addition is" + c);
    Dialogs.alert ("Subtraction is" + d);
    Dialogs.alert ("Multiplication is" + e);
    Dialogs.alert ("Division is" + f);
    Dialogs.alert ("Integer Division is" + g);
    Dialogs.alert ("Modulo Division is" + h);
    Dialogs.alert ("Prefix Increment is" + i);
    Dialogs.alert ("Postfix Decrement is" + j);
}

```

On executing the above program the following card will be on view



### 9.5.2 Logical Operators

Logical operators perform calculations on Boolean variables that have true or false values. Boolean values are used for decision-making.

WMLScript supports the basic logical operators:

Logical Operator	Meaning
&&	AND
	OR
!	NOT

Table : Logical Operator

### 9.5.3 Assignment Operators

The process by which a variable is given a value is called assignment. The assignment operator in WMLScript is = (Equals). But a variable has to be declared before it can be used.

#### Syntax

```
var variablename;
variablename = value;
or
var variablename = value;
```

There are also shorthand versions available for the arithmetic operators. The table below show the arithmetic operations and their shorthand versions.

Operator	Expression	Equivalent Expression
Assignment	a = 5	Setting 5 to the value of a.
Shorthand Addition	a += 10	a = a + 10
Shorthand Subtraction	a -= 10	a = a - 10
Shorthand Multiplication	a *= 10	a = a * 10
Shorthand Division	a /= 10	a = a / 10
Shorthand Integer Division	a div= 10	a = a div= 10
Shorthand Modulus	a %= 10	a = a %= 10
Shorthand AND	a &= 10	a = a & 10
Shorthand XOR	a ^= 10	a = a ^ 10
Shorthand OR	a  = 10	a = a   10

Table : Assignment shorthand operation

### 9.5.4 Comparison operators

Comparison Operators operate on members or strings and return a Boolean value. The boolean value is either true or false.

Operation	Expression	Boolean Value
Equal	<code>a == b</code>	True; if two operands are strictly equal.
Not Equal	<code>a != b</code>	True; if two operands are not strictly equal.
Greater than	<code>a &gt; b</code>	True if LHS operand is greater than RHS operand.
Greater Than Or Equal	<code>a &gt;= b</code>	True if LHS operand is greaterthan or equal to RHS operand.
Less than	<code>a &lt; b</code>	<i>True if LHS operand is less then RHS operand.</i>
Less than or equal	<code>a &lt;= b</code>	True if LHS operand is less than or equal to <i>RHS operand.</i>

### 9.5.5 Conditional Operator

WMLScript supports the conditional operator (?) which takes three operands. The operator selectively evaluates either the second operand or the third operand based on the boolean value of the first operand. If the value of the first operand (condition) is true then the result of the operation is the result of the evaluation of the second operand. If the value of the first operand is false or invalid then the result of the operation is the result of evaluation of the third operand.

#### Syntax

```
Result = conditional_expression ? expression1 : expression2
```

Here the conditional expression is evaluated. If it evaluates to true, then expression1 is evaluated and the result is assigned to the variable Result. Otherwise, the result of expression2 is assigned.

The following program shows how the conditional operator works.

#### Example 5.

```
<?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="main" title="Conditional Operator">
    <p align="center">
      First Number
      <input type="text" name="fn" format="N*N" title="First Number" />
      Second Number
      <input type="text" name="sn" format="N*N" title="Second Number" />
      <do type="accept" label="submit">
        <go href="conditional.wmls#con($(fn),$(sn))" />
      </do>
    </p>
  </card>
```



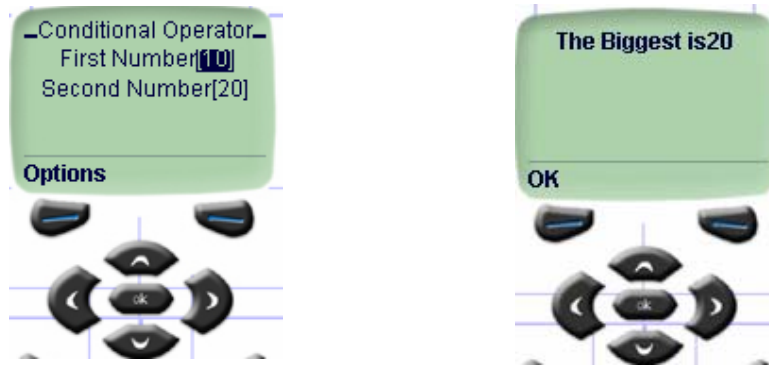
```

</wml>

extern function con(a,b)
{
  var c;
  c = a > b ? a : b;
  Dialogs.alert ("The Biggest is" + c);
}

```

In the above program if values 10 and 20 are assigned to the first and second numbers respectively the card would appear as shown below.



On the accepting these values the output would appear as shown above

### 9.5.6 Typeof operator

This operator returns a number representing the type of the variable which is not evaluated.

#### Syntax

```
typeof (variable)
```

The following table shows the types supported by the typeof operator and the values that will be returned depending on the typeof of the variable:

Type	Value returned by typeof
Integer	0
Float	1
String	2
Boolean	3
Invalid.	4

Table : Typeof Operator

The program given below is an illustration of the use of the typeof operator.

#### Example 5.

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

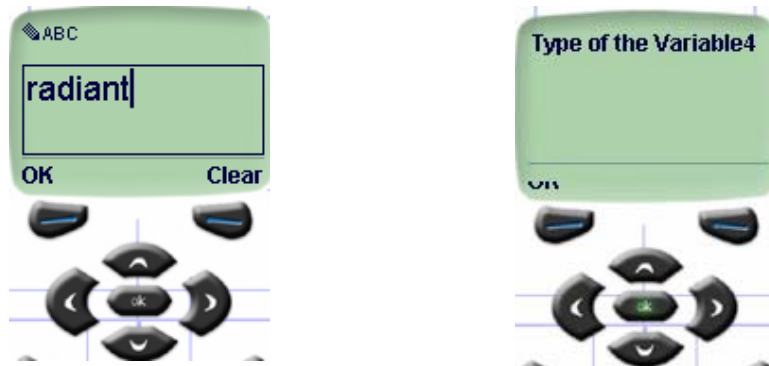
```

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="main" title="typeof Operator">
    <p>
      Enter input
      <input type="text" name="a"/>
      <do type="accept" label="submit">
        <go href="typeof.wmls#fun($(a))" />
      </do>
    </p>
  </card>
</wml>

extern function fun(a)
{
  Dialogs.alert ("Type of the Variable" + typeof(a));
}

```

The output given below shows the user input as “radiant” which is a string type of Operator



The result on accepting the above input is variable as 4 since a string type is not supported by the typeof operator

### 9.5.7 isvalid operator

This operator returns a boolean. It returns true if the variable is not of type invalid. i.e. it is valid on the other hand, it returns false if the variable is of type invalid. This is illustrated in the program given below.

#### Example 5.

```

<?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="main" title="isValid Operator">
    <p>
      Enter input
      <input type="text" name="a"/>
      <do type="accept" label="submit">
        <go href="isvalid.wmls#fun($(a))" />
      </do>
    </p>
  </card>
</wml>

```

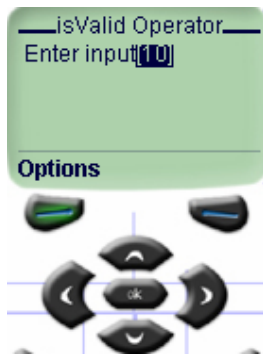
```

        </do>
    </p>
</card>
</wml>

extern function fun(a)
{
    Dialogs.alert (" a is Valid or Not ? " + isvalid(a));
}

```

The card below shows the user input



On submitting the input the result would be as shown below



### Operator Precedence

WMLScript follows the standard mathematical for evaluating expressions. Moreover precedence the order in which expressions are evaluated can be forced through the use of parentheses. The operators are listed in order of precedence in the following table. The precedence decreases on going down the table. The operators on the same row have the same precedence.

```

++ -- ~! + (unary)-(unary) typeof isvalid
• / div %
- +
<< >> >>>
< <= > >=
== !=
&
^
|
&&
||
?:
= *= -= /= %= div= += <<= >>= >>>= &= ^= |=

```



## 9.6 Short Summary

- WMLScript is based on JavaScript, but it has been modified to provide better support for low bandwidth communication and thin clients.
- WMLScript is based on JavaScript, but it has been modified to provide better support for low bandwidth communication and thin clients.
- The smallest unit of execution in WMLScript is a statement and each statement must end with a semicolon (;).
- 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** supports a variety of operators that support assignment operations, arithmetic operations, logical operations, string operations, comparison operations, and array operations.

## 9.7 Brain Storm

- 1 Discuss about the Language Syntax in WMLScript?
- 2 What are the different types of Data types used in WMLScript?
- 3 Discuss about the different Operators used in WMLScript?



## Lecture 10

---

# WML Script - Control and Functions

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ Control Constructs
- ✔ Functions
- ✔ Continue statement
- ✔ Break statement
- ✔ Return statement

---

## Coverage Plan

---

### Lecture 10

- 10.1 Snap Shot
- 10.2 Flow Control Statements
- 10.3 Functions
- 10.4 Break Statement
- 10.5 Continue Statement
- 10.6 Return Statement
- 10.7 Short Summary
- 10.8 Brain Storm

## 10.1 Snap Shot

This lecture describes about the control statements and the functions, which are often used in WML Script.

## 10.2 Flow Control Statements

Flow control statements allow programs to make decisions and progress according to the outcome of these decisions. In WMLScript a decision made based on the value of a boolean.

There are three main flow control statements in WMLScript:

- ◆ if-else statement
- ◆ while loop
- ◆ for loop

### 10.2.1 if – else statement

This statement is used to specify conditional execution of statements. This is the most common and straightforward of the flow control statements.

#### Syntax

```

if (conditional expression)
{
    Set of statements-true block;
}
else
{
    Set of statements-false block;
}

```

If the conditional expression evaluates to true then the true-block statements are executed. Otherwise the false-block statements are executed.

The use of if-else statement is shown in the program given below:

#### 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 id="main" title="Example of If">
    <p>
      Enter Two Numbers <br/>
      First Number
      <input type="text" name="fn" format="N*N" />
      <br/>
      Second Number
      <input type="text" name="sn" format="N*N" />
      <do type="accept" label="Submit">
        <go href="ifexample.wmls#func($(fn),$(sn))" />
      </do>
    </p>
  </card>
</wml>
extern function func(a,b)

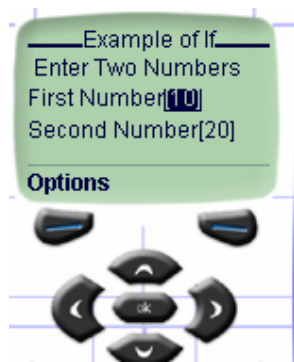
```

```

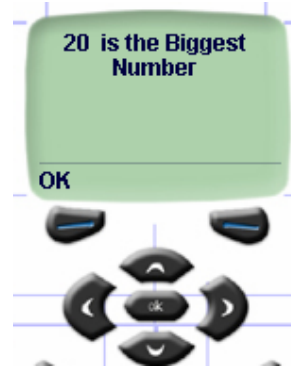
{
  if (a >= b)
  {
    Dialogs.alert (a + " is the Biggest Number");
  }
  else
  {
    Dialogs.alert (b + " is the Biggest Number");
  }
}

```

The card below shows the inputs of the two numbers that are to be compared



On submitting these values the outcome is shown below



### 10.2.2 while loop

This statement is used to create a loop that evaluates expression and, if it is true, a particular statement is executed. The loop repeats as long as the specified condition is true.

#### Syntax

```

while (expression)
{
  statement;
}

```

Expression can be any WMLScript expression that evaluates to a boolean or an invalid value. The condition is evaluated before each execution of the loop statement.

The program given below shows the use of the while loop.

#### 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 id="main" title="While loop example">
    <p>
      This is the Example of While loop...
    <br/><br/>
    Click to display 1 to 10

```



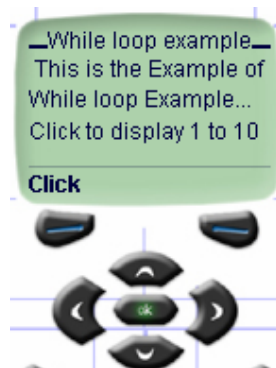
```

        <do type="accept" label="Click">
            <go href="while.wmls#fun()" />
        </do>
    </p>
</card>
</wml>
extern function fun()
{
    var i;
    i = 1;
    while (i<=10)
    {
        Dialogs.alert (i);
        i = i+1;
    }
}

```

On executing the code the display on the mobile device would look as follows.

The numbers 1 to 10 can be viewed one after another. Shown below is the number 3 on view



#### for loop statement

It is very useful to carry out an operation a set number of times. This statement consists of three optional expressions enclosed in parenthesis and separated by semicolons. This is followed by a statement to be executed in the loop.

#### Syntax

```

for (initial exp; conditionalexp; incremental exp)
{
    set of statements;
}

```

To setup a for loop, the following steps are necessary:

Initialization of the loop variable is executed when the loop first starts. Conditional expression that causes the loop to continue. If it is true then the statement block will be executed. If it is false then the loop will be terminated. Incremental expression is used to increment the loop variable.

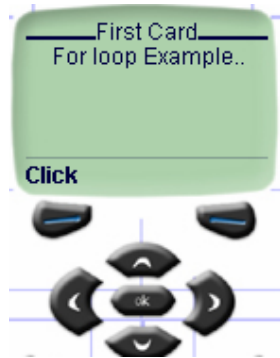
The following program illustrates the use of the for loop:

#### 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 id="main" title="First Card">
    <p align="center">
      For loop Example..
      <do type="accept" label="Click">
        <go href="forloop.wmls#fun()" />
      </do>
    </p>
  </card>
</wml>
extern function fun()
{
  var i;
  for (i=1; i<=10; i++)
  {
    Dialogs.alert (i);
  }
}

```



### 10.3 Functions

A WMLScript function is a named part of the WMLScript compilation unit that can be called to perform a specific set of statements and to return a value.

#### Function declaration

Function declaration can be used to declare a WMLScript function name with the optional parameters and a block statement that is executed when the function is called.

#### Characteristics of Functions

Function declaration cannot be nested. Function names should be unique within one compilation unit. All parameters to functions are passed by value.

Function calls must pass exactly the same number of arguments to the called function as specified in the function declaration. Function parameters behave like local variables that have been initialized before the body of the function is executed.

A function always returns a value. By default it is an empty string (""). However, a return statement can be specified to return other values.

#### Syntax

```
extern function identifier (parameter list)
{
    statement block;
return ( expression);
}
```

The extern keyword is used to specify that the function is externally accessible. Such functions can be called from outside the compilation unit in which they are defined. There must be atleast one externally accessible function in a compilation unit. This keyword is optional.

Identifier is the name specified for the function. Parameter is a comma-separated list of argument names. Block is the body of the function that is executed when the function is called and the parameters are initialized by the passed arguments. The return statement is optional. The default return value is an empty string.

## 10.4 Break statement

This statement is used to terminate the current while or for loop and continue the program execution from the statement following the terminated loop. It is an error to use break statement outside a while or a for statement.

#### Syntax

```
break;
```

The Example has been modified shown the use of the break statement.

#### 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 id="main" title="First Card">
<p align="center">
    For loop Example..
    <do type="accept" label="Click">
        <go href="break.wmls#fun()" />
    </do>
</p>
</card>
</wml>
extern function fun()
{
    var i;
    for (i=1; i<=10; i++)
    {
        if (i==5)
        {
            break;
        }
        Dialogs.alert (i);
    }
}
```

```
}
```

The above program would only display numbers from 1 to 4. On encountering number 5 the break statement executed. This leads to the termination of the for loop.

### 10.5 Continue Statement

This statement is used to terminate execution of a block of statements in a while or for loop and continue execution of the loop with the next iteration. Continue statement does not terminate the execution of the loop. It is an error to use continue statement outside a while or a for loop.

#### Syntax

```
continue;
```

#### Example

```
extern function conex()
{
    var i;
    for (i=1; i<=10; i++)
        {
            if (i%2 == 0)
                {
                    continue;
                }
            Dialogs.alert(i);
        }
}
```

The above code on execution would display the odd numbers between 1 and 10.

### 10.6 Return Statement

This is used inside the body of the function to specify the function return value. If no return statement is specified, the function returns empty string by default.

#### Syntax

```
return (expression);
```

The following program brings out the use of the return statement.

#### Example

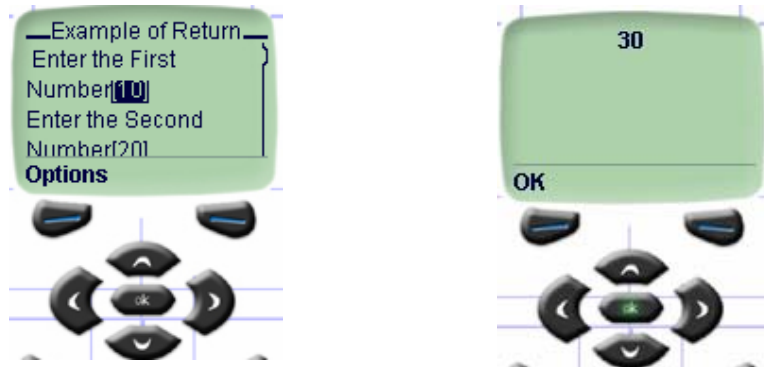
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="main" title="Example of Return">
    <p>
      Enter the First Number
      <input type="text" name="fn" format="N*N" />
      Enter the Second Number
      <input type="text" name="sn" format="N*N" />
      <do type="accept" label="Add">
        <go href="return.wmls#fun($(fn),$(sn))" />
      </do>
    </p>
  </card>
</wml>
extern function fun(a,b)
```

```

{
  var c;
  c = addition(a,b);
  Dialogs.alert (c);
}
extern function addition(a,b)
{
  return (a+b);
}

```

The card shown displays the use inputs. In this program there is call to a function named fun (a,b) which in turn calls another function named addition (a,b). This function invokes a return statement which returns the sum of a and b. This would be displayed as shown below.



### 10.7 Short Summary

A WMLScript function is a named part of the WMLScript compilation unit that can be called to perform a specific set of statements and to return a value.

Function calls must pass exactly the same number of arguments to the called function as specified in the function declaration.

### 10.8 Brain Storm

1. Discuss about the Control Statement in WML?
2. Explain about the Functions in WML?
3. What is Break Statement?
4. Define Continue Statement?

END

## Lecture 11

---

# WML Script - Library

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ Dialogs Library
- ✔ Float Library
- ✔ Lang Library

---

## Coverage Plan

---

### Lecture 11

- 11.1 Snap Shot
- 11.2 Libraries
- 11.3 Dialogs Library
- 11.4 Float Library
- 11.5 Lang Library
- 11.6 Short Summary
- 11.7 Brain Storm

## 11.1 Snap Shot

This lecture gives you a detailed description about the Libraries and their functions.

## 11.2 Libraries

Libraries are named collections of functions that belong logically together. These functions can be called by a dot (.) separator with the library name and the function name with parameters. So a function called `alert()` inside the dialogs library can be called by using `dialogs.alert(message)`.

There are six different libraries in WMLScript.

- Dialogs Library
- Float Library
- Lang Library
- String Library
- URL Library
- WML Browser Library

## 11.3 Dialogs Library

The functions in the Dialogs Library are quite similar to JavaScript alerts and message boxes. The `alert` function is used to display a message, the `confirm` function is used when the user should select between two answers (like "yes" or "no"), and the `prompt` function is used when the user should input an answer.

Functions	Description
<code>alert ()</code>	Displays a message, waits for a confirmation, and then returns an empty string
<code>confirm ()</code>	Displays a message, waits for an answer, and return a boolean value depending on the selected answer
<code>Prompt ()</code>	Displays a question, waits for an input, and then returns the user's answer

*Table : Dialog Library functions*

### **alert () function**

The **alert()** function displays the specified message, waits for a confirmation, and then returns an empty string.

### **Syntax**

```
Dialogs.alert(message)
```

Here is a string containing the message to be displayed on the WAP font device

### **Example**

```
extern function fun()  
{  
    Dialogs.alert ("Welcome");  
}
```



The output on the WAP device would be something like this



### **confirm () function**

The **confirm()** function displays a specified message, waits for an answer, and then returns a boolean value depending on answer that is selected. If the user selects *ok*, the return value is true, and if the user selects *cancel*, the return value is false.

#### **Syntax**

```
n = Dialogs.confirm(message, Ok, Cancel)
```

Here

**n** is the Boolean value returned from the function.

**message** is a string containing the message.

**Ok** is a string containing the OK text.

**Cancel** is a string containing the CANCEL text.

#### **Example**

```
extern function fun()
{
  Dialogs.confirm ("R U interested in making a Wireless world","ok","cancel");
}
```

If the above confirm function is executed, the following output would appear on the mobile device.



### **prompt() function**

The **prompt()** function displays a specified message and waits for an input. This function takes two parameters. The first parameter is the message to be displayed. The second parameter is a default input, that will be returned if the user does not enter a value. This function returns the user input or the default value.

Syntax

```
n = Dialogs.prompt(message, defaultinput);
```

Here

**n** is the String returned from the function,  
**message** is a string containing the message (question)  
**defaultinput** is a string containing the default input (answer)

Example

```
extern function fun()
{
    Dialogs.prompt("Enter yr name", "Radiant");
}
```

On executing the above function the output would appear as follows:



In this output the default string "Radiant" appears on the screen. If the user does not specify any other output this default string will be returned.

## 11.4 Float Library

The Float library contains a set of mathematical functions that is normally a part of the JavaScript's Math object. The Float library works only on those clients, which support floating-point numbers. If floating-point numbers are not supported, all functions would return *invalid*.

<i>Function</i>	<i>Description</i>
ceil()	Returns the nearest integer that is not smaller than a specified number
floor()	Returns the nearest integer that is not larger than a specified number
int()	Returns the integer part of a specified number
MaxFloat()	Returns the largest possible floating-point number
MinFloat()	Returns the smallest possible floating-point number
pow()	Raises a number to the power of a second number and returns the result
Round()	Returns the nearest integer to a specified number
sqrt()	Returns the square root of a specified number

---

Table : Float Library Functions

### **ceil() function**

The **ceil()** function returns the nearest integer that is not smaller than the *number* parameter value.

#### **Syntax**

```
Float.ceil(number)
```

Here

**n** is the integer returned from the function,

**number** represents the numeric constant that is given as input.

#### **Example**

```
1.      extern function fun()
        {   var n=Float.ceil(10.2);
            Dialogs.alert (n);
        }
```

This can equivalently be written as

```
Extern function fun( )
{
dialogs.alert(float.ceil(10.2));
}
```

It returns 11, as 11 is the nearest integer which is not smaller than 10.2

```
2.      extern function fun()
        {
            Dialogs.alert (Float.ceil(-10.2));
        }
```

It returns -10 which is the nearest integer that is not smaller than -10.2.

### **floor() function**

The **floor()** function returns the nearest integer that is not larger than the *number* parameter value.

#### **Syntax**

```
Float.floor(number)
```

Here

**number** is the numeric input.

#### **Example**

```
1.      extern function fun()
        {
            Dialogs.alert (Float.floor(10.2));
        }
```

It returns 10; as it is the nearest integer that is not larger than 10.2.

```
2.      extern function fun()
        {
        Dialogs.alert (Float.floor(-10.2));
        }
```

It returns -11, as it is the nearest integer that does not exceed -10.2.

### **int() function**

The **int()** function returns the integer part of the *number* parameter.

#### **Syntax**

**Float.int (number)**

Here

number is a **number (input)**

#### **Examples**

```
1.      extern function fun()
        {
        var n = Float.int(10.2)
        Dialogs.alert (n);
        }
        It returns 10, the integer part of 10.2.
```

```
2.      extern function fun()
        {
        Dialogs.alert (Float.int(-10.2));
        }
        It returns -10, the integer part of -10.2.
```

### **maxFloat() function**

The **maxFloat()** function returns the largest possible floating-point number (3.4028235E38).

#### **Syntax**

**Float.maxFloat**

#### **Example**

```
extern function fun()
{
    Dialogs.alert ("maxFloat =" + Float.maxFloat());
}
It returns maxFLOAT 3.4028235E38.
```

### **minFloat() function**

The **minFloat()** function returns the smallest possible floating-point number (1.4E-45).

#### **Syntax**

```
Float.minFloat()
```

**Example**

```
extern function fun()
{
    Dialogs.alert ("minFloat  =" + Float.minFloat());
}
```

It returns 1.4E-45.

**pow() function**

The **pow()** function raises the first parameter to the power of the second parameter and returns the result.

**Syntax**

```
Float.pow(number1, number2)
```

Here,

number1 and number 2 are numeric constant.

**Example**

```
extern function fun()
{
    Dialogs.alert ("pow  =" + Float.pow(2,3));
}
```

It returns pow 8 which is 2<sup>3</sup>.

**round() function**

The **round()** function returns the nearest integer to the *number* parameter.

**Syntax**

```
Float.round(number)
```

Here,

**number** is a numeric constant

**Example**

```
extern function fun()
{
    Dialogs.alert ("round  " + Float.round(10.22));
}
```

It returns round 10, which is the nearest integer to which the 10.22 is rounded.

**sqrt() function**

The **sqrt()** function returns the square root of the *number* parameter value.

### Syntax

Float.sqrt(number)

Here,

Number is a numeric constant for which the square root is to be calculated.

### Example

```
extern function fun()
{
    Dialogs.alert (Float.sqrt(9));
}
It returns 3.
```

The following code illustrates the use of seven of the Float library functions.

### Program

```
extern function fun()
{
    Dialogs.alert ("ceil +ve Number " + Float.ceil(10.2));
    Dialogs.alert ("ceil -ve number " + Float.ceil(-10.2));
    Dialogs.alert ("floor +ve number " + Float.floor(10.2));
    Dialogs.alert ("floor -ve number " + Float.floor(-10.2));
    Dialogs.alert ("maxFloat " + Float.maxFloat());
    Dialogs.alert ("minFloat " + Float.minFloat());
    Dialogs.alert ("pow " + Float.pow(2,3));
    Dialogs.alert ("round " + Float.round(10.22));
    Dialogs.alert ("sqrt " + Float.sqrt(9));
}
```

The output given below corresponds to the first statement in the above function. On clicking "OK", the subsequent statement can be output of the viewed.



## 11.5 Lang Library

The Lang Library contains functions that are used for absolute value calculations, data type manipulation, and random number generation.

<i>Function</i>	<i>Description</i>
abort()	Aborts a WMLScript and returns a message to the caller of the script
abs()	Returns the absolute value of a number
characterSet()	Returns the character set supported by the WMLScript interpreter
exit()	Exits a WMLScript and returns a message to the caller of the script
float()	Returns true if floating-point numbers are supported, and false if not
isFloat()	Returns true if a specified value can be converted into a floating-point number by the parseFloat() function, and false if not
isInt()	Returns true if a specified value can be converted into an integer by the parseInt() function, and false if not
max()	Returns the largest value of two numbers
maxInt()	Returns the maximum possible integer value
min()	Returns the smallest value of two numbers
minInt()	Returns the minimum possible integer value
parseFloat()	Returns a floating-point value defined by a string
parseInt()	Returns an integer defined by a string
random()	Returns a random integer between 0 and a specified number
seed()	Initializes the random number generator with a number, and returns an empty string

Table : Lang Library Functions

### abort() function

The **abort()** function aborts a WMLScript and returns a message to the caller of the script.

#### Syntax

```
Lang.abort ( text )
```

Here,

**text** is a string

#### Example

```
var errtxt = "Illegal value";
Lang.abort("Error in function: " + errtxt);
```

#### Result

The script is aborted, and the string "Error in function: Illegal value" is returned to the caller of the script.

### abs() function

The **abs()** function returns the absolute value of a number.

**Syntax**

```
Lang.abs (number)
```

Here,

◆ **number** is the numeric constant for which the absolute value is calculated.

**Example**

```
extern function fun()  
{  
    Dialogs.alert (Lang.abs(-10));  
}
```

It returns 10, which is the absolute value of -10.

**characterSet() function**

The **characterSet()** function returns an integer that defines the character set supported by the WMLScript interpreter.

**Syntax**

```
Lang.characterSet()
```

**Example**

```
extern function fun()  
{  
    Dialogs.alert (Lang.characterSet());  
}
```

If returns 1000.

**exit() function**

The **exit()** function ends a WMLScript and returns a value to the caller of the script.

**Syntax**

```
Lang.exit(value)
```

Here,

Value is any value

**Example**

```
Lang.exit ("value is 500");
```

**Result**

The script ends, and "Value is 500" is returned to the caller of the script.

**float() function**

The **float()** function returns a boolean value that is true if floating-point numbers are supported, and false if not.

**Syntax**

```
Lang.float()
```



**Example**

```
extern function fun()
{
    Dialogs.alert ("float " + Lang.float());
}
```

It returns float true.

**isFloat() function**

The **isFloat()** function returns a boolean value that is true if the *value* parameter can be converted into a floating-point number by the **parseFloat()** function, and false if not.

**Syntax**

```
Lang.isFloat(value)
```

**Example**

```
extern function fun()
{
    Dialogs.alert (Lang.isFloat(-10.8790));
}
```

It returns true.

**isInt() function**

The **isInt()** function returns a boolean value that is true if the *value* parameter can be converted into an integer by the **parseInt()** function, and false if not.

**Syntax**

```
Lang.isInt(value)
```

**Example**

```
extern function fun()
{
    Dialogs.alert (Lang.isInt(-10));
}
```

It returns true.

**maxInt() function**

The **maxInt()** function returns the maximum possible integer value.

**Syntax**

```
lang.maxInt()
```

**Example**

```
extern function fun()
{
    Dialogs.alert ("maxInt    " + Lang.maxInt());
}
```

It returns maxInt2147483647.

**min() function**

The **min()** function returns the smallest of two numbers.

**Syntax**

Lang.min(number1, number2)

Here number 1 and number2 are two numeric constants that are to be compared.

**Example**

```
extern function fun()
{
    Dialogs.alert (Lang.min(-10,29));
}
```

It returns -10, which is the minimum number.

**minInt() function**

The **minInt()** function returns the minimum possible integer value.

**Syntax**

Lang.minInt()

It returns -2147483647.

**Example**

```
extern function fun()
{
    Dialogs.alert (Lang.minInt());
}
```

It returns -2147483647.

**parseFloat() function**

The **parseFloat()** function returns a floating-point value defined by the *string* value.

The parsing ends on the first character that can not be parsed as a floating-point value.

**Syntax**

Lang.parseFloat(string)

Here,

String is a string of characters

**Example**

```
extern function fun()
{
    Dialogs.alert (Lang.parseFloat("-10.203 Radiant"));
}
```

It returns -10.203.

**parseInt() function**

The **parseInt()** function returns an integer defined by the *string* value. The parsing ends on the first character that is not a leading "-", "+", or a decimal digit.

**Syntax**

```
Lang.parseInt(string)
```

Here,

String is a string of characters.

**Example**

```
extern function fun()
{
    Dialogs.alert (Lang.parseInt("10 Venkatesh"));
}
```

It returns 10, which is the integer number in the given string.

**random() function**

The **random()** function returns a random integer between 0 and the *number* parameter specified in the function.

**Syntax**

```
Lang.random(number)
```

Here,

Number is a numeric constant specified by the user.

**Example**

```
extern function fun()
{
    Dialogs.alert (Lang.random(10*34));
}
```

It returns 277.

**seed() function**

The **seed()** function initializes the random number generator with the *number* parameter specified in the function, and returns an empty string.

**Syntax**

```
Lang.seed(number)
```

**Example**

```
extern function fun()
{
    Dialogs.alert ("seed      " + Lang.seed(-10));
}
It returns an empty string
```

The following code exemplifies the use of the lang library functions

**Program**

```
extern function fun()
{
    Dialogs.alert ("abs " + Lang.abs(-10));
    Dialogs.alert ("character set      " + Lang.characterSet());
    Dialogs.alert ("float " + Lang.float());
    Dialogs.alert ("isfloat      " + Lang.isFloat(-10.8790));
    Dialogs.alert ("isInt      " + Lang.isInt(-10));
    Dialogs.alert ("max " + Lang.max(2,-19.09));
    Dialogs.alert ("maxInt      " + Lang.maxInt());
    Dialogs.alert ("min " + Lang.min(-10,29));
    Dialogs.alert ("minInt      " + Lang.minInt());
    Dialogs.alert ("parseInt      " + Lang.parseInt("10 Venkatesh"));
    Dialogs.alert ("parseFloat "+ Lang.parseFloat("-10.204 Radiant"));
    Dialogs.alert ("random      " + Lang.random(10*34));
    Dialogs.alert ("seed      " + Lang.seed(-10));
}
}
```

The output shown corresponds to the first statement of the above function.



### 11.6 Short Summary

- The **prompt()** function displays a specified message and waits for an input. This function takes two parameters. The first parameter is the message to be displayed. The second parameter is a default input, that will be returned if the user does not enter a value.
- The Lang Library contains functions that are used for absolute value calculations, data type manipulation, and random number generation.

### 11.7 Brain Storm

1. Discuss briefly about Lang Library?
2. Explain about Float Library in brief Manner?
3. What is Dialog Library? Explain in brief Manner.



## Lecture 12

---

# WML Script - Library

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ String Library
- ✔ URL Library
- ✔ WML Browser Library

---

## Coverage Plan

---

### Lecture 12

- 12.1 Snap Shot
- 12.2 String Library
- 12.3 URL Library
- 12.4 WML Browser Library
- 12.5 Short Summary
- 12.6 Brain Storm

## 12.1 Snap Shot

In this lecture you are going to learn about some of the Library functions

## 12.2 String Library

The string library contains functions for the manipulation and conversion of string data. The table below lists the string library functions and their description in itself.

Function	Description
CharAt()	Returns a character that is placed in a specified position of a string
Compare()	Compare two strings, and returns an integer that tells if the one string is identical, smaller or larger than the other
elementAt()	Separates a string into elements, and then return a specified element
elements()	Returns the number of times a specified value appears in a string
find()	Returns the position of a substring in a string
format()	Formats a value, and returns the result
insertAt()	Separates a string into elements, inserts a substring, and then return the result
isEmpty()	Returns a boolean value that is true if the string is empty, and false if not
length()	Returns the length of a string
removeAt()	Separates a string into elements, removes an element, and then return the result
replace()	Replaces a part of a string with a new string, and return the result
replaceAt()	Separates a string into elements, replaces an element, and then return the result
squeeze()	Reduces all white spaces to single spaces, and returns the result
subString()	Returns a string that is a specified part of another string
toString()	Creates a string from a number, and return the result
trim()	Returns a string without leading and trailing spaces

Table : String Library Functions

### charAt () functions

The **charAt()** function returns a string. The string contains only one character. The character returned is the character that is placed in the position of the *index* parameter.

#### Syntax

```
String.charAt(string, index)
```

Here

String is a string, index is an integer

#### Example

```
extern function fun()
{
    var str;
    str = "Radiant Software";
```



```

        Dialogs.alert (String.charAt(str,2));
    }

```

It returns "d", as it is the third character of the input string.

### compare() function

The **compare()** function takes two parameters *string1* and *string2*. It returns 0 if the value of *string1* and *string2* are identical, -1 if the value of *string1* is smaller than *string2*, and 1 if the value of *string1* is larger than *string2*.

#### Syntax

```
String.compare(string1, string2)
```

Here,

String1 and String2 are strings that are to be compared.

#### Example

```

extern function fun()
{
    var str;
    str = "Radiant Software";
    Dialogs.alert (String.compare(str, "Radiant Software"));
}

```

It returns 0, as the two strings under consideration are identical.

### elements() function

The **elementAt()** function separates the value of the *string* parameter into elements (sub strings) according to the specified *separator* parameter, and then returns the element with the given index.

#### Syntax

```
String.elementAt(string, index, separator)
```

Here,

n is the string returned from the function,

String is the string to be parsed,

Index is an integer, defining the subString to be returned,

Separator is the separator that divides the strings into elements.

#### Examples

```

extern function fun()
{
    var str;
    str = "Radiant Software";
    Dialogs.alert
    (String.elementAt("Radiant,Software,chennai",2,""));
}

```

It returns "chennai", as chennai is the substring that is present at the second occurrence of the comma (,) separator.

### elements() function

The **elements()** function returns the number of elements that are separated by a specified separator in a given string.

**Syntax**

```
String.elements(string, separator)
```

Here,

String is a string that is separated in to elements by the separator.

**Example**

```
extern function fun()
{
    Dialogs.alert      (String.elements("Radiant      Software,
chennai", ","));
}
```

It returns 2 as there are two elements (substrings) in the given string that are separated by the comma separator.

**find() function**

The **find()** function returns the position of the first character in the *string* parameter that matches with the beginning character of the *substring* parameter.

**Syntax**

```
String.find(string, substring)
```

Here,

String is a string,

Substring is a string

**Example**

```
extern function fun()
{
    var str;
    str = "Radiant Software";
    Dialogs.alert (String.find(str,"Soft"));
}
```

It returns 8. The given substring "Radiant Software" begins with the letter S. The position first occurrence of this character in the string is 8.

format() function

The **format()** function formats a value to the format specified in the *formatspec* parameter.

**Syntax**

```
String.format(formatspec, value)
```

The syntax for format specification is % width.precision type. Here width (optional) specifies the minimum number of characters to be printed, precision (optional)sets the precision of the output. The default precision for integer type is 1. The default number of digit after the decimal point for floating point type is 6.

Part	Description
Formatspec	<p>The syntax of the format specification:</p> <p>% width.precision type</p> <p>width - Optional. Specifies the minimum number of characters printed.</p> <p>precision - Optional. Sets the precision of the output value. Can take one of the following values:</p> <p>d        The minimum number of digits to print. Default is 1</p> <p>f        The number of digits after the decimal point. Default is 6</p> <p>s        The maximum number of characters to print. By default, all characters are printed</p> <p>type - Required. Determines how to interpret the formatted value. Can take one of the following values:</p> <p>d        Integer - the output value has nd digits</p> <p>f        Floating-Point - the output value has nf decimal digits</p> <p>s        String - Characters are printed to the end of the string or until the precision value is reached</p>
Value	Any value

Table: Format specification

The default the characters of a string are printed type (mandatory) the type of the value. d stands for integer type, f for floating-point and s for string type of data.

#### Example

```
extern function fun()
{
    Dialogs.alert (String.format("%1f", 10.123423223));
}
```

It returns 10.123424.

#### insertAT() function

The **insertAt()** function separates the *string* parameter into elements according to the specified *separator* parameter, and then inserts the value of the *substring* parameter at the given index position.

#### Syntax

String.insertAt(string, substring, index, separator)

#### Example

```
extern function fun()
{
    Dialogs.alert (String.insertAt("Radiant Software Ltd", "WAP", 1, " "));
}
```

It returns "Radiant WAP Software Ltd" . Here the separator is a white space. Since the index is the substring "WAP" is inserted at the first occurrence of the separator.

### **isEmpty() function**

The **isEmpty()** function returns a boolean value that is true if the string is empty, and false if not.

#### **Syntax**

`String.isEmpty(string)`

#### **Example**

```
extern function fun()
{
    var str;
    str = "Radiant Software";
    Dialogs.alert (String.isEmpty(str));
}
```

It returns false. Here since the string *str* is not empty the value returned is *false*.

### **length() function**

The **length()** function returns the length of a string.

#### **Syntax**

`String.length(string)`

Here,

String is a string

#### **Example**

```
extern function fun()
{
    var str;
    str = "Radiant Software";
    Dialogs.alert (String.length(str));
}
```

It returns 16, which is the length of the string *str*,

### **removeAt() function**

The **removeAt()** function separates the value of the *string* parameter into elements according to the specified *separator* parameter, and then removes the element at the given index.

#### **Syntax**

`String.removeAt(string, index, separator)`

#### **Example**

```
extern function fun()
{
    var str;
    str = "Radiant Software";
    Dialogs.alert (String.removeAt("Radiant
Software,Chennai",1,""));
}
```

It returns "Radiant Software". Here the specified separator is a ",". The index value is 1 which indicates that the string that is positioned at the first occurrence of the separator is to be removed.

### replace() function

The **replace()** function replaces a part of a string with a new string, and returns the result.

#### Syntax

```
String.replace(string, oldstring, newstring)
```

Here

String is the given string OldString is a substring of string that is to be replaced, Newstring is a string that replaces the oldstring

#### Example

```
extern function fun()
{
  Dialogs.alert (String.replace("Radiant Software","Software","Chennai"));
}
It returns "Radiant Chennai".
```

Here, the substring *Software* is replaced by the substring *Chennai*.

### replaceAt() function

The **replaceAt()** function separates the value of the *string* parameter into elements according to the specified *separator* parameter, and then replaces the element at the given index with the *substring* parameter.

#### Syntax

```
String.replaceAt(string, substring, index, separator)
```

#### Example

```
extern function fun()
{
  Dialogs.alert
  (String.replaceAt("Radiant,Software","Chennai",1,""));
}

```

It returns "Radiant,Chennai". Here the separator is a comma ",". The index value 1 indicates that at the first occurrence of the separator is to be replaced by the substring "chennai" elements (substring,"software").

### squeeze()

The **squeeze()** function reduces all white spaces to single spaces, and returns the result.

#### Syntax

```
String.squeeze(string)
```

**Example**

```
extern function fun()
{
    Dialogs.alert (String.squeeze("Radiant Software Ltd.    Chennai"));
}
```

It returns "Radiant Software Ltd. Chennai". Here the extra whitespaces have been removed and the resultant string is returned.

**subString()**

The **subString()** function returns a substring that is a specified part of the given *string*.

**Syntax**

```
String.subString(string, start, length)
```

Here

**string** is the given string,

**start** specifies the position of the first character of the substring on the given string

**length** specifies the member of characters that from the substring.

**Example**

```
extern function fun()
{
    var str;
    str = "Radiant Software";
    Dialogs.alert (String.subString(str,8,4));
}
It returns "Soft".
```

The above function specifies the length of the substring as 4 and the position of the first character of 8.

**toString() function**

The **toString()** function creates a string of a value.

**Syntax**

```
String.toString(value)
```

**Example**

```
extern function fun()
{
    var str;
    str = 10;
    Dialogs.alert (String.toString(str));
}
It returns "10".
```

**trim() function**

The **trim()** function returns a string without leading and trailing spaces.

#### Syntax

String.trim(*string*)

#### Example

```
extern function fun()
{
Dialogs.alert (String.trim("    Radiant    Software    Ltd    "));
}
```

It returns "Radiant Software Ltd". Here the heading and trailing spaces have been removed to give the resultant string.

The difference between the squeeze() function and the trim() function is that while the squeeze() function removes all the whitespaces leading and trailing whitespaces.

#### Program

```
extern function fun()
{
var str;
str = "Radiant Software";
Dialogs.alert (String.charAt(str,2));
Dialogs.alert (String.compare(str,"Radiant Software"));
Dialogs.alert (String.elementAt("Radiant,Software,chennai",2,","));
Dialogs.alert (String.elements("Radiant Software, chennai",","));
Dialogs.alert (String.find(str,"Soft"));
Dialogs.alert (String.format("%1f", 10.123423223));
Dialogs.alert (String.insertAt("Radiant Software Ltd","WAP",1," "));
Dialogs.alert (String.isEmpty(str));
Dialogs.alert (String.length(str));
Dialogs.alert (String.removeAt("Radiant Software,Chennai",1,","));
Dialogs.alert (String.replaceAt("Radiant,Software","Chennai",1,","));
Dialogs.alert (String.squeeze("Radiant Software Ltd.    Chennai"));
Dialogs.alert (String.substring(str,7,5));
Dialogs.alert (String.toString(str));
Dialogs.alert (String.trim("    Radiant    Software    "));
}
```

On compiling the above program the outputs the for all the statements can be viewed one after another. The output given below corresponds to the last statement.



This output corresponds to the last statement of the previous program. By scrolling, the outputs of the previous statements can be viewed.

### 12.3 The URL Library

The URL library contains a set of functions for handling relative and absolute URLs.

Function	Description
<code>escapeString()</code>	Replaces special characters in a URL with an escape sequence, and return the result
<code>getFragment()</code>	Returns the fragment in a URL
<code>getHost()</code>	Returns the host specified in a URL
<code>getParameters()</code>	Returns the parameters in the last path segment of a URL
<code>getPath()</code>	Returns the path specified in a URL
<code>getPort()</code>	Returns the port number specified in a URL
<code>getQuery()</code>	Returns the query part in a URL
<code>getScheme()</code>	Returns the scheme in a URL
<code>isValid()</code>	Returns true if a URL has the right syntax, and false if not
<code>resolve()</code>	Returns an absolute URL from a base URL and a relative URL
<code>unescapeString()</code>	Replaces the escape sequences in a URL with characters, and return the result

Table : URL library functions

#### **escapeString()**

The **escapeString()** function replaces special characters in a URL with an escape sequence, and returns the result.

#### **Syntax**

```
URL.escapeString(url)
```

#### **Example**

```
extern function fun()
{
    Dialogs.alert
    (URL.escapeString("http://www.radisoft.com/wml/first.wml"));
}
It returns as "http%3A%2F%2Fwww.radisoft.com%2Fwml%2Ffirst.wml"
```

The special characters : and / have been replaced by the corresponding escape sequences.

#### **getFragment()function**

The **getFragment()** function returns the fragment in the *url* parameter.

#### **Syntax**

```
URL.getFragment(url)
```



**Example**

```
extern function fun()
{
    Dialogs.alert
    (URL.getFragment("http://www.radisoft.com/wml/first.wml#firstcard"));
}
```

It returns "firstcard" which is the fragment in the given URL.

**getHost() function**

The **getHost()** function returns the host specified in the *url* parameter.

**Syntax**

URL.getHost(*url*)

**Example**

```
extern function fun()
{
    Dialogs.alert
    (URL.getHost("http://www.radisoft.com/wml/first.wml"));
}
```

The result would be "radisoft.com" which is the host specified in the URL.

**getParameters() function**

The **getParameters()** function returns the parameter in the last path segment of the *url* parameter.

**Syntax**

URL.getParameters(*url*)

**Example**

```
extern function fun()
{
    Dialogs.alert
    (URL.getParameters("http://www.radisoft.com/wml/first.wml"));
}
```

The result in this case would be an empty string as there is no parameter in the last path of the URL.

**getPath() function**

The **getPath()** function returns the path specified in the *url* parameter.

**Syntax**

URL.getPath(*url*)

**Example**

```
extern function fun()
{
    Dialogs.alert (URL.getPath("http://www.radisoft.com/wml/first.wml"));
}
```

It returns /wml/first.wml which is the path specified in the URL.

**getPort() function**

The **getPort()** function returns the port number specified in the *url* parameter.

**Syntax**

`URL.getPort(url)`

**Example**

```
extern function fun()
{
    Dialogs.alert
    (URL.getPort("http://www.radisoft.com/wml/first.wml"));
}
```

The output is 80 which is the port number specified in the URL.

**getQuery() function**

The **getQuery()** function returns the query part in the specified URL.

**Syntax**

`URL.getQuery(url)`

**Example**

```
extern function fun()
{
    Dialogs.alert
    (URL.getQuery("http://www.radisoft.com/wml/first.wml"));
}
```

It returns an empty string as there is no query in the URL.

**isValid() function**

The **isValid()** function returns a boolean value that is true if the specified URL has the right syntax, and false if not.

**Syntax**

`URL.isValid(url)`

**Example**

```
extern function fun()
{
    Dialogs.alert (URL.isValid("http://www.radisoft.com/wml/first.wml"));
}
```

It returns true as the given URL has the right syntax.

### **unescapeString() function**

The **unescapeString()** function replaces the escape sequences in a URL with characters, and return the result.

#### **Syntax**

URL.unescapeString(url)

#### **Example**

```
extern function fun()
{

Dialogs.alert
(URL.unescapeString("http%3A%2F%2Fwww.radisoft.com%2Fwml%2Ffirst.wml
"));
}
```

The result would be

<http://www.radisoft.com/wml/first.wml>. Here the escape sequence are replaced by the special characters.

The following program incorporates all the URL library functions.

#### **Example**

```
extern function fun()
{
Dialogs.alert (URL.escapeString("http://www.radisoft.com/wml/first.wml"));
Dialogs.alert (URL.getFragment("http://www.radisoft.com/wml/first.wml#firstcard"));
Dialogs.alert (URL.getHost("http://www.radisoft.com/wml/first.wml"));
Dialogs.alert (URL.getParameters("http://www.radisoft.com/wml/first.wml"));
Dialogs.alert (URL.getPath("http://www.radisoft.com/wml/first.wml"));
Dialogs.alert (URL.getPort("http://www.radisoft.com/wml/first.wml"));
Dialogs.alert (URL.getQuery("http://www.radisoft.com/wml/first.wml"));
Dialogs.alert ("valid" + URL.isValid("http://www.radisoft.com/wml/first.wml"));
Dialogs.alert
(URL.unescapeString("http%3A%2F%2Fwww.radisoft.com%2Fwml%2Ffirst.wml"));
}
```

The output given below corresponds to the first statement of the above example. The outputs of the subsequent statements can be obtained by clicking "ok".



## 12.4 WMLBrowser Library

Information stored by the browser is often called the browser context. The browser context can contain variables set by a program and variables set by the browser, like the current URL, and the card history stack.

The WML Browser library contains functions to set and access the browser context: These functions are listed in the table below:

Function	Description
<code>getCurrentCard()</code>	Returns the (relative) URL of the current card
<code>getVar()</code>	Returns the value of a variable
<code>go()</code>	The browser goes to a new card, specified by the new URL, and returns an empty string
<code>newContext()</code>	Clears all variables, and returns an empty string
<code>prev()</code>	The browser goes back to the previous card, and returns an empty string
<code>refresh()</code>	Refreshes the current card, and returns an empty string
<code>setvar()</code>	Sets the value of a variable, and returns true if the new value where implemented successfully, and false if not

### Note :

- The WML specification states that calls to library functions that are not supported by the browser should return *invalid*. Because of this, all the above functions should be tested against their return value, and proper action should be taken in case a function returns *invalid*.

### getVar()

The `getVar()` function returns the value of the specified *variable* in the browser context. If the variable does not exist this function returns an empty string ("").

### Syntax

```
WMLBrowser.getVar(variable)
```

### Example

### Result

```
a = "52"
```

**newContext()**

The **newContext()** function clears all variables of the WML context, and returns an empty string.

**Syntax**

```
WMLBrowser.newContext()
```

**Example**

```
var a = WMLBrowser.newContext();
```

**Result**

```
a = ""
```

**prev() function**

The **prev()** function tells the WML browser to go back to the previous WML card, and returns an empty string.

**Syntax**

```
WMLBrowser.prev()
```

**Example**

```
var a = WMLBrowser.prev();
```

**Result**

```
a = ""
```

**setVar()**

The **setVar()** function sets the value of the specified *variable* in the browser context, and returns true if the new value were implemented successfully, and false if not

**Syntax**

```
WMLBrowser.setVar(variable, value)
```

**Example**

```
var a = WMLBrowser.setVar("weeks",  
34);
```

**Result**

```
a = true
```

**JavaScript vs. WMLScript**

The following table summarizes the differences between JavaScript and WMLScript:

Feature	WMLScript	JavaScript
Semicolons	Mandatory	Optional
Supports HTML Comments	No	Yes

Non-escape characters preceded by a backslash	No	Yes
Names may include \$ characters	No	Yes
Supports global variables	Yes	Yes
Supports automatic declarations	Yes	Yes
Supports the generic term number	Yes	No
Specifies maximum and minimum values for integers	Yes	No
Supports double-precision floating-point values	No	Yes
Supports objects	No	Yes
Supports arrays	No	Yes
typeof returns	an integer	a string
Supports for..in	No	Yes
Supports with	No	Yes
Supports Libraries	Yes	No
Supports invalid	Yes	No
Supports isvalid	Yes	No
>Supports delete	No	yes
Supports void	No	Yes
Supports div	Yes	No
Supports extern	Yes	No
Supports pragmas	Yes	No

## 12.5 Short Summary

- The **charAt()** function returns a string. The string contains only one character. The character returned is the character that is placed in the position of the *index* parameter.
- The **insertAt()** function separates the *string* parameter into elements according to the specified *separator* parameter, and then inserts the value of the *substring* parameter at the given index position.
- The difference between the `squeeze()` function and the `trim()` function is that while the `squeeze()` function removes all the whitespaces leading and trailing whitespaces.
- The WML specification states that calls to library functions that are not supported by the browser should return *invalid*. Because of this, all the above functions should be tested against their return value, and proper action should be taken in case a function returns *invalid*.

## 12.6 Brain Storm

1. What is String Library? What are the different functions of the String Library?
2. Explain briefly about URL Library?
3. Discuss in detail about Web Browser Library?

## Lecture 13

---

# ASP and Object Model

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✧ Introduction of ASP
- ✧ ASP object Model

---

## Coverage Plan

---

### **Lecture 13**

- 13.1 Snap Shot
- 13.2 ASP Object Model
- 13.3 Short Summary
- 13.4 Brain Storm



## 13.1 Snap Shot

ASP is a Technology from Microsoft that provides the capability for the Web Server to process application logic and then deliver standard HTML to the client browser. The Results can then be delivered to a variety of client web technologies, such as standard HTML, or WML.

## 13.2 ASP Object Model

### The fundamental ASP objects

ASP 2.0 has six objects. These objects have their own predefined functionality and their own respective properties, methods and events.

- Request object
- Response object
- Session object
- Application object
- Server object
- Object context object

### The Request Object

The request object is responsible for retrieving information from the Web browser. The Request object is filled with various types of collections, properties, and methods that provide many ways to retrieve information from the user.

#### Collections

There are five types of collections in this Request Object.

- ◆ Query String
- ◆ Form
- ◆ Server Variables
- ◆ Cookies
- ◆ Client Certificate
- Query String Collection

The querystring collection is used by the request object to extract variables from the HTTP querystring. When sending a request, the client can include name/value pairs of information within the URL, after the file name. This collection stores any values provided in the URL. It receive the values from the client which is send by the user through get method.

- **Form Collection**

The Request object form collection helps to facilitate data retrieval from HTML forms. The Form collection captures the value from an HTML that submits information via the HTTP Post method.

- **The server variable Collections**

This server variable collection is used to obtain server environmental variables. These server variables are not specific to the server and can obtain information from the requesting client.

- **Cookies**

Cookie is nothing but some information about user stored in Client machine. The cookies collection is used by the Request object to retrieve values stored in text files on the client's machine.

- **Client Certification Variables**

It is used to provide proper security identification across unsecured environments. Privacy is used to prevent unwanted parties from viewing information on the net.

### **Authentication**

That the information can safely pass between points, it necessary to verify that the users are they say they are.

To deal with these security concerns, the clientcertificate collection used to retrieve the certification fields issued by the Web Browser.

### **The Request Object Property**

The Request object has only one property called TotalBytes. The totalbytes property is used to return the number of bytes sent by the browser. To transfer this information to the server, the client uses the POST method to send form information in the HTML body tags.

### **Request Object Method**

There is only one method called BinaryRead Method used to read binary information that is sent to the server from a Post request.

Request Object

<b>Collections</b>	<b>Syntax</b>	<b>Description</b>
1. QueryString	Request.QueryString("var")	Extract variables from the HTTP QueryString
2. Form	Request.Form("var")	Data retrieval from HTML forms, send through post method
3. Servervariables	Request.servervariables("key")	Retrives the values of the variable in the HTTP querystring.
4. Cookies	Request.cookies("var")	Retrives the values of cookies sent in an HTTP request.
5. Client Certification	Request.clientcertificate("key")	Digital certification variables.

<b>Property</b>	<b>Syntax</b>	<b>Description</b>
1. TotalBytes	Request.Totalbytes	Total amount of bytes sent in the request.

Method	Syntax	Description
1. BinaryRead	Request.binaryread("var")	Accept and store data sent from a browser.

### The Response Object

This Response Object is responsible for sending output from the server to the requesting client. The three most common methods of this object are write, redirect, and response methods. Write method used to send HTTP output to the client and the redirect method provides the capability to redirect the client browser to a different URL.

Collection	Syntax	Description
1. Cookie	Response.cookies("var")	Set the cookies in the client's machine.

Properties	Syntax	Description
1. Expire	Response.Expires ("min")	Sets the amount of time in which the page cached on the browser expire.
2. ExpiresAbsolute	Response.ExpiresAbsolute("dateatime")	Sets the date and time in which a page cached on a browser expires.
3. ContentType	Response.contentType "MIME type"	Indicates the media type of the body of the response.
4. CharSet	Response.charset "charsettype"	Indicates the character set for interpretation.
5. Buffer	Response.buffer = "true/false"	Sets whether or not page output is buffered.
6. Status	Response.status	Indicates the status line returned by the server.
7. Isclientconnected	Response.isclientconnected	Determines whether the browser has disconnected from the server.
8. PICS	Response.pics "var"	Adds the value of a PICS label to the pics-label field of the response header.
9. Cachecontrol	Response.cachecontrol "public/private"	Enables dynamic output of ASP to be cached by proxy servers.

Method	Syntax	Description
1. Write	Response.write "msg"	Writes a string to the current HTTP output.
2. BinaryWrite	Response.binarywrite "msg"	Writes information to the current HTTP output without any character conversion.
3. Clear	Response.clear	Erases any buffered HTML

4. Flush	Response.flush	outout. Bypasses buffering and sends output immediately to client.
5. End	Response.end	Stops the Web server from processing the script and return the current result to the client.
6. AppendtoLog	Response.appendtolog "msg"	Writes a string to the end of the web server log entry.
7. Addheader	Response.addheader "msg"	Writes a string to the HTML header.
8. Redirect	Response.redirect "url"	Attempts to automatically route the browser to URL.

### The session Object

This session object used to pass the values from one page to another page for the particular user. The session variables can easily pass through all pages.

### Declaring session Variables

Session variables are declared in a manner similar to declaring application variables.

Syntax

```
Session(varName)
```

Where varName is the name of the session-level variable. The session-level variables are only available to the user session that created the object and cannot be shared among separate user sessions.

### Example

```
Session("A") = 10
```

Collectons	Syntax	Description
1. Contents	Session.contents("var") = value	Contains all session level variables that have not been created with the <OBJECT> tag.
2. StaticObjects	Session.staticobjects ("name") = object	Contains all session-level objects declared with the <OBJECT> tag.

Properties	Syntax	Description
1. Sessionid	Session.sessionid	Returns the unique user session identifier.
2. timeout	Session.timeout "min"	Returns or sets the user timeout value in minutes.
3. LCID	Session.LCID "identifier"	Sets the local identifier used to set local date, currency, and time formats.

Methods	Syntax	Description
1. Abandon	Session.abandon	Explicitly disconnect the session.
2. Remove	Session.remove("var")	Remove the particular session variable.
3. RemoveAll	Session.removeall	Remove all the session variables.

Events	
1. SessionOnStart	Fires when the new client enters into the site.
2. SessionOnEnd	Fires when the client disconnected from the site.

### Server Object

The server object is responsible for the administrative functionality of the server. One of the most exciting aspects of the Server object is the capability to create an instance of a server via its Createobject method.

Properties	Syntax	Description
1. Scripttimeout	Server.scripttimeout = "sec"	Sets the default time of the execution of the script.

Methods	Syntax	Description
1. Createobject	Server.createobject "object"	Creates a server instance of an object.
2. Mappath	Server.Mappath	Translates the web server's virtual path settings to the physical path on the server.
3. HTML Encode	Server.HTML Encode "url"	Utilizes HTML encoding to deliver text to browser.
4. URL Encode	Server.URL Encode "url"	Utilizes URL encoding techniques.

### The Object Context Object

This is used by ASP to control transaction processing using the Microsoft Transaction Server (MTS). This direct processing within the MTS environment enables ASP to explicitly control, commit, and rollback features of objects managed by MTS.

### The Application Object

The Application Object is used to manage all information in the ASP application. The information can be accessed and passed between different users in the application. The application object enables locking and unlocking of its variables. When the variable locked, others cannot modify the properties of an Application Object.

Collectoins	Syntax	Description
1. Contents	Application.contents("var") = value	Contains all application level variables.
2. StaticObjects	Application.staticobjects ("obj") = "object"	Contains all application-level objects.

Methods	Syntax	Description
1. Remove	Application.remove ("var")	Remove the particular application variable.
2. RemoveAll	Application.removeall	Remove all the application variables.
3. Lock	Application.lock	Lock the variable for the particular user.
4. Unlock	Application.unlock	Unlock the application variable.

### Program

```

<%@ Language = "Vbscript" %>
<html>
<head>
<title>This is Application and Session Object Example program</title>
</head>
<body>
<%
    Application("a") = "Radiant"
    Application("b") = "Software"
    A = "Chennai"
%>
</body>
</html>

```

Save it as asp file. Try to display these application variables from another program.

### Global.ASA file

The Global.ASA file is an optional file that is used to manage the Application object. It has no displayable contents and has only four events that are Application\_onStart, Application\_onEnd, Session\_onStart and Session\_onEnd.

The Global.ASA file is an optional file that does not have to be used when creating ASP applications. If the file exists, the Application and Session events can be processed and the Application and Session events cannot be trapped.

The Global.ASA file is primarily accessed based on session-level events, and is called in three situations:

- When the Application OnStart or OnEnd event is triggered.
- When the Session\_Onstart or Session\_onEnd event is triggered.
- When a reference is made to an object that has to be instantiated in the Global.ASA file.

### 13.3 Short Summary

- ASP is a Technology from Microsoft that provides the capability for the Web Server to process application logic and then deliver standard HTML to the client browser.
- Cookie is nothing but some information about user stored in Client machine.
- Response Object is responsible for sending output from the server to the requesting client. The three most common methods of this object are write, redirect, and response methods. Write method used to send HTTP output to the client and the redirect method provides the capability to redirect the client browser to a different URL.

### 13.4 Brain Storm

1. What are the fundamental ASP Object?
2. What are the different types of Collections?
3. Explain all the methods used in the Response Object?
4. What is Server Object?

❧

## Lecture 14

---

# WAP MIME Types

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ MIME types
- ✔ Receiving values from WML pages
- ✔ Response to the WAP client



---

## Coverage Plan

---

<b>Lecture 14</b>
-------------------

- |   |
|---|
| <ul style="list-style-type: none"><li>14.1 Snap Shot</li><li>14.2 Mime Types</li><li>14.3 Request &amp; Response Object</li><li>14.4 Short Summary</li><li>14.6 Brain Storm</li></ul> |
|---|

## 14.1 Snap Shot

This lecture gives you a good idea about the MIME Types and the Request and Response Object

## 14.2 MIME Types

The IIS will forward the contents of an ASP file to the browser using the MIME type for HTML or WML. MIME (Multipurpose Internet Mail Extensions) is a specification for the format of data that can be sent over the Internet. The following are the MIME type of WML.

Content	MIME	Extention
WML	text/vnd.wap.wml	Wml
Compiled WML	application/vnd.wap.wmlc	Wmlc
WMLScript	text/vnd.wap.wmlscript	Wmls
Compiled WMLScript	application/vnd.wap.wmlscriptc	Wmlsc
Wireless (WBMP) bitmap	image/vnd.wap.wbmp	wbmp

When the server sends the data in response to a request it receives, it sends a MIME type with it. This MIME type can also be explicitly set by the application. Normally the file extension of the requested file is associated with a MIME type and so the server automatically issues the correct MIME type. Then, when the browser receives the information from the server, it checks it MIME type to see what to do with it.

Now, as with static HTML content, static WML content has very limited scope for providing services. For more complex applications, dynamic generation of pages is a must. This essentially means deploying an application that will dynamically generate pages from a database, which keeps maintains to a minimum. These applications are typically hosted on application servers. To include WAP support, we need to do is to configure them to use the above MIME types.

### Program

The Plain WML file

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
  <card id="first" title="Radiant Software Ltd">
    <p>
      Welcome to Radiant family...
    </p>
  </card>
</wml>
```

To turn this into an ASP file we just need to give it a .asp extension and add the following line of code:

```
<% Response.ContentType = "text/vnd.wap.wml" %>
<! DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
```

```
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
<card id="first" title="Radiant Software Ltd">
  <p>
    Welcome to Radiant family...
  </p>
</card>
</wml>
```

or

```
<% Response.ContentType = "text/vnd.wap.wml" %>
<! DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
<card id="first" title="Radiant Software Ltd">
  <p>
    <% Response.write "Welcome to Radiant family..." %>
  </p>
</card>
</wml>
```

The <% and %> characters delimit sections of script in an ASP file, and within these delimiters we are setting the MIME type.

### 14.3 Request and Response Objects

The Request Object used to receive the information that a user agent sends to the web server. In WAP, data entered by a user in an application can be sent back to the server using HTTP GET or POST. If the GET method used, then the parameters are accessible in ASP via the Request object as a collection called Querystring.

#### Program

```
<! DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
<card id="first" title="Radiant Software Ltd">
  <p>
    Enter your Name
  <input type="text" name="sn" />
    <anchor>Submit
    <go href="http://www.radisoft.com/welcome.asp" method="get">
    <postfield name="a" value="$(sn)"/>
  </go>
```

```
</anchor>
</p>
</card>
</wml>
```

In asp program,

```
<% Response.ContentType="text/vnd.wap.wml" %>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
<card id="Main" title="Radiant Software Ltd">
<p>
Welcome <%= Request.QueryString("a") %>
</p>
</card>
</wml>
```

#### 14.4 Short Summary

- When the server sends the data in response to a request it receives, it sends a MIME type with it. This MIME type can also be explicitly set by the application.
- In WAP, data entered by a user in an application can be sent back to the server using HTTP GET or POST. If the GET method used, then the parameters are accessible in ASP via the Request object as a collection called Querystring.

#### 14.5 Brain Storm

1. Explain briefly about the MIME types?
2. How do you receive values from WML pages?

☺☺☺

## Lecture 15

---

# Introduction to ADO

---

## Objectives

After completing this Lecture, you should be able to do the following

- ✔ Cookies
- ✔ ADO Introduction
- ✔ ADO Objects

---

## Coverage Plan

---

<b>Lecture 15</b>	
15.1	Snap Shot
15.2	Cookies
15.3	ADO Object Model
15.4	Short Summary
15.5	Brain Storm

## 15.1 Snap Shot

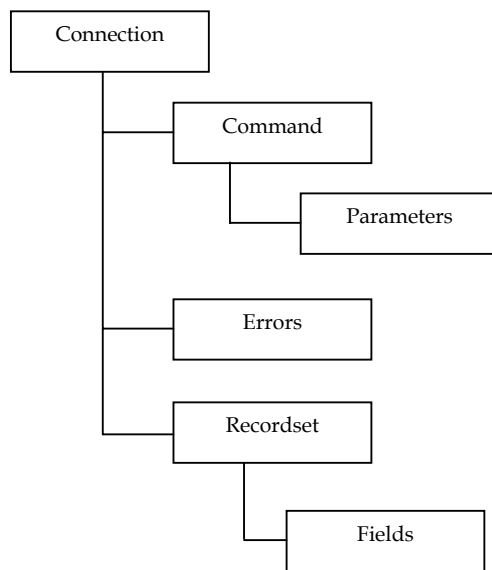
ADO is a collection of objects that enable ASP developers to connect to databases. ADO is part of OLE DB, an exciting new way to access and manipulate data that Microsoft has Developed.

## 15.2 Cookies

Cookies are information about client stored in the client's machine. Cookies are not part of the WAP 1.1 specification, and it is currently very rare for a WAP device to support them. However, many gateways can store cookies on behalf of devices.

## 15.3 ADO Object Model

The ADO object model provides developers a quick, powerful, method of accessing a datasource.



### The Connection Object

The top-level object is the connection object, which contains Command and Recordset objects and the Errors collections. It is used to create a connection to a data provider.

#### Syntax

```

Dim objc
Set connobj = Server.Createobject ("Adodb.Connection")
Connobj.open "connection String"
  
```

### Connection String

A Connection String is a simple character string that lists all of the information needed to connect to a source of data a typical connection string will contain sum or all of the following key process of information.

- Provider - the type of OLE-DB provider used in the Connection
- Driver - the type of the ODBC driver, such as the ODBC driver

for MSAccess or ODBC driver for MS SQL Server.

- Initial File name or Data Source
  - The physical database path and the file name.
  
- Initial Catalog
  - The name of the Database
  
- User Id
  - The user name needed to connect to the database (sa is the default for the administrator user name)
  
- Password
  - The password user specified in the user id field above.

### **The Command Object**

The command object has a slightly more specialized use than the connection and Recordset objects. Its best use is sending parameters to stored procedures and queries in your database.

### **Properties**

#### ActiveConnection

Used to set the Which connection is the Command object is part of. The string that this property is set to can be either a connection object, or the definition of the a connection.

#### Syntax

```
Set dataconn = server.Createobject ("Adodb.Connection")
Set datacmd = Server.createobject ("Adodb.command")
Datacmd.Activeconnection = dataconn
```

#### CommandText

It is a string value that represents the command you want to execute against the provider.

#### Syntax

```
Set datacmd = server.createobject ("Adodb.command")
Datacmd.activeconnection = dataconn
Datacmd.commandtext = "Select Query"
```

#### CommandType

This property tells the provider what type of command is being set. It may be either a stored procedures, SQL statement or a table name. This allows for a faster execution time when the provider executes the command.

#### Syntax

```
Datacmd.commandtext = adcmdtext
```

Adcmdtext has values of

Adcmdtext - This tells the provider that the commandText will be a SQL String.



- Adcmdtable - This tells the provider that the commandtext is a table.
- AdcmdStoredProc - this value enables the parameter to prepare for a stored procedure.
- AdcmdUnknown - This says that the CommandText is Unknown.

### Commandtimeout

This property specifies how long to wait before timing out a command. The default value of this property is 15 Secs.

#### Syntax

```
Set datacmd.Commandtimeout = 30
```

#### Recordset Object

Recordset object used to retrieve the data from the Database.

#### Syntax

```
Set rsobj = Server.createobject ("Adodb.Connection") Rsobj.open "tablename",  
conn, cursortype, locktype
```

#### Properties of the Recordset Object

- **BOF/EOF**

BOF stands for Beginning of File. EOF stands for End of file. This property is used to find whether the current position is before the first record or after the last record. If the property returns a value of True, then it is before or after the recordset.

- **RecordCount**

This property returns the number of records in the recordset.

- **Absoluteposition**

The AbsolutePosition property enables the developer to exactly which record to set the current pointer to. If the current record is the first in the recordset, it is equal to 1.

### Program

```
<%@ Language = "vbscript" %>  
<%  
Dim conn, rs  
Set conn = server.createobject ("Adodb.connection")  
Conn.open "amru","sa",""  
Set rs = Server.createobject ("Adodb.Recordset")  
Rs.open "select * from employee",conn,2,3  
Response.Write Rs.BOF  
Rs.movenext  
Response.write Rs.EOF  
Response.write Rs.absoluteposition  
Response.write Rs.Recordcount  
%>
```

### Methods

- ◆ AddNew - Creates a new Record
- ◆ Update - Saves the changes made to the current record.
- ◆ Open - Opens a Recordset
- ◆ Close - Closes the Recordset
- ◆ Delete - Deletes a Current Record
- ◆ Movefirst - Moves the cursor pointer to the First Record
- ◆ MoveLast - Moves the cursor pointer to the Last Record
- ◆ Movenext - Moves the cursor pointer to the Next record forward to the current position.
- ◆ Moveprevious- Moves the cursor pointer to the Next record backward to the current position.
- ◆ Move - Moves to a specific record.

### Cursor Types

Every recordset has only one cursor, which at any given time points to exactly one of the records in the recordset. When we open the recordset, we can specify the cursor type – and this will effect available functionality of the recordset that is returned to us. There are four type of Cursors

- **Forward-Only (adOpenForwardOnly) - 0**

This is the default type and gives you a non-scrollable recordset. This is often perfect in ASP code, because we can often find ourselves just stepping through a list of records in order to display each of them on the browser. In this case, we just need to start at the first record and move forward through the recordset until we get the end. We don't need to move backward. Its also static, so changes to the underline data are not represented.

- **Static - (adOpenStatic) - 1**

This is similar to a forward-only recordset, except that it is scrollable, so you can move back to previous records as well as moving forward.

- **Dynamic (adOpenDynamic) - 2**

The recordset is fully dynamic, and lets you see addition, amendments and deletion that are made by other users. It's fully scrollable so you can move around the recordset and way you like.

- **Keyset (adOpenKeyset) - 3**

This is similar to the dynamic recordset, but you can't see records that other users add, although you can see the changes to existing records.

### Lock Type

The locking typing is the closely related to whether or not the recordset is updatable. There are four types of Lock types.

- ◆ ReadOnly
- ◆ Pessimistic
- ◆ Optimistic

- ◆ Optimistic Batch
- ◆ ReadOnly - (adLockReadOnly)

This gives you a non-updatable Recordset. No locking is performed since you can't change the data in a Read-Only Recordset. This is a default.

- **Pessimistic**

This gives you an updatable recordset, in which the lock type is very protective. In this case, the copy of the record that exists in the data stores is locked as soon as you start editing it. This means that no one else can change the record until you release the lock, which is after you finished editing the recordset and have committed the update.

- **Optimistic**

This also gives you an updatable recordset, but the lock type is a little more carefree. In this case, the copy of the records in the data store remains unlocked while you are editing your changes within the Recordset. The data store records are only locked when you update your changes. So, if you choose this setting you are assuming that no one else will edit the record while you are editing it.

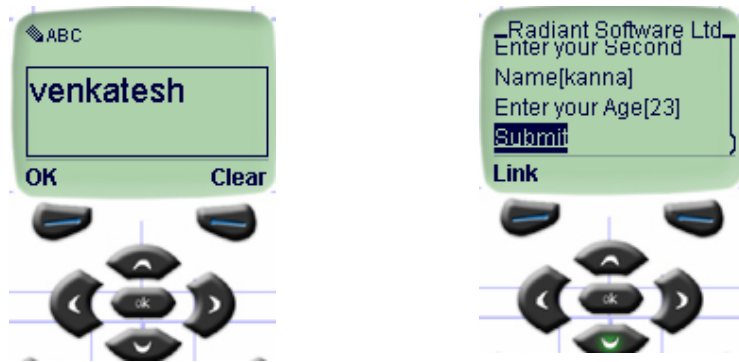
WAP with ASP programs

Receiving the values from the WAP client and Stored in the Database

```

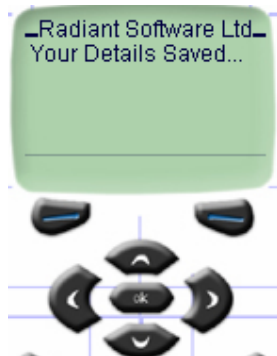
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
<card id="first" title="Radiant Software Ltd">
  <p>
    Enter your First Name
    <input type="text" name="fn" />
    Enter your Second Name
    <input type="text" name="sn" />
    Enter your Age
    <input type="text" name="age" />
    <anchor>Submit
  <go href="http://www.radisoft.com/yrdetail.asp" method="post">
    <postfield name="fname" value="\$(fn)" />
    <postfield name="sname" value="\$(sn)" />
    <postfield name="age" value="\$(age)" />
  </go>
  </anchor>
</p>
</card>
</wml>

```



```
fvdetail.asp
fvdetail.asp
```

```
<% Response.contentType="text/vnd.wap.wml" %>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
<card id="sec" title="Radiant Software Ltd">
  <p>
    <%
      a = Request.form ("fname")
      b = Request.form ("sname")
      c = Request.form ("age")
      dim conn,rs
      set conn = server.createObject ("Adodb.connection")
      conn.open "detaildsn","uid","pwd"
      set rs = server.createObject ("Adodb.recordset")
      rs.open "yrdetailtable",conn,2,3
      rs.addnew
      rs(0) = a
      rs(1) = b
      rs(2) = c
      rs.update
      Response.write "Your Details Saved..."
      rs.close
      conn.close
    %>
  </p>
</card></wml>
```



**Example 2:**

Retrieving from the database and send to the client.

```
<% Response.ContentType="text/vnd.wap.wml" %>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
<card id="first" title="Radiant Software Ltd">
  <p>
    The Radiant Branches...<br/>
    <% dim conn,rs
      set conn = server.createObject ("Adodb.connection")
      conn.open "detaildsn","uid","pwd"
      set rs = server.createObject ("Adodb.recordset")
      rs.open "branches",conn,2,3
      do while not (rs.eof)
        Response.write rs("brname") & "<br/>"
      Loop
      Rs.close
      Conn.close
    %>
  </p>
</card>
```

**15.4 Short Summary**

- ADO is part of OLE DB, an exciting new way to access and manipulate data that Microsoft has Developed.
- The top-level object is the connection object, which contains Command and Recordset objects and the Errors collections. It is used to create a connection to a data provider.
- Every recordset has only one cursor, which at any given time points to exactly one of the records in the recordset. When we open the recordset, we can specify the cursor type – and this will effect available functionality of the recordset that is returned to us.

**15.5 Brain Storm**

1. Define Cookies?
2. What is an ADO?
3. Discuss briefly about ADO Object?

## Lecture 16

---

# Database Handling in WAP

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✧ Receiving Values from the WAP client and stored in a Database
- ✧ Sending values from the Database to the WAP client.

---

## Coverage Plan

---

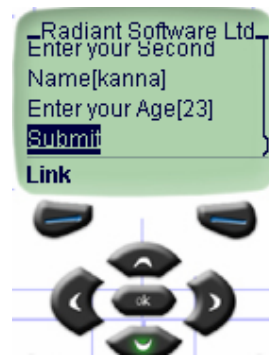
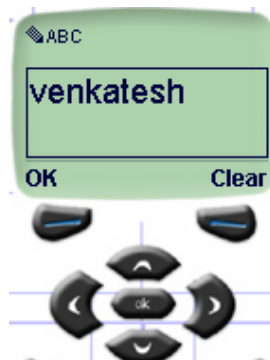
<b>Lecture 16</b>
16.1 Snap Shot
16.2 Receiving Values From The WAP Client And Stored In A Database
16.3 Retrieving From The Database and send to client.
16.4 Brain Storm

## 16.1 Snap Shot

In this lecture you are going to learn about how to receive values from the client and to store the data and also about how to retrieve the saved data.

## 16.2 Receiving the values from the WAP client and Stored in the Database

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
<card id="first" title="Radiant Software Ltd">
  <p>
    Enter your First Name
    <input type="text" name="fn" />
    Enter your Second Name
    <input type="text" name="sn" />
    Enter your Age
    <input type="text" name="age" />
    <anchor>Submit
  <go href="http://www.radisoft.com/yrdetail.asp" method="post">
    <postfield name="fname" value="\$(fn)" />
    <postfield name="sname" value="\$(sn)" />
    <postfield name="age" value="\$(age)" />
  </go>
</anchor>
</p>
</card>
</wml>
```



yrdetail.asp

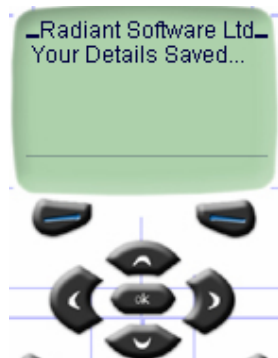
```
<% Response.contentType="text/vnd.wap.wml" %>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
<card id="sec" title="Radiant Software Ltd">
```



```

<p>
  <%
    a = Request.form ("fname")
    b = Request.form ("sname")
    c = Request.form ("age")
    dim conn,rs
    set conn = server.createobject ("Adodb.connection")
    conn.open "detaildsn","uid","pwd"
    set rs = server.createobject ("Adodb.recordset")
    rs.open "yrdetailtable",conn,2,3
    rs.addnew
    rs(0) = a
    rs(1) = b
    rs(2) = c
    rs.update
    Response.write "Your Details Saved..."
    rs.close
    conn.close
  %>
</p>
</card></wml>

```



### 16.3 Retrieving from the database and send to the client

```

<% Response.contentType="text/vnd.wap.wml" %>

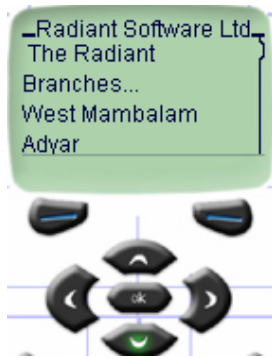
<! DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml" >
<wml>
<card id="first" title="Radiant Software Ltd">
  <p>
    The Radiant Branches...<br/>

  <% dim conn,rs
    set conn = server.createobject ("Adodb.connection")

```

```
conn.open "detaildsn","uid","pwd"
set rs = server.createobject ("Adodb.recordset")
rs.open "branches",conn,2,3
do while not (rs.eof)
    Response.write rs("brname") & "<br/>"
Loop
Rs.close
Conn.close

%>
```



#### 16.4 Brain Storm

1. How do you Retrieve the data from database and send it to the Client?

END

## Lecture 17

---

# Using Servlets in WAP

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ Power of Servlets
- ✔ Life Cycle of Servlets
- ✔ Request and Response Headers

---

## Coverage Plan

---

### **Lecture 17**

- 17.1 Snap Shot
- 17.2 Introduction to Servlets
- 17.3 Power of Servlets
- 17.4 Servlet API
- 17.5 Basic Servlets Structure
- 17.6 Life Cycle Of Servlets
- 17.7 Request And Response Headers
- 17.8 Servlet And Cookies
- 17.9 Short Summary
- 17.10 Brain Storm

## 17.1 Snap Shot

WAP applications, which will be hosted on normal web servers, can be written in WML and WMLScript. But these WAP applications can also be written using existing web technologies. Dynamic WML documents can be generated by CGI scripts, servlets, Java Server Pages, Active Server Pages, Perl, TCI, and so forth. In other words, WAP service can be implemented in Java, with the help of the WAP technologies described earlier. In this chapter, we are going to see how the WAP applications can be easily developed in Java using servlets and JSP.

## 17.2 Introduction to Servlets

Servlets are programs that run on a web server and build web pages. Building web pages on the fly is useful (and commonly done) for a number of reasons:

- The web pages are based on data submitted by the user. For example, the result pages from search engines are generated this way, and programs that process orders for e-commerce sites do this as well.
- The data changes frequently. For example, a weather-report or news headlines page might build the page dynamically, perhaps returning a previously built page if it is still up to date.
- The web page uses information from corporate databases or other such sources. For example, a web page can be developed at an on-line store that lists current prices and number of items in stock.

## 17.3 The Power of Servlets

### PORTABILITY

Because servlets are written in Java and conform to a well-defined and widely accepted API, they are highly portable across operating systems and across server implementations.

### Harness the Power of Java

Servlets can harness the full power of the core Java APIs: networking and URL access, multithreading, image manipulation, data compression, database connectivity, internationalization, remote method invocation (RMI), CORBA connectivity and object serialization, among others.

### Efficiency and Endurance

Servlet invocation is highly efficient. Once a servlet is loaded, it generally remains in the server's memory as a single object instance. Thereafter, the server invokes the servlet to handle a request using a simple, lightweight method invocation. Multiple, concurrent requests are handled by separate threads, so servlets are highly scalable.

### Safety

Servlets support safe programming practices on a number of levels. Because they are written in Java, servlets inherit the strong type safety of the Java language. In addition, the servlet API is implemented to be type-safe. Java's automatic garbage collection and lack of pointers mean that servlets are generally safe from memory management problems like dangling pointers, invalid pointer references, and memory leaks.

### Elegance

The elegance of a servlet code is striking. A Servlet code is clean, object oriented modular, and amazingly simple. One reason for this simplicity is the servlet API itself, which includes methods and classes to handle many of the routine chores of servlet development. Even advance operations, like cookie handling and session tracking, are abstracted into convenient classes.

### Integration

Servlets are tightly integrated with the server. This integration allows a servlet to cooperate with the server. For example, a servlet can use the server to translate files paths, perform logging, check authorization, perform MIME type mapping, and, in some cases, add users to the server's user database.

### Extensibility

The servlet API is designed to be easily extensible. As it stands today, the API includes classes that are optimized for HTTP servlets. But at a later date, it could be extended and optimized for another type of servlets.

## 17.4 The Servlet API

Servlets use classes and interfaces from two packages: **javax.servlet** and **javax.servlet.http**. The javax.servlet package contains classes that support, generic protocol independent servlets. These classes are extended by javax.servlet.http package to add HTTP specific functionality. Every servlet must implement **javax.servlet.Servlet** interface. Most servlets implement it by extending one of two special classes: **javax.servlet.GenericServlet** or **javax.servlet.HttpServlet**.

Unlike a regular java program, and just like an applet, a servlet does not have a main method. Instead, certain methods of a servlet are invoked by the servlet in the process of handling requests. Each time the servlet dispatches a request to a server, it invokes the servlet's service() method.

A generic servlet should override its service () method to handle requests as appropriate for the servlet. The service () method accepts two parameters: a request object and a response object. The request object tells the servlet about the request, while the response object is used to return a response. The following figure shows how a generic servlet handles requests.

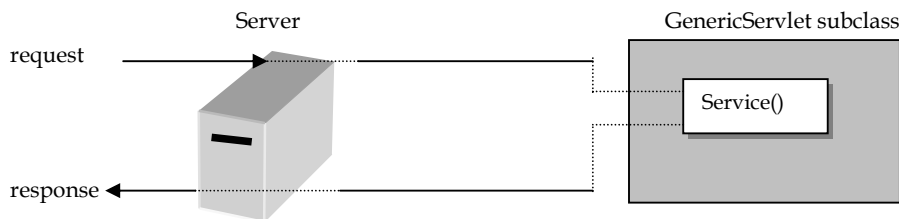


Fig: A generic servlet handling a request

In contrast, an HTTP servlet usually does not override the service () method. Instead, it overrides doGet() to handle GET requests and doPost() to handle POST requests. An HTTP servlet can override either or both of these methods, depending on the type of requests it needs to handle. The service () method of HttpServlet handles the setup and dispatching to all the doXXX() methods, which is why it usually should not be usually overridden. Figure shows how an HTTP servlet handles GET and POST requests.

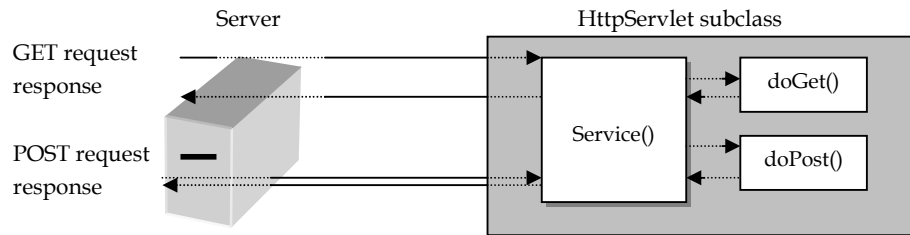


Fig An HTTP servlet handling GET and POST requests

## 17.5 Basic Servlet Structure

Here's the outline of a basic servlet that handles a GET request. GET request, for those unfamiliar with HTTP, is a request made by browsers when the user types in a URL on the address line, follows a link from a web page, or makes an HTML form that does not specify a METHOD.

Program

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class someservlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

// Use "request" to read incoming HTTP headers (e.g. cookies )
// and HTML form data (e.g. data the user entered and submitted)

// Use "response" to specify the HTTP response line and headers
// (e.g. specifying the content type, setting cookies).

PrintWriter out = response.getWriter ();

// Use "out" to send content to microbrowser
    }
}
  
```

To be a Servlet, a class should extend `HttpServlet` and override `doGet` or `doPost` (or both), depending on whether the data is being sent by a GET request a POST request. These methods take two arguments: an `HttpServletRequest` and an `HttpServletResponse`. The `HttpServletRequest` has methods to find out about incoming information such as form data, HTTP request headers etc. The `HttpServletResponse` has methods to specify the HTTP response line (200, 404, etc), response headers (Content-Type, Set-Cooke, etc.). Most importantly, it has methods to obtain the `PrintWriter` that is used to send output back to the client. For simple Servlets, most of the effort is spent in `println` statements that generate the desired page. Note that `doGet` and `doPost` throw two exceptions, so these have to be included in the declaration. Also note that classes in `java.io` (for `PrintWriter`, etc.) `javax.servlet` (for `HttpServlet`, etc.), and `javax.servlet.http` (for `HttpServletRequest` and `HttpServletResponse`) are to be imported. Finally, note that `doGet` and `doPost` are called by the service method, and sometimes you may want to override service directly, e.g. for a Servlet that handles both GET and POST request.

## 17.6 The Life Cycle of a Servlet

Since servlets come in the form of Java objects, there are many different variations on not only how they are loaded but also how they are unloaded again, if at all. When the server decides to load particular servlet, it uses the standard Java class loading mechanisms to create the class instance. Using this technique, servlets can be loaded from anywhere on the network and if the server is connected to the Internet, from anywhere in the world.

```
Class c = class . forName ( "http:// <server>/testclass");
```

Once this method has returned, the class can be accessed as normal. Assuming the URL doesn't upset the security of the system, classes can be located anywhere on the network.

### Note :

- Loading classes from within a Java servlet, applet or any Java program is achieved through the `java.lang.class`. This class method returns the class object associated with the full URL that was passed in.

Each servlet has the same life cycle:

- A server loads and initializes the servlet.
- The servlet handles zero or more client requests.
- The server removes the servlet. (some servers do this step only when they shut down)

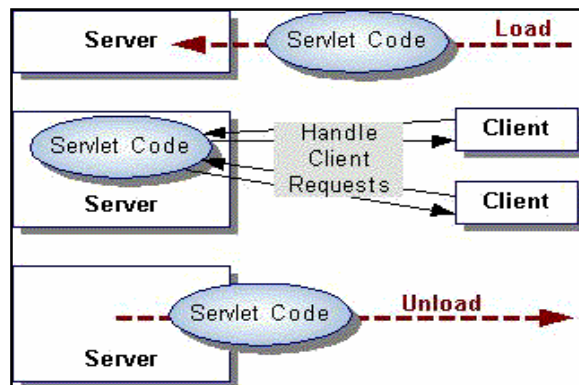


Fig Life cycle of a servlet

### Initializing a Servlet

When a server loads a servlet, the server runs the servlet's `init` method. Initialization completes before client requests are handled and before the servlet is destroyed.

The server calls the `init` method once, when the server loads the servlet, and will not call the `init` method again unless the server reloads the servlet. The server can not reload a servlet until after the server has destroyed the servlet by running the `destroy` method.



### Interacting with Clients

After initialization, the servlet is able to handle client requests.

### Destroying a Servlet

Servlets run until the server destroys them, for example, at the request of a system administrator. When a server destroys a servlet, the server runs the servlet's destroy method. The method is run once; the server will not run the destroy method again until after the server reloads and reinitializes the servlet.

### Starting with Servlets

Dynamic WML documents for wireless devices can be easily developed using Java servlets. Once you know the WML syntax, building WAP applications using Java servlets can be an easy task.

Following is an example of a simple servlet that displays the current date and time on a wireless device when invoked.

Program

To show the use of servlets in developing WAP applications.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
/**
 * This is a simple servlet that will run on a cell-phone. It displays the
 * current date and time.
 */
public class MobileDate extends HttpServlet {
    public void service (HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        // set content type for wireless data
        response.setContentType("text/vnd.wap.wml");
        // get the communication channel with the requesting client
        PrintWriter out = response.getWriter();
        // write the data
        out.println("<?xml version=\"1.0\"?>");
        out.println("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"");
        out.println(" \<a href='\"http://www.wapforum.org/DTD/wml_1.1.xml\">");
        out.println("<wml>");
        out.println("<card title=\"MobileDate\">");
        out.println(" <p align=\"center\">");
        out.println("Date and Time Service<br/>");
        out.println("Date is: " + new java.util.Date());
        out.println("</p>");
        out.println("</card>");
        out.println("</wml>");
    }
}
```

Most of the preceding code uses pure WML tags, with the exception of the line:

```
Response.setContentType("text/vnd.wap.wml");
```

This line ensures that the correct MIME type is set for the WML document. The rest of the above listing basically outputs a WML document.

#### STEP TO RUN THE ABOVE EXAMPLE

- ◆ Compile the above program (For successful compilation you may need to set the classpath to *root\_directory/javawebserver2.0/lib/servlet.jar*).
- ◆ Copy the .class to *root\_directory/javawebserver2.0/servlets* directory
- ◆ Run the Java web server. Type *httpd* (go to *javawebserver2.0/bin* directory) to start the service.
- ◆ Type **http://localhost:8080/servlet/MobileDate** in the load location option of the Browser menu.
- ◆ After loading is over, you can see the result in the Simulator.

When this servlet is invoked from a mobile phone, it displays the current date and time as shown below:

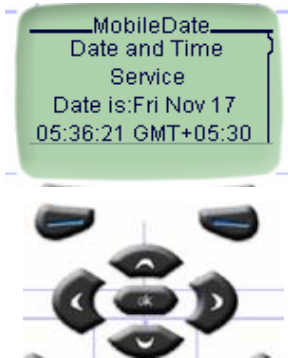


Fig A servlet running on the Nokia Toolkit

### 17.7 Request and Response Headers

#### RequestHeader

When a HTTP client sends a request, it is required to supply a request line (usually GET or POST). If it wants to, it can also send a number of headers, all of which are optional except for Content-Length, which is required only for POST requests. Here are the most common headers:

- ◆ Accept MIME types that the browser prefers.
- ◆ Accept-Charset.
- ◆ Accept-Language.
- ◆ Authorization information.
- ◆ Content-Length (for POST requests to show how much data is attached)
- ◆ Cookie (if the browser supports).
- ◆ Host (host and port as listed in the original URL)
- ◆ User-Agent (type of the browser)

Reading headers is very straightforward. The `getHeader` method of the `HttpServletRequest` can be called. This would return a string if the header is supplied with the request, or null otherwise.

### EXAMPLE

To show the printing of all headers

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

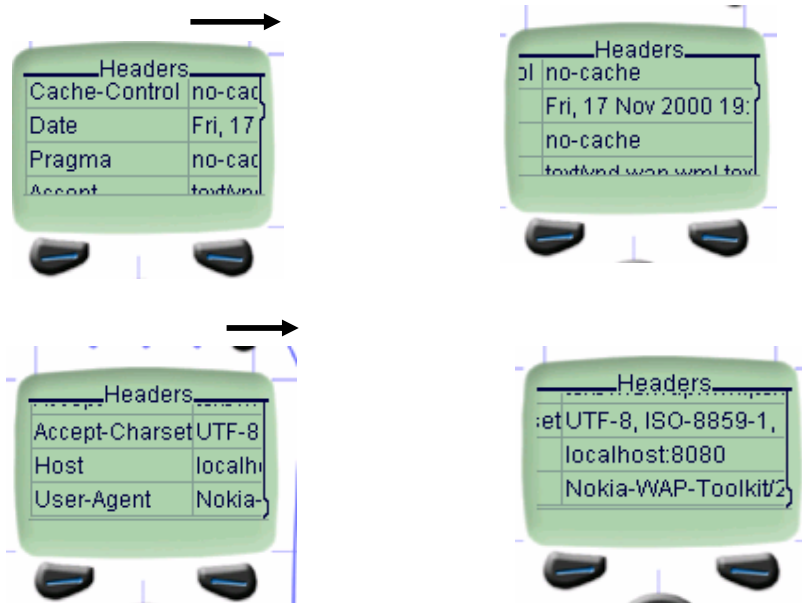
public class ShowHeaders extends HttpServlet {
    public void service (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // set content type for wireless data
        response.setContentType("text/vnd.wap.wml");

        // get the communication channel with the requesting client
        PrintWriter out = response.getWriter();

        // write the data
        out.println("<?xml version=\"1.0\"?>");
        out.println("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"");
        out.println(" \"http://www.wapforum.org/DTD/wml_1.1.xml\">");
        out.println("<wml>");
        out.println("<card title=\"Headers\">");
        out.println("  <p  mode=\"nowrap\">");
        out.println("    <table columns=\"2\">");
        Enumeration headerNames = request.getHeaderNames();
        while(headerNames.hasMoreElements())
        {
            String headerName = (String)headerNames.nextElement();
            out.println("      <tr><td>"+headerName+"</td>");
            out.println("        <td>"+request.getHeader(headerName)+"</td>");
            out.println("      </tr>");
        }
        out.println("    </table>"+<br>");

        out.println("  </p>");
        out.println("</card>");
        out.println("</wml>");
    }
}
```

The output of the following program might look like this:



A response from a Web server normally consists of a status line, one or more response headers, a blank line, and the document. Setting the HTTP response headers often goes hand in hand with setting the status codes in the status line. Response headers can be used to supply the modification date (for caching), to instruct the browser to reload the page after a designated interval, to specify how long the file is, and many other tasks. Here are the most common response headers:

- ◆ Allow (Which request methods (GET, POST, etc) are supported by the server support?)
- ◆ Content-Length (How many bytes are being sent?)
- ◆ Content-Type (What is the MIME type of the document?)
- ◆ Date (Current time in GMT)
- ◆ Last-Modified (When was the document last changed?)
- ◆ Expires (At what time should content be considered out of date and thus no longer cached?)
- ◆ Server (Type of server I?)

**HttpServletResponse** class supplies a number of convenient methods for specifying common headers. For example, `setContentLength` method sets the Content-Type header, `setContentLength` method sets the Content-Length header and so on.

What servlets can do with Request and Response headers?

A servlet's main purpose in life is to receive a request and send back a response. In order to process requests and responses, the servlet needs to be able to:

- ◆ Get request parameters out of a request
- ◆ Get environmental variable information out of a request
- ◆ Store Java objects in a request before forwarding it to another 'computational resource'
- ◆ Obtain a stream to write a response to
- ◆ Serve multiple requests and responses within a single session

## 17.8 Servlets and Cookies

By the way, cookies are useful for maintaining the state and keeping track of users' sessions. Cookies are part of the WAP specification, but unfortunately, are not yet implemented by all WAP browsers. The Nokia 7110 device does not yet support cookies. However, Phone.com's UP.Simulator does support them.

### Session Tracking

A session is a series of request-response exchanges with the same client. Since HTTP is a stateless protocol, identifying the client as 'the same' is not a trivial task. Over the many years of CGI programming, several approaches to this task have been developed, using it is not cookies, hidden form fields and so on. When working with servlets, necessary to be aware of these approaches because the Servlet API has higher-level session tracking facilities that hides these details from the programmer.

Session management depends on a 'session ID' usually a number. When the server create and stores a session, it sends the session ID to the client (browser) which echoes that ID so that the server knows which session to retrieve. The session ID can be made part of the URL itself (this is called 'URL rewriting'). But is more usually, it is stored as a 'cookie'; a small file on the client represented by an object of class **javax.servlet.http.Cookie**. However, not all browsers will accept cookies. In such case, the `HttpServletResponse.encodeURL(String url)` method can be used to generate the 'ACTION=target which is to be rewritten.

### Session Tracking API

Methods for session tracking are declared in the **HttpSession** interface. An object implementation of that interface is associated with every visitor to the site. Arbitrary name-value pairs can be stored, retrieved are deleted from this object. The association between a visitor and a session is established by giving each visiting client a unique session ID, typically a very long string created and maintained by the server. Every time a new request comes in, the client is checked to see whether it has a 'valid' ID. If the answer is no, the session is considered to be 'new'. If the answer is yes, then the client is in the middle of an ongoing session.

To obtain an existing or a new HttpSessionobject use `getSession ()` method:

```
public HttpSession getSession ();
Or
public HttpSessionon getSession(boolean create);
```

If the boolean create is true and there is no current session, then a new HttpSession object is created and given an ID. If the Boolean is false then a new session object is not returned, only an existing one if it does indeed exist. The default value of the boolean is true.

Once you have a session object you can inquire whether or not its ID is saved using a cookie or URL rewriting as follows:

```
public boolean isRequestedSessionIdfromCookie();
```

Or

```
public boolean isRequestedSessiononIdFromURL();
```

More importantly, you can ask for the Id itself, and whether or not it is valid using:

```
public string getRequestedSession Id();
```

Or

In summary, the session object goes through these stages: new, valid, invalid. Once it is invalid, it is removed from memory, and its session ID on the client becomes invalid. All this is taken care of by the servlet engine; the serlet programmer can sit back and relax.

The actual methods of HttpSession fall into two groups. One is associated with value: getting, putting and removing them. The other is association with the various properties of a session, such as it newness and validity.

First off, there is:

```
public boolean isNew();
```

A session is considered new if it has been created by the server and not received from the client as part of the current request. In a situation where you require the user to go through a login procedure, this method should be used to check whether the client has indeed logged in or has arrived at your page in some illegitimate way.

If the session is not new but is not valid either, then trying to do anything with it, including asking whether or not it is new, will result in an `IllegalStateException`. If the session is valid, then you can get its ID and ask it various questions about its age and what it's been doing lately using the following.

```
public string getId( );
```

```
public long getCreationTime( );
```

```
public long getLastAccessedTime( );
```

```
public int getMaxInactiveInterval ( );
```

A session with an ID remains valid until it is invalidated, either by an explicit call on HttpSession's `invalidate ( )` or if the session remains inactive for a specified period of time. There is usually a default value of about 20-30 minutes for that period. The maximum period of inactivity can be found by `getMaxInactiveInterval ( )`, and it can be changed by `setMaxInactiveInterval( )`.

Any number of name-value pairs can be associated with a session object, depending on what is to be done. The same is a string and the value is an object. you manipulate them using the following method:

```
public object gatvalue ( String name);
```

```
public String [] getValueNames ( );
```

```
public void putValue (string name, object value);
```

```
public void removeValue (String name);
```

Where can a Session Object be used?

A common example of using a session object is when your users do multiple database access and you want the database connection (or a hook into a connection pool containing it) to last for the duration of the session. In this case, you store the connection object or the hook in the session object and pass the Session object to the code that handles database queries. A more general example is security information, such as the username and password to be associated with database activity.

### 17.9 Short Summary

- K Once a Servlet is loaded, it generally remains in the server's memory as a single object instance.
- K Generic Servlet should override its `service()` method. In contrast, HTTP servlet does not override its `service()` method.
- K A session is a series of request-response exchange with the same client.

### 17.10 Brain Storm

1. Discuss about the powers of servlet?
2. Explain the life cycle of Servlet?
3. What is Request and Response headers? How they are implemented through WAP?
4. Explain briefly about session tracking and session Tracking API?



## Lecture 18

---

# WAP and JSP

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ JSP Architecture
- ✔ Using JDBC in JSP
- ✔ JSP Error Page



---

## Coverage Plan

---

<b>Lecture 18</b>
-------------------

- |   |
|---|
| <ul style="list-style-type: none"><li>18.1 Snap Shot</li><li>18.2 JSP Request Model</li><li>18.3 JSP Architecture</li><li>18.4 Using JDBC In JSP</li><li>18.5 JSP Error Page</li><li>18.6 Application Using JSP</li><li>18.7 Short Summary</li><li>18.8 Brain Storm</li></ul> |
|---|

## 18.1 Snap Shot

JSP provides a number of server-side tags that allow developers to perform most dynamic content operations without ever writing a single line of Java code. So developers who are only familiar with scripting, or even those who are simply HTML designers, can use JSP tags for generating simple output without having to learn Java. Advanced scriptures or Java developers can also use the tags, or they can use the full Java language if they want to perform advanced operations in JSP pages.

## 18.2 JSP Request Model

Now let's take a look at how HTTP requests are processed under the JSP model. In the basic request model, a request is sent directly to a JSP page. Figure illustrates the flow of information in this model. JSP code controls interactions with JavaBeans components for business and data logic processing, and then displays the results in dynamically generated HTML mixed with static HTML code.

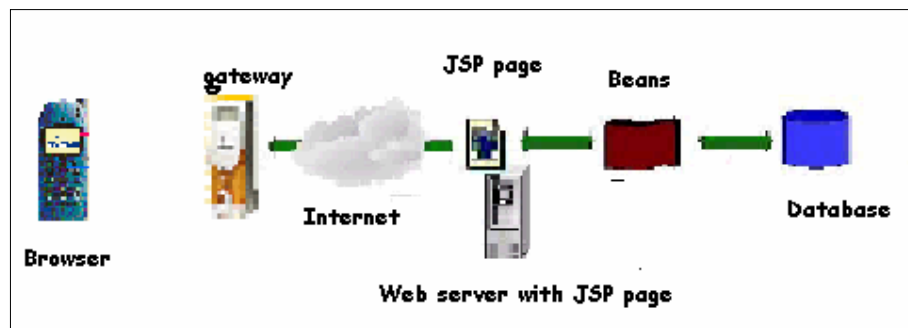


Figure . Basic JSP request model

The beans depicted can be JavaBeans or EJB components. Other, more complicated request models include calling out to other JSP pages or Java Servlets from the requested JSP page.

## 18.3 JSP Architecture

The source code of a JSP page is essentially just HTML (or text—or even XML) sprinkled here and there with either special JSP tags and/or Java code enclosed in these tags. The file's extension is .jsp rather than the usual .html or .htm, and it tells the server that this document requires special handling.

The special handling, accomplished with a Web server extension or plug-in, involves four steps.

1. The JSP engine parses the page and creates a Java source file.
2. It then compiles the file produced in Step 1 into a Java class file. The class file created in Step 2 is a servlet, and from this point on, the servlet engine handles the class file in the same manner as all other servlets.
3. The servlet engine loads the servlet class for execution.
4. The servlet executes and streams back the results to the requestor.

Although this process might seem time consuming and expensive, it's much more efficient than it sounds. Steps 1 and 2 occur only once, when the JSP first deployed or updated.

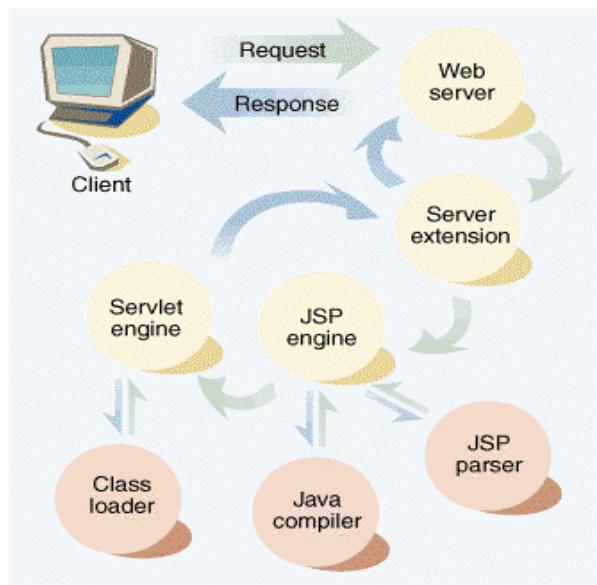


Figure: JSP Architecture

(By the way, most engines allow you to update a page on the fly.) The servlet engine performs Step 3 only upon the first request of that servlet since the last server restart. After that, the class loader loads the class once and is available for the life of that JVM. Finally, some application servers provide page caching, which can further improve the performance and reduce the cost of executing the request. With page caching, even Step 4 may execute only once depending on how dynamic the page data is.

The following example is similar to the preceding servlet example; it displays the current date and time on a wireless device.

Program

JSP to display the current date and time

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

<%
response.setContentType("text/vnd.wap.wml");
out.println("<wml>");
out.println("<card title=\"MobileDate\">");
out.println(" <p align=\"center\">");
out.println("Date and Time Service<br/>");
out.println("Date is: " + new java.util.Date());
out.println("</p>");
out.println("</card>");
out.println("</wml>");
%>
```

Again, you need to set the MIME type has to be set correctly to make sure that the mobile phone browser is able to parse the contents. The following line sets the MIME type to a WAP/WML document:

```
<% response.setContentType("text/vnd.wap.wml") %>
```

#### Steps to run the example?

- ◆ Copy the .jsp to *root\_directory/javaawebserver2.0/public\_html* directory.
- ◆ Run the Java Web Server. Type *httpd* (go to *javaawebserver2.0/bin* directory) to start the web server.
- ◆ Type *http://localhost:8080/MobileDate* in the Load Location option of Browser menu.
- ◆ After Loading is over, you can see the result in the Simulator.

Once this JSP document is invoked, it displays the current date and time as shown in Figure.



## 18. 4 Using JDBC in Java Server Pages

The JDBC interface is a pure Java API that is used to execute SQL statements. The JDBC provides set of classes and interfaces that can be used by developers to write database applications. Basic JDBC interaction, in its simplest form can be broken into four steps:

- ◆ Open connection to the database.
- ◆ Execute a SQL statement.
- ◆ Process the results
- ◆ Close the connection to the database

Now we are going to create a JSP page which performs a query on a predefined table called MOVIE. The table has the following columns

Movie name, Rating and Theatre name in which the movie is running.

#### Program

To shows the use of JDBC in WAP application

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

<%
response.setContentType("text/vnd.wap.wml");
Connection con = null;

try {
```

```

//Load the driver class file
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

//Make a connection to the ODBC datasource- MovieCatalog
con = DriverManager.getConnection("jdbc.odbcmoviecatalog","","");

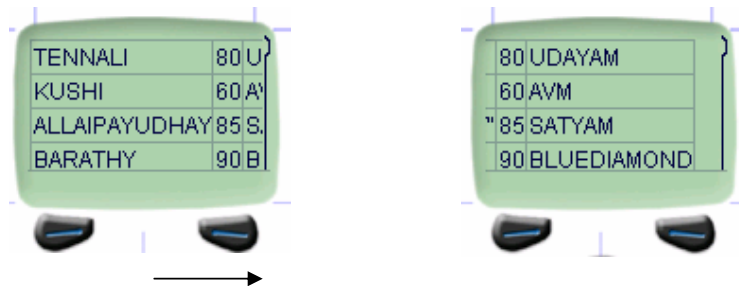
//create the statement
Statement statement = con.createStatement();

//Use the created statement to SELECT the data from the movie table.
ResultSet rs = statement.executeQuery("SELECT * FROM MOVIE");

//Iterate over the ResultSet
%>
<!--Add a table to format the results -- >
out.println("<wml>");
out.println("<card>");
out.println("<p mode='nowrap'>");
out.println("<table columns='3'>");
<%
while(rs.next()) {
//get the title name which is a String
out.println("<tr>\n<td>"+rs.getString("title_name")+"</td>");
//get the rating
out.println("<td>"+rs.getString("rating")+"</td>");
//get the theatre name
out.println("<td>"+rs.getString("th_name")+"</td>");
out.println("</tr>");
}
out.println("</p>");
out.println("</card>");
out.println("</wml>");
%>

```

The output of the following program might look like this:



## 18.5 JSP Error Page

A JSP error page is a page that is displayed whenever a request time error occurs. To create a JSP error page, you need to create a Java Server Page is to be created and you tell the JSP engine is to be told that the page is an error page. This can be done by setting its page attribute isErrorPage to true.

Creating an error page

**EXAMPLE**

To show how an error page is to be created.

```
<wml>
<card>
<%@ page isErrorpage="true">
<!-- -Use the implicit exception object which holds a reference - - >
<!-- - To the thrown exception -- >
<p>
Error:<%= exception.getMessage() %> has been generated.
</p>
</card>
</wml>
```

There are two lines of code that need to be looked at to understand just how easy it is to create a JSP page. The first is the page directive line, which indicates that this JSP is an error page. This code is

```
<%@ page isErrorPage="true" %>
```

The second line of code designates where the thrown exception is being used. This line is

```
Error:<%= exception.getMessage() %> has been generated.
```

## 18.6 Using the ErrorPage

To use the error page, it takes only one attribute, in the page directive, to make the JSP aware of an error page. The errorpage attribute and has to be simply added set its value equal to the location of the JSP error page.

The following listing uses the error page we created in the previous section.

**EXAMPLE**

To show the use of an error page

```
<%@ page errorpage = "errorpage.jsp" %>
<%
    if(condition)
    {
        // Just throw an exception
        throw new Exception(" JSP exception ");
    }
%>
```

### Application Using JSP

Before we start writing an application, we will have a look at what a Java bean is. This is essential because the following application just counts the number of requests coming for a JSP page using a JavaBean. A JavaBean is a 100% Java component that works on any Java Virtual Machine. There is nothing magical about creating a JavaBean. You create a Java Class that implements the java.io.Serializable interface and uses public get/set methods to expose its properties. Given below are the steps to create an application using JSP.

**Step 1: Creating a JavaBean**

The following listing is a JavaBean that acts as a counter. It has a single *int* property, *count*, that holds the current number of times the bean's property has been accessed. It also contains the appropriate methods for getting and setting this property.

**EXAMPLE**

To show how to create a JavaBean

```
Public class Counter {
// Initialise the bean on creation
int count=0;
// Parameterless constructor
public Counter() {
}
// Property Getter
public int getCount()
{
    // Increment the count property with every request
    count++;

    return this.count;
}

// Property Setter
public void setCount(int count) {

    this.count = count;
}
} //end of Counter
```

**Step 2: Integrating Bean into a JSP**

The following listing contains the JSP that will use the Counter bean.

**EXAMPLE**

To show how a Bean is integrated into a JSP

```
<wml>
<card title="JSP and Bean">
<!-- Set the scripting language to java - - >
<%@ page language="java" %>
<%@ page import ="Counter">
<!-- - Instantiate the Counter bean with an id of "counter" - - >
<jsp:useBean id="counter" scope="session" class="Counter" />
<!-- - Set the bean's count property to the value of request parameter "count"
- - >
<!-- - using jsp:setProperty action - - >
<jsp:setProperty name="counter" property="count" param="count">
<%
```

```
// write the current value of the property count
out.println("count is " +counter.getCount()+"<br>");
%
```

### 18.7 Short Summary

- K JSP code controls interaction with javaBeans component for business and data logic processing and then displays the result in dynamically generated HTML code mixed with static HTML code.
- K A JSP error page is a page that is displayed whenever a request time error occurs.

### 18.8 Brain Storm

1. Explain about JSP Request Model?
2. Discuss briefly about JSP Architecture?
3. How to use JDBC in JSP?
4. What is a JSP error page? How do you create an error page?

❧



## Lecture 19

---

# WAP Security

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✧ Security concepts
- ✧ Authentication
- ✧ Authorization

---

## Coverage Plan

---

### Lecture 19

- 19.1 Snap Shot
- 19.2 Security Concepts
- 19.3 Short Summary
- 19.4 Brain Storm

## 19.1 Snap Shot

Security has always been a hot topic. In the past the technologies involved were fairly simple, so the steps required to implement security were correspondingly easy to identify and implement. Commercial computing has raised the stakes, both in terms of the technology – and in terms of the damage that could be caused or power gained by breaches of security. Earlier security was only a preserve of large corporations, and very few people had to deal with its complexities. However, nowadays it is not only a concern for large corporations, but for everyone who has Internet access. Security has been, and remains, one of the biggest concerns for both users of the Internet and for the vendors and service providers who wish to use it as a channel to market. As WAP expands into new markets, the context within which security becomes a consideration also broadens.

In this chapter, we will be looking at some of the security issues that have an impact on the WAP applications and the environment in which they will be deployed and used. We will also look at how the security layer in the WAP protocol stack viz., Wireless Transport Layer Security (WTLS) operates.

## 19.2 Security – Basic Concepts

Let us investigate some of the basic concepts that the lie behind the art of security. In this section, we will be focusing on:

- ◆ Authentication
- ◆ Confidentiality
- ◆ Integrity
- ◆ Authorization
- ◆ Non-repudiation

### Authentication

Authentication is the process of ensuring that the other party is actually who they claim to be. Although this sounds trivial, it is actually quite complex. There are many different ways of authenticating parties to a transaction or interaction. The participants in the transaction need to agree on an authentication protocol that they are capable of using and that meets their requirements.

At times it becomes necessary to authenticate people whom we have never seen before. In this case a token can be used. A token has certain characteristics that make it acceptable as an authentication. A recognized Certification Authority (CA) issues an authentication token, for example the government. It usually contains information to prove the identity of the bearer. A passport is a typical example of an authentication token. Similarly, digital certificates and the signatures that they contain can be used to authenticate the participants in an electronic transaction.

The purpose of authentication is try to catch an activity called spoofing. This is occurs when one party tries to hide its true identity and poses as another, by carrying a false passport, for example. Spoofing need not occur only at the beginning of a transaction. It is possible for spoofers to interpose themselves after the initial authentication has already taken place, and then pretend to be either one or both the ends

of the transaction. Re-authenticating participants during the transaction can overcome this kind of problem.

### **Confidentiality**

Confidentiality is one of the most important aspects of security. It is very easy to intercept digital communication. It is virtually impossible to prevent it or even identify when it has happened. Therefore, it is often necessary to assume that communications will be intercepted and take steps to make sure that when the information is intercepted it cannot be understood or used. Information is usually encrypted to ensure confidentiality. Encryption is simply the process of encoding information into a different representation that cannot easily be understood. Effective encryption, however, is a very complex issue, which we will discuss in a bit more detail in the next section. If a message is encrypted, it is obviously necessary to have a method to decrypt the information so that the original message can be obtained.

### **Integrity**

If a message can be intercepted, it can also be interfered with along the way. This might involve either changing some of the contents of the message, or substituting the message with another. Either way, it has to be ensured that the message has not been tampered with during transmission. That is what integrity is all about. In the world of computing, message integrity is usually assured by deriving a hash value for the message and transmitting that value either along with the message or independently. A hash function is a mathematical algorithm that is used to calculate a number (the hash value). This value is derived from and dependent on the content of the message. If there is a change in the content, the hash value also changes. Hence, by comparing the original hash value of a message (sent separately to the message itself) against the hash value of the message as calculated at the receiving end, it can be verified whether or not the message has been tampered with. A hash value is also sometimes called a message digest because it is a 'digested' form of the message.

### **Authorization**

Authorization is the process that determines whether a particular party has the right to perform a particular action with respect to a particular object, in a particular situation.

Authorization would be meaningful only if there is an effective mechanism of authentication. Authentication is, therefore, a necessary prerequisite for authorization. Authorization is not directly dependent on – or related to – the channel by which communication takes place. Therefore, irrespective of whether the transaction takes place over the Internet, at the local branch, or using a mobile phone, the issues will be the same.

### **Non-repudiation**

Non-repudiation means implementing a mechanism such that it is impossible for the parties to a transaction to deny either that the transaction took place, or that they were party to it. Non-repudiation is also dependent on an effective means of authentication. Non-repudiation is an issue over both the Internet and the wireless communication channel. The need for effective non-repudiation mechanism has implications both for the protocols that are implemented and the types of transactions that can take place in the absence of these protocols.

### 19.3 Short Summary

- ◆ WTLS is the security layer of WAP. Its primary goal is to provide privacy, data integrity, and authentication for WAP applications. It provides the transport service interface for the upper level layers. The interface is similar to the transport service interface below the WTLS. The WTLS is based on the well-known TLS v1.0 security layer used in the Internet. The wireless networks require support for both datagram and connection-oriented transport layer protocols. The mobile equipment sets requirements for the algorithms because of the limited processing power and memory. In addition, the low bandwidth must be dealt with and the restrictions on exporting and using cryptography must be considered.
- ◆ The WTLS incorporates new features such as datagram support, optimized packet size and handshake, and dynamic key refreshing. It has been optimized for low-bandwidth bearer networks with relatively long latency. Fast algorithms are chosen into the algorithm suite. The mobile equipment like cellular phone can be constructed to support only a set of cipher suites.
- ◆ The objective of WTLS is to be a lightweight and efficient protocol with respect to bandwidth, memory and processing power.

### 19.4 Brain Storm

1. What is the security layer in WAP?
2. What is Authentication?
3. What is Privacy?
4. What is integrity?
5. Explain WTLS?
6. Explain SSL and TLS.

## Lecture 20

---

# Encryption Technologies

---

## Objectives

After completing this Lecture, you should be able to do the following

- ✔ Cryptography
- ✔ Cipher Suites Certificates
- ✔ WTLS

---

## Coverage Plan

---

<b>Lecture 20</b>
20.1 Snap Shot
20.2 Cryptography
20.3 Keys
20.4 Ciphers
20.5 Cipher Suites
20.6 Certificates
20.7 Security in WAP
20.8 Role of SSL and WTLS in WAP communication
20.9 Short Summary
20.10 Brain Storm

## 20.1 Snap Shot

Encryption and its related technologies are the cornerstones of secure electronic communications. Encryption is the process of encoding data such that only the intended recipients can understand it. This simple concept has been extended and enhanced over the years. Today's encryption and related techniques are used in a number of ways to protect data. In the next section we shall see, how the various encryption techniques that are currently available can be used to address several of the key security issues such as confidentiality, integrity, authentication, and - non- repudiation.

The encryption technologies are not unique to wireless communication alone and apply equally to communication over the Internet. Before we proceed further, there are two important terms viz. Plaintext and ciphertext that we should be familiar with to understand how encryption works.

- Plaintext is text or data that has not been encoded.
- Ciphertext is the result of applying a cryptographic algorithm to the Plaintext.

The encryption technologies that we shall look at include:

- Cryptography
- Keys
- Symmetric and Asymmetric Ciphers
- Cipher Suites
- Certificates

## 20.2 Cryptography

Cryptography is the art of keeping messages hidden or secure. This can be achieved by encoding or enciphering data using some reversible algorithm to hide its meaning. By reversible, we mean that the encoded messages can also be decoded to derive the plaintext from the ciphertext. The algorithm that is used is referred to as a cipher. Robust cryptographic algorithms are extremely difficult to invent. Hence, there are only a few algorithms in the world that have any level of credibility.

There is no point in using just an algorithm to encode data, because cracking an encrypted message would reduce to identifying the algorithm that is used to encode it. To overcome this problem, cryptographic algorithms use keys to control how the plaintext is converted to ciphertext. As a result is an encoded message cannot be deciphered without knowing both the key and the algorithm used for the encoding.

## 20.3 Keys

Keys are critical to cryptography. It is a value used in the algorithm to influence its operation so that it is capable of producing different ciphertexts for identical plaintext. Therefore, to decode a message it is necessary to know both the algorithm and the key. The length of the key used is an important factor, because the longer the key the more possible values there are that the key could take. But the size can also impede the performance. Moreover, the longer keys make the process of encryption and decryption even more expensive. The strength of an encryption algorithm does not merely depend on of the size of the key. Keys of a similar size may differ in terms of relative ease with which they can be cracked.

## 20.4 Ciphers



Ciphers can be classified according to the number of keys that they use, and how the keys are used to encrypt and decrypt data.

There are two types of ciphers:

- Symmetric ciphers
- Asymmetric ciphers.

Both these ciphers are used in securing electronic transactions, although in different ways. Symmetric ciphers use the same key for encryption and decryption, whereas asymmetric ciphers use a pair of keys – one to encrypt and one to decrypt.

### **Symmetric Ciphers**

Both the parties to a transaction need to have a copy of the key if the same key is used for encryption and decryption of data. This means that if any one else gets the key, then all the messages encrypted with that key can be decrypted. Hence, the security of the whole system depends on keeping the key secret.

Although, exchanging the key securely is a big problem, it does not mean that symmetric keys are not used. In fact, they are used by far in the majority of the encryption that takes place, including the encryption that is used on the Internet. The reason for this is because they are significantly faster than the alternative, asymmetric keys. Moreover, there is actually a way by which the key can be exchanged securely. We shall see how this can be achieved in the section on key exchange.

### **Asymmetric Ciphers**

Asymmetric ciphers use a matched pair of keys. One of the keys is a public key. This key need not be kept secret. The other is a private key, which does have to be kept secret. The public key is derived from the private key using a mathematical algorithm. Data that is encrypted using the public key can only be decrypted using the private key that the public key was derived from.

The problem associated with exchanging symmetric keys is eliminated by the use of public and private keys. But this entire system is dependent on the fact that both the parties should keep their respective private keys secret, otherwise it breaks. The disadvantage of asymmetric keys is that they take considerably more computing power than symmetric keys. So it is not really feasible to use them for encrypting each and every piece of data that is transmitted. To surmount this problem, they are often combined with symmetric keys in cipher suites to facilitate the exchange of keys that are then used with symmetric ciphers for encrypting data. This helps to reduce the amount of computation that is required.

## **20.5 Cipher Suites**

Cryptography comes pre-packaged in suites. Cryptographic suites are collections of algorithms. These suites individually or together, support one or more of the operations that are required for secure transmission of data, namely:

- Key Exchange
- Encryption
- Message digests

### **Key Exchange**

Key exchange algorithms allow secure exchange of symmetric keys. These algorithms use asymmetric keys to facilitate the exchange of a shared secret, which is used to derive a symmetric key. This symmetric key can then be used with a symmetric encryption algorithm to encrypt the remaining data in the communication. What is encrypted and exchanged is not actually the secret key itself, but rather a value that is used to derive the secret key.

The requirement to derive the secret key from the shared secret and the public key actually makes this process even more secure, because if the value is intercepted it cannot be used as a secret key directly. The secret key has to be generated from the exchanged value after it has been successfully decrypted. Algorithms that can be used as part of the key exchange process include Diffie-Hellman (DH), RSA (named after its creators: Rivest, Shamir and Adelman ) and Elliptic Curve Cryptography (ECC).

### **Encryption**

Encryption includes the algorithms that are actually used to encrypt the data. Many of the algorithms used in key exchange can also be used to encrypt data. But due to the performance implication of asymmetric algorithms, they are not usually used to encrypt data beyond the initial key agreement phase. Symmetric algorithms are usually used to encrypt data during communication, once an asymmetric algorithm has been used to facilitate the exchange or generation of a symmetric key. Typical examples of the algorithms that are used are: DES (Data Encryption Standard), 3DES, IDEA, and the Rivest ciphers (RC2, RC4, RC5).

### **Message Digests**

As we have discussed earlier, hash functions are used to generate a message digest. Message digest is a value that can be used to test the integrity of the message and prove that it has not been tampered with. A hash function is that it takes some data as input and produces a 'digested' hash value that is dependent on the content of the input and the hash algorithm used.

A good hash function algorithm needs to be collision resistant. This means that two different input values should never produce the same hash value. Another characteristic of a robust hash function is that it must be difficult to derive the original input from the hash value and the algorithm alone. In this sense, these are one-way functions. In order to verify the integrity of the data it is necessary to re-hash it and compare the two hash values. A Message Authentication Code (MAC) can be used for this purpose. A MAC is basically a message digest that has been generated using a key. A number of hash algorithms have MAC versions. The most common hash functions are MD2, MD4, MD5 and SHA-1. When a message digest is combined with reversible asymmetric encryption it can be used to generate a signature.

## **20.6 Certificates**

Certificates are designed for the purpose of authentication and can be regarded as an electronic equivalent of a passport. The certificates contain information about the party being vouched for, called the subject of the certificates, including the subject's public key. In this respect, the purpose of the certificate is to vouch that the public key contained in the certificate belongs to the subject of the certificate. The certificate also contains some information about the authority that is vouching for the subject. This authority signs the certificate. Certificates may expire or become invalid for some reason.

### **Certificate Authorities**

Certificates are issued by a trusted organization called a Certification Authority (CA). In order to make the job of certification of a large numbers of users feasible; a decentralized certification mechanism is supported by delegating the certification authority. Certified parties can certify other parties by signing their certificates, and organizations can issue certificates for their own use.

## 20.7 Security in WAP

Security, as we know, is needed to safely connect to the services, such as online banking and e-commerce. The client and the server must be authenticated and the connection has to be encrypted. Man-in the middle attacks should also be prevented so that the data can not be modified during the transfer. The subscriber has to be sure that the service being offered is really the one it claims to be. In some cases, the service might also want to use a strong authentication. Although, several mobile networks encrypt the traffic in the air, the mobile network does not provide the complete end-to-end security.

### Wireless Security Challenges

The wireless mobile networks pose new challenges for implementing security architecture compared to the traditional connection oriented models like that used in the Internet.

Security in WAP architecture should enable services to be extended over potentially mobile networks while also preserving the integrity of the user data. The denial of service should also be prevented. The wireless mobile networks set many requirements to the security layer. The existing secure protocols cannot be used in mobile networks without adaptation.

The requirements to be met by the security layer in the WAP architecture are as follows:

- It should provide support for low data transfer rates.
- The amount of overhead should be small because of low bandwidth.
- The datagram transport layer should also be supported because of the nature of the wireless networks.
- It should be able to handle lost, duplicated and out of order datagrams without breaking the connection state.

Other issues include slow interactions, limited processing power, and memory capacity. The round-trip times can be long and the connection should not be closed because of that. For instance, the time between the request and response using the SMS bearer can be as long as 10 seconds. The cryptographic algorithms that are used must be light enough so that the mobile terminals are able to execute them. The number of cryptographic algorithms has to be minimized and small-sized algorithms must be used. The amount of available RAM in the mobile terminals must be taken into account. They also include the restrictions on exporting and using cryptography. Export laws in some countries do not let strong cryptography to be exported outside the country. There are also differences between exporting a strong authentication and encryption. In many cases, strong authentication is allowed but strong encryption is prohibited.

### Need for WTLS

If the WAP gateway acts as a proxy for the mobile device and uses normal HTTP 1.1 protocols then there is no reason why TLS cannot be used to secure communication between the WAP gateway and the web

server, exactly as it happens on the Internet. But TLS cannot be used to secure the communication between the WAP device and the WAP gateway, because TLS requires a reliable transport- in particular TCP – and the mobile device does not use TCP to communicate with the WAP gateway. TCP/IP is suited to higher bandwidth and more reliable networks than those encountered with a mobile device. A typical mobile device is too processor and resource challenged to run a TCP/IP protocol stack. Hence, the WAP Forum has designed another protocol stack that is lightweight and better suited to the environment. This protocol stack uses UDP over IP based networks, or WDP over non-IP networks.

To address the issue of security between the device and the WAP gateway the WAP forum has defined a new protocol, Wireless Transport Layer Security (WTLS). WTLS is based on TLS and provides a similar level of security. It is capable of running over WDP or UDP, and is optimized for use on devices that do not have significant processing resources. Therefore, there is one security mechanism in place from the device to the gateway, and another from the gateway to the web server. This implies that there must be a transition from WTLS to TLS at the gateway.

### **Wireless Transport Layer Security (WTLS)**

WTLS is the solution to the security issue, provided by the WAP Forum. WTLS is an optional layer and is based on TLS (Transport Layer Security) v1.0, which in turn is based on SSL (Secure Socket Layer) v3.0, which are Internet protocols. WTLS operates over the WDP.

The WTLS layer operates above the transport protocol layer and it provides the upper level layers of the WAP protocol with a secure transport service interface. The interface preserves the transport interface below it, and it also presents methods to manage secure connections.

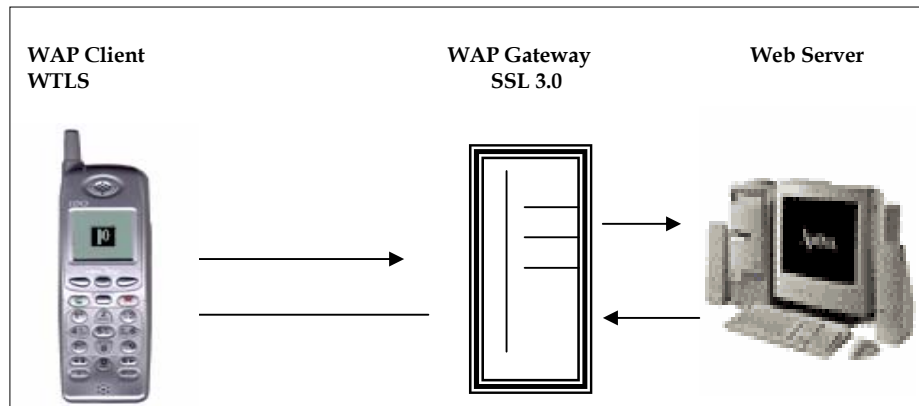
Since, WTLS is an optional layer in the WAP stack. This means that security in WAP is only available on demand and is not a built in feature of the WAP architecture. Hence, the information travelling to and from the WAP gateway is normally not encrypted, unless we use SSL connections to communicate between the origin servers and the gateway

The WTLS layer operates above the transport protocol layer and it provides the upper level layers of the WAP protocol with a secure transport service interface. The interface preserves the transport interface below it, and it also presents methods to manage secure connections.

## **20.8 Role of SSL and WTLS in WAP Communication**

Secure Sockets Layer (SSL) which is widely used in the “web” world to encrypt the data stream between the browser and the webserver is actually also used in the WAP environment.

In the above figure a standard SSL session is opened between the web server and the WAP gateway and a WTLS session is initialized between the gateway and the mobile device. WTLS is specialized for the wireless environment. The encrypted content is sent through this connection from the server to the gateway, which translates it and sends it to the mobile phone. However, there is a potential security problem where the two protocols meet, and that is inside the WAP gateway.



SSL is not directly compatible with WTLS. The translation between SSL and WTLS takes place in the memory of the WAP gateway. The WAP gateway decrypts the SSL protected data stream coming from the webserver and then re-encrypts it using WTLS before passing the data on to the WAP device. Inside the memory of the WAP gateway, the data is unprotected. It is important that unencrypted information is not stored anywhere in the gateway, since this defeats all the security measures that are used to protect the stored data from being seen by unauthorized people. This is the current model.

#### Methods to Manage Secure connection in WTLS

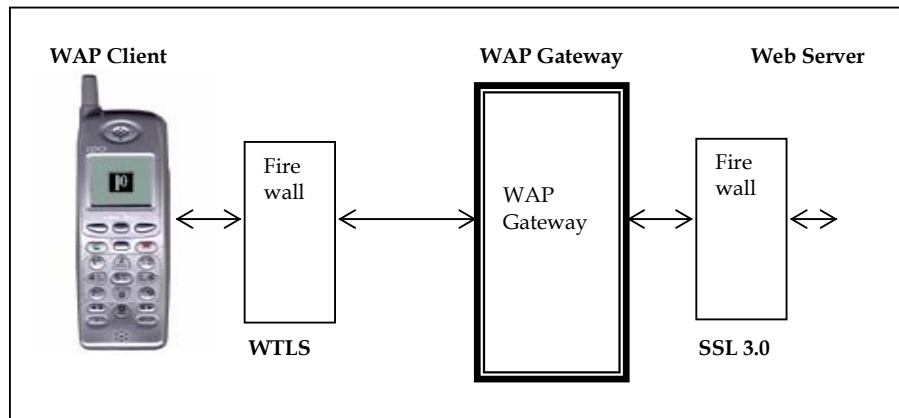
All the major WAP players are developing solutions to overcome this problem. The complete secure connection between the client and the service provider can be achieved in two different ways depending on where the WAP gateway is placed.

#### WAP Gateway in Service Provider's Network

The safest way is to place a WAP gateway in the service provider's own network. Then the whole connection between the client and the service can be trusted because the decryption will take place only when the transmission reaches the service provider's own network and not in the mobile operator's network.

If the WAP gateway is placed outside the mobile operator's network, the remote access server is needed. The origin server could be created to include the functionality of the WAP gateway. Developers of so the called "WAP servers", or web servers with WAP gateway capabilities are attempting to provide end-to-end security in a way because the data stream that leaves the content server (the "WAP Server") is already encrypted with WTLS. This gives the highest security solution available. The proposed model would look as follows:

But at present these solutions create other problems. In the proposed model, the mobile operator's WAP gateway can no longer be part of the chain, and the user has to reconfigure his WAP device to point of the "WAP server" which will become the WAP gateway for this session. But this WAP gateway only provides access to this one service, and when the user wants to access other favorite WAP sites the mobile phone has to be reconfigured again. Some WAP devices are fairly easy to reconfigure, while others are extremely difficult to reconfigure.



### WAP Gateway in the Mobile Operator's Network

The service and content providers can also trust the mobile operator's gateway and use virtual private networks to connect their servers to the WAP gateway. At present, the WAP gateway is commonly owned and operated by the mobile operator. But there is a potential drawback. The service providers do not have possibility to manage and control the parameters used by the WTLS at the WAP gateway. If the mobile operator's network is vulnerable to attacks, so is the data.

But the negotiating parties can decide on the security features that they want to utilize during the connection. According to the security requirements, the applications enable and disable the WTLS features. For instance privacy may be left out if the network already provides this service at the lower layer.

### Security in WTLS

WTLS, when present, provides the same level of security that is supplied by SSL 3.0. It provides services that ensure *privacy, server authentication, client authentication and data integrity*. These aspects are discussed below.

### Authentication

Authentication in WTLS is carried out with certificates. Authentication can occur between the client and the server or the client alone authenticates the server. The latter procedure can happen only if the server allows it to occur. The server can also require the client to authenticate itself to the server. However, the WTLS specification defines that authentication is an optional procedure.

### Privacy

Privacy in WTLS is implemented by means of encrypting the communication channel. The encryption methods to be used and all the necessary values for calculating the shared secret are exchanged between the client and the server at the time of initiating a secure connection.

### Integrity

Data integrity is ensured using the message authentication codes (MAC). The MAC algorithm that is to be used is decided at the same time as the encryption algorithm. The client sends a list of supported MAC algorithms where the preferred algorithm is the first in the list. The server returns the selected algorithm. The WTLS supports common MAC algorithms, such as SHA and MD5.

Although the WAP specification provides many features to supply the maximum level of security, there is still a lot of concern surrounding the WAP security solution. Banks and other companies that really have to protect their data, still prefer to host and install their own gateway, giving them the ability to send encrypted data right to the mobile phones, with no need for translation at the WAP gateway. Time will show whether WTLS will be gradually adopted as the standard or if it will just be ignored.

## 20.9 Short Summary

- ◆ WTLS is the security layer of WAP. Its primary goal is to provide privacy, data integrity, and authentication for WAP applications. It provides the transport service interface for the upper level layers. The interface is similar to the transport service interface below the WTLS. The WTLS is based on the well-known TLS v1.0 security layer used in the Internet. The wireless networks require support for both datagram and connection-oriented transport layer protocols. The mobile equipment sets requirements for the algorithms because of the limited processing power and memory. In addition, the low bandwidth must be dealt with and the restrictions on exporting and using cryptography must be considered.
- ◆ The WTLS incorporates new features such as datagram support, optimized packet size and handshake, and dynamic key refreshing. It has been optimized for low-bandwidth bearer networks with relatively long latency. Fast algorithms are chosen into the algorithm suite. The mobile equipment like cellular phone can be constructed to support only a set of cipher suites.
- ◆ The objective of WTLS is to be a lightweight and efficient protocol with respect to bandwidth, memory and processing power.

## 20.10 Brain Storm

1. What is the security layer in WAP?
2. What is Authentication?
3. What is Privacy?
4. What is integrity?
5. Explain WTLS?
6. Explain SSL and TLS.

---

## Push Technology and WTA

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ WAP push model
- ✔ Push Proxy Gateway
- ✔ Push Access Protocol
- ✔ Push Over The Air Protocol



---

## Coverage Plan

---

### Lecture 21

- 21.1 Snap Shot
- 21.2 WAP Push Model
- 21.3 Push Framework
- 21.4 Push Proxy Gateway
- 21.5 Push Access Protocol
- 21.6 Push Over The Air Protocol
- 21.7 Short Summary
- 21.8 Brain Storm

## 21.1 Snap Shot

Wireless Application Protocol (WAP) is a result of continuous work to define an industry-wide specification for developing applications that operate over wireless communication networks. The scope for the WAP forum is to define a set of specifications to be used by service applications. The Wireless market is growing very quickly and reaching new customers and providing new services. To enable operators and manufactures to meet the challenges in advanced services, differentiation and fast/flexible service creation, WAP defines a set of protocols in transport, session and application layers. In this chapter we shall deal with two important aspects of WAP 1.2 specification, i.e. **Push Technology** and **Wireless Telephony Application (WTA)**.

### Push vs Pull

In the normal client/server model, a client requests a service or information from a server, which then responds by transmitting information to the client. This is known as “pull” technology: the client “pulls” information from the server. The World Wide Web is a typical example, where a user enters a URL (the request) which is sent to a server, and the server answers by sending a Web page (the response) to the user.

In contrast to this, there is also “push” technology, which is also based on the client/server model, but where there is no explicit request from the client before the server transmits its content.

Another way to say this is that while “pull” transactions are always from the client, “push” transactions are server-initiated.

## 21.2 WAP Push Model

Generally, the various aspects of WAP enable content to be delivered to a wireless device, but only when the user of such a device requests the content. In other words, the content is constantly present in the network, but the users only gets to view it if they voluntarily request it, at least for the first time in a content pull session.

However, when we look at real-life applications, we realize that there is a lot of information that is useful to a user, but which a user does not see because they do not know when it is available or when there is a change in the status of information. Ideally such information should be “pushed” to the user, either at predefined intervals or when certain events occur that would then make the information useful to the user.

Some examples of possible push situations are the arrival of new emails and stock market information when certain stock price thresholds are crossed.

As a matter of fact, push services already exist in mobile phone networks, using the well-known Short Message Service (SMS) and the Cell – Broadcast mechanism in GSM networks. Many people already use SMS regularly, for information on key news items, scores from a sports game, or notification about a new message.

Here is an example of how push services work in a mobile phone. Suppose that a user has registered with a cellular network that provides messages as soon as a certain stock price crosses a threshold. On receiving the information, if the user chooses to buy or sell the particular stock, the user would have to separately

communicate with the broker to place the order, as would be the case in SMS. On the other hand, if this whole transaction is defined using WAP push technologies, the notification message could be combined with a menu option to follow a link to a site where the user can buy or sell a stock.

In the WAP model a complete push framework for delivering content to a user in an asynchronous manner, without the user being required to request the information has been defined. The push architecture has been defined for the first time in the WAP 1.2 specifications. This enables implementations of push to be standardized. Moreover, such implementations are independent of vendors.

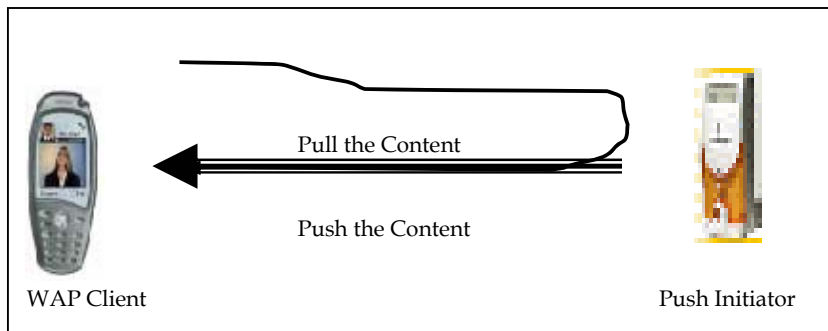


Fig: Comparison of pull vs. Push technology

Push technology in WAP is more likely to be successful than its counterpart on the Internet. This is due to fact mobile devices, to which the WAP content is targeted, are usually with the users. Moreover, push services in the wireless networks offer a definite advantage over its Internet counterpart - that is, the ability to produce location-based push content. For example, tourist or hotel information specific to a particular geographical area can be pushed to wireless device users in that area. This is impossible in internet-based push.

### 21.3 WAP Push Framework

The WAP push framework introduces a means within the WAP effort to transmit information to a device without a request from the user.

A push operation in WAP occurs when a *Push Initiator (PI)* transmits content to a client using either the *Push Over-The-Air (O TA) protocol* or the *Push Access Protocol (PAP)*. In its simplest form, the architecture would look like this:

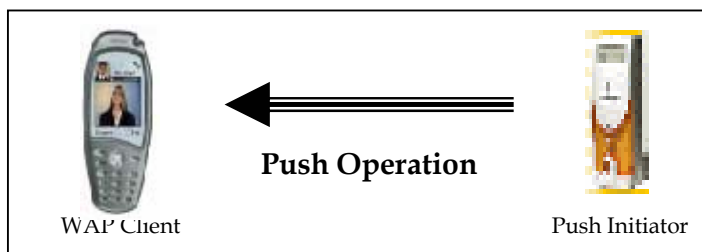


Fig. The Push Framework at its simplest

However, the PI shares no protocol with the WAP client. While the PI is on the Internet side, the WAP client is in the WAP domain. Therefore, the Push Initiator cannot contact the WAP client without an intermediary, so we need to insert a translating gateway. This translating gateway is called the **Push Proxy Gateway (PPG)**.

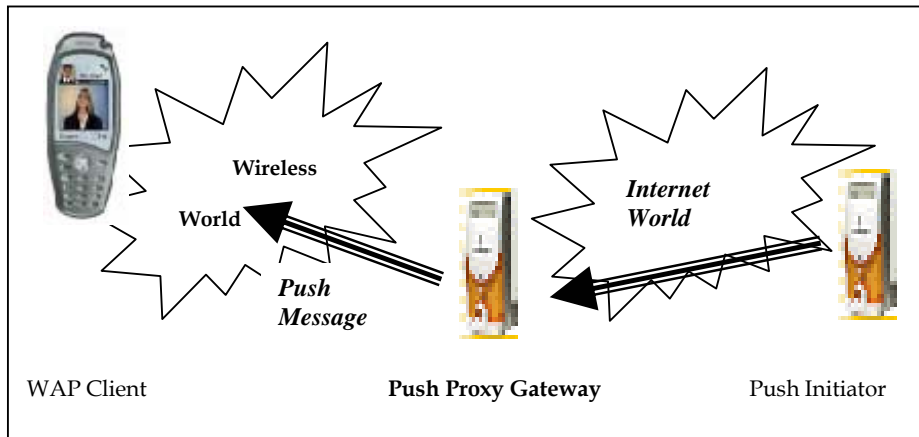


Fig : The Push Framework with the Push Proxy Gateway

Thus, the PI contacts the PPG from the Internet side, delivering content for the destination client using internet protocols. The PPG does what is necessary to push content to the WAP domain, and the content is then transmitted over-the-air in the mobile network to the destination client.

In addition to providing simple proxy gateway services, the PPG is capable of notifying the Push Initiator about the final outcome of the push operations, and it may wait for the client to accept or reject the content in two-way mobile networks. It may also provide the push initiator with client capability to lookup services.

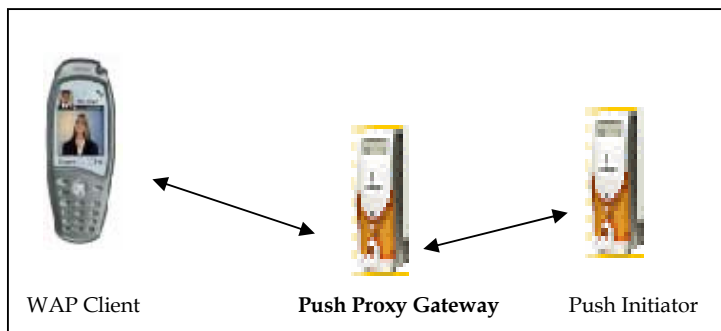


Fig: The Push Framework with protocols

The Internet-side PPG access protocol is called the Push Access Protocol (PAP). The WAP side Protocol is called the Push Over-The-Air Protocol (OTA).

The Push Access Protocol or PAP uses an XML message that may be tunneled through various well-known Internet protocols, for example HTTP. The OTA protocol is based on the WSP Services (WSP).

## 21.4 Push Proxy Gateway

The Push Proxy Gateway (PPG) is the entity that does most of the work in the Push architecture. Its responsibilities include acting as an access point for content pushes from the Internet to the mobile network.

The push proxy Gateway (PPG) has to implement the entire WAP protocol stack in addition to the two push specific protocols: PAP and Push OTA. Therefore, it makes sense to have push functionality as a component or add-on for a WAP gateway.

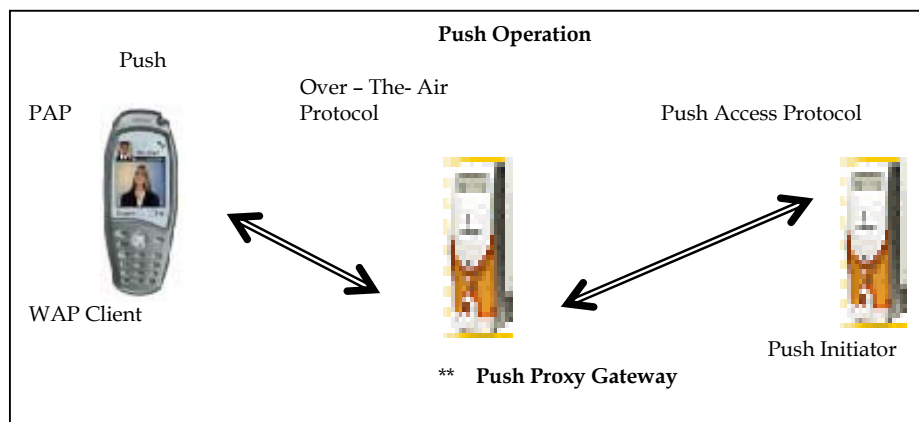


Fig: PPG highlighted

The PPG hosts the push framework with several services. Since, the entry point for pushed content from the Internet destined for the WAP domain, it must be able to perform the following:

- Push Initiator (PI) identification and access authentication
- Parsing and error detection in content control information
- Client discovery services
- Client Address resolution
- Binary encoding and compilation of certain content types to improve the efficiency OTA.
- Protocol Conversion

### PPG Functionality

In the following we shall discuss a few of the services that can be provided by the PPG.

#### Access from the Internet

The PPG accepts pushed content from the Internet using the Push Access Protocol (PAP). This content is divided into several sections using a multipart/related content type, where the first part contains information for the PPG itself. Such information includes recipient information, timeouts, callback requests, and similar pieces of information.

The PPG will acknowledge successful (or report unsuccessful) parsing of this control information, and may in addition report debug information about the content itself. It may also do a callback to the pushing server when the final status of the push submission (delivered, cancelled, expired, etc..) has been reached, if the push initiator so requests.

#### Message Handling Services

Once the content has been accepted for delivery, the PPG attempts to find the correct destination device and deliver the content to that client using the Push Over-The-Air Protocol. The PPG will attempt to deliver the content until a timeout expires. This timeout may be set by the PI and/or policies of the mobile operator.

#### Encoding and Compilation

The PPG may encode WAP content types (e.g., WML) into their binary counterparts, if feasible. This textual-to-binary translation would take place before the delivery Over-The-Air. It is also possible for the Push Initiator to precompile its content into binary form, to take workload of the PPG. When the PPG receives precompiled WML, WMLScript, they are forwarded as received.

### 21.5 Push Access Protocol (PAP)

PAP is a protocol that is used for delivering content from the Push Initiator (PI) to the Push Proxy Gateway (PPG), which in turn delivers it to the wireless devices. This PAP is an XML application.

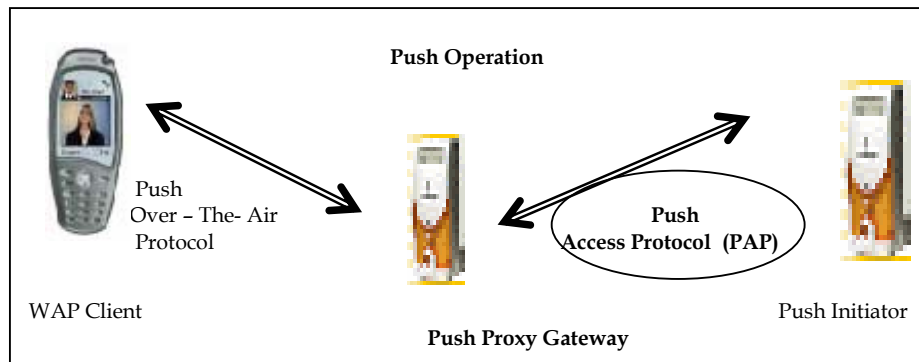


Fig: Push Access Protocol Highlighted

PAP defines the following operations as part of the protocol:

#### ◆ Push Submission

This operation, not only involves pushing the content from the PI to the PPG, but also giving the PPG enough information so that the message can be delivered Over-The-Air to the device.

#### ◆ Result Notification

The PPG informs the PI of the outcome of a previous push submission using this operation.

#### ◆ Push Cancellation

The PI might choose to cancel a particular push submission using this operator.

◆ Status Query

The PI can find out about the status of a previous push submission using this operation. This would typically be used before a result notification message has arrived.

◆ Client Capabilities Query (CCQ)

A PI might want to know about the capabilities of the client device that the content is to be pushed to, so that it can customize the content accordingly.

◆ Bad Message Response

This is a PAP message that does not fall under any particular category of operation. It is only used when the PPG is not able to determine the format of a received message. In other words, the received message is completely unrecognizable for the receiver to be able to parse the PAP entity.

### 21.6 Push Over-The- Air Protocol

The Push Over-The-Air (OTA) protocol is a thin protocol layer architecturally on top of the WSP. It is this part of the push framework that is responsible for transporting content from the PPG to the client and its user agents. It is implemented in both the mobile device and the PPG. It forms the bridge between a push user agent and the WSP layer.

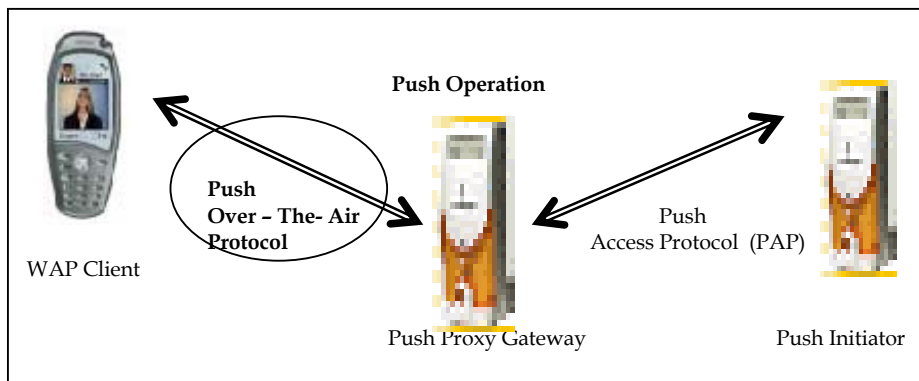


Fig : Push Over - The - Air (OTA) Protocol Highlighted

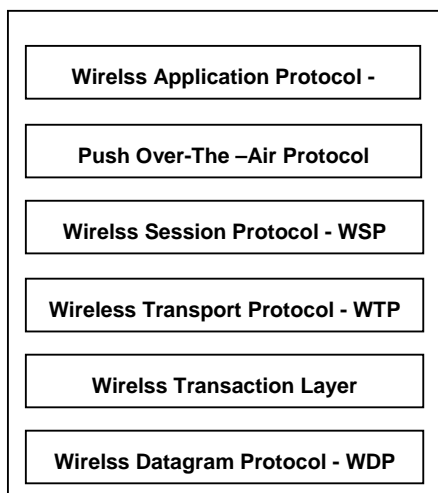


Fig WAP Protocol Architecture with OTA

There are different types of OTA push:

- Connectionless Push
- Unconfirmed connection-oriented push
- Confirmed connection-oriented push

**Connectionless Push**

A connectionless push is the simplest mechanism of the three options. Two registered WDP ports, one secure and the other non-secure, are reserved in every client capable of connectionless push. On receiving the message in the device, the OTA layer forwards it to the correct application, based on the application ID provided in the message headers.

**Unconfirmed connection-oriented Push**

This primitive is used to send the information from the server in an unconfirmed manner on a push session using the connection oriented service.

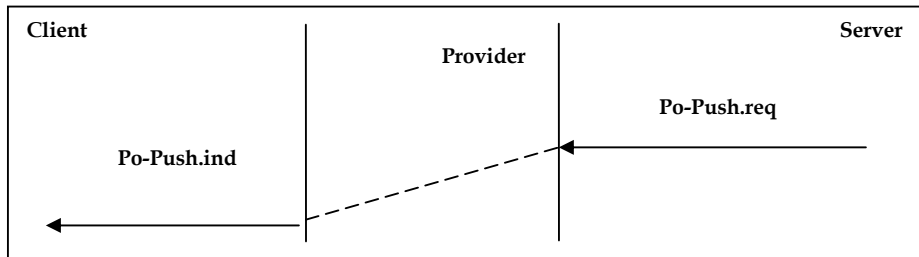


Fig: Unconfirmed Push

**Confirmed Connection –oriented Push**

A primitive is used to send information from the server in a confirmed manner on a Push session using the connection oriented service. It is the service user who confirms the push by response primitive (**Po-ConfirmedPush.res**) when the service user takes responsibility for the push message. If the service user can not take responsibility for the push message, it must abort the push operation by invoking the abort response (**Po-PushAbort.req**).

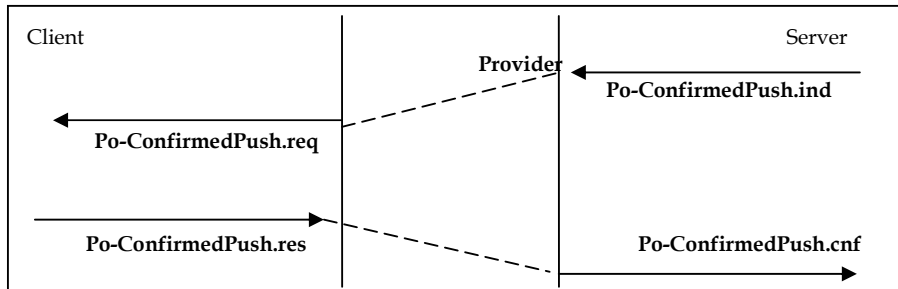


Fig: Confirmed Data Push

**Problems with PUSH technologies**



Wireless devices are a lot more personal than PC. Because of the nature of usage, there are privacy and unsolicited content issues associated with pushing content. Rouge PIs can push junk that could disturb users. Any content that the user has not previously registered for can be considered junk. Any content cannot be called junk if the user wants it! What is normal content to one is junk to another. Therefore, this distinction is to be made on a per-user basis.

With some packet-based wireless networks like GPRS, junk push content could become a major problem, as the device user ends up paying for the airtime. (for both incoming and outgoing packets). Unlike fixed networks, radio resources are expensive and scarce. Even legitimate push initiators should only push user notifications that are small in size and allow the user the choice of whether to load a service that might involve a lot more airtime usage. In spite of filter rules and PI authentication mechanisms being available with the PPG provided by the network operator, it may be possible for an anonymous PI to setup a PPG on the Internet to Push unsolicited content. This is totally dependent on the type of bearers used for push and security policies implemented by the wireless operator. If security policies are strict and only trusted parties are allowed to initiate traffic towards the devices, users are safer from spam/junk attacks. On the other hand, with bearers like SMS, it may be virtually impossible to solve spam/junk attacks unless the service is stopped entirely.

### 21.7 Short Summary

- K A push operation occurs when a push initiator transmits content to a client using the push OTA protocol or push access protocol
- K Push proxy gateway acts as an intermediary between the initiator and the WAP client.
- K Push OTA protocol is WAP side protocol and push access protocol is server side protocol.
- K The PPG may encode WAP content types into their binary contents if feasible. This textual-to-binary translation would take place before the delivery Over-The-Air.

### 21.8 Brain Storm

1. Explain briefly about WAP push model?
2. What is the function of PPG?
3. Explain about the push Over-The-Air protocol?
4. Discuss about the problems in PUSH technologies?



## Lecture 22

---

# Introduction to WTA

---

## Objectives

After completing this Lecture,  
you should be able to do the  
following

- ✎ WTA architecture
- ✎ WTA Examples

---

## Coverage Plan

---

<b>Lecture 22</b>
22.1 Snap Shot
22.2 WTA Architecture
22.3 WTA Services
22.4 Short Summary
22.6 Brain Storm

## 22.1 Snap Shot

WTA (Wireless Telephony Application) is an extension of the WAE (Wireless Application Environment) by providing a set of interfaces to the telephony functionality of a mobile device. These applications have mechanisms defined to interact with the telephony related functions present in the mobile phones and those provided by the phone network. Here are some of the telephony related functions that are available on a commonly available mobile phone:

- Making a mobile originated call
- Receiving a call
- Sending and Receiving Short Text Messages
- Adding, searching and removing phonebook entries
- Examining call logs, calls made, received or missed on the phone in the past
- Pressing Keys on the Keypad during a call

The WAP 1.2 specifications have defined WTA in a more precise manner than the previous versions.

## 22.2 WTA Architecture

The WTA architecture defines the components required in a client device to make telephony services available to WAP content.

The main components that constitute the WTA framework are

- WTA user agent
- WTA Server
- WTAI
- A persistent storage or "Repository"

### WTA User Agent

The WTA user agent is quite similar to a WAE user agent. It can execute and present WML and WMLScript content to a user like a WAE user agent. In a WAE user agent previously loaded decks are stored on the history stack. This does not happen in the case of WTA user agent. WTA user-agent is an extension of 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 a calls.

### WTA Server

The WTA server can be thought of as a web server delivering content requested by a client. Like an Internet web server a WTA user agent uses URLs to reference content on the WTA server. With the WTA server it is possible to create services that use the URLs to interact with the mobile network.

Typically WTA servers are under the administrative control of the operator so that the nature of the services provided is under the operator's complete control.

### Wireless Telephony Application Interface - (WTAI)

WTAI is a set of interfaces that extend the WAE to include telephony applications. All mobile phones that support the WAP1.2 specifications implement interface libraries on the phone that can be used to build telephony applications. These interface libraries consist of functions for various telephony and network

services. These functions be invoked either as URLs in a WML card/deck or via WMLScript functions executing in a WTA or WAE user agent context. Content providers can, therefore, write simple WML and WMLscript code for invoking the telephony functions of the mobile phones. These libraries provides a device independent and network independent interface that is completely under the control of the content provider.

### Repository

The Repository is a persistent storage module within the mobile terminal that may be used to eliminate the need for network access when loading and executing frequently used WTA services. This is an important component because, it helps in removing the latency associated with pulling the content over the network.

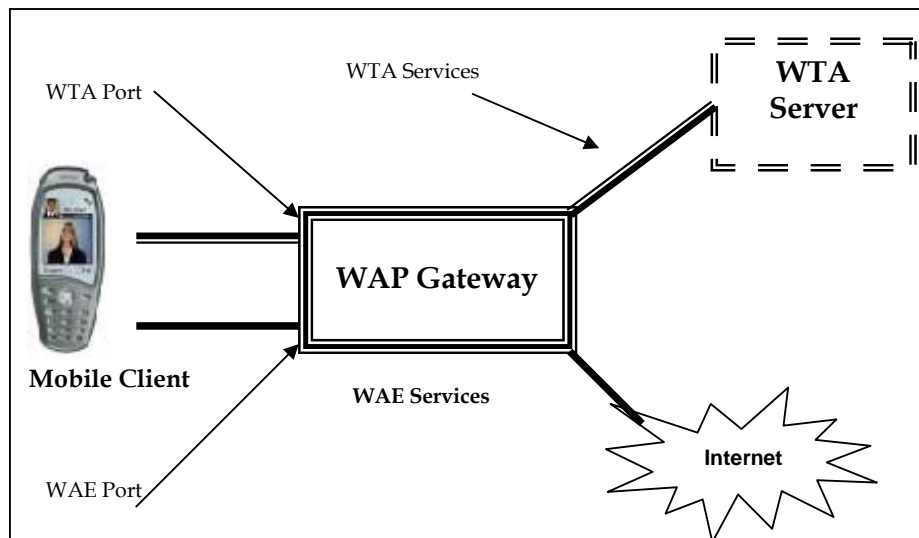


Fig: WTA overall view

## 22.3 WTA Services

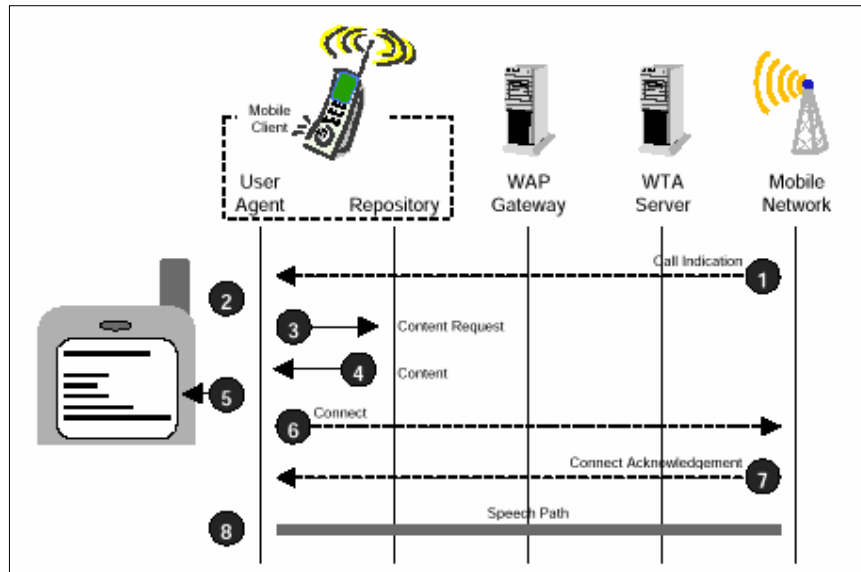
WTA services are what the end-user ultimately experiences on using the WTA framework. A WTA service appears to the client in the form of various formats, e.g. WML and WMLScript etc.,

The WTA user agent executes content that is persistently stored in the client's repository or content retrieved from a WTA server. The framework also allows the WTA user-agent to act on events from the mobile network (e.g., an incoming call). Given below are two examples of WTA services.

### Examples of WTA Services

#### 1. Incoming Call Selection

The “incoming Call selection” service is started when an incoming call is detected in the client, and a menu with various call-handling options is presented to the user. This is shown in figure



Given below are the explanation for the numbers in fig

1. The mobile network receives an incoming call and sends a “call indication” to the mobile subscriber.
2. In the client the incoming call WTA event is generated. The repository is consulted in order to find a dedicated channel. The channel provides the URL to the “Incoming Call Selection” service stored in the repository.
3. The user agent requests the content from the repository.
4. The repository returns the requested content.
5. The content is loaded into a clean context and starts executing. The service presents a list of options to the user from which the user can decide how to proceed with the call in progress In this example she selects to answer the call. The WTAI function “WTAIVoiceCallaccept” is invoked.
6. A connect request is sent to the mobile network.
7. A “connect Acknowledgement” is generated in the mobile network. A result code indicating the outcome of the call is generated internally in the phone.
8. A speech path between the mobile network and the client is established.

#### Make a Call from the WML Application

This function initiates a call, originating from a mobile device, using the specified phone number. Implementations of this function are required to display the phone number before the call is made.

The following application initiates a call to “Radiant WAP” on the request of the mobile user.

**Example 9.**

```

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

<wml>
  <card id="makecall" title="Radiant">
    <p align="center">
      Radiant - WAP
      For details...contact at...
      <do type="accept">
        <go href="wtai://wp/mc;912324566" />
      </do>
    </p>
  </card>
</wml>

```

The WTA services will be available to the mobile users only if such services are provided by the cellular service providers.

**22.4 Short Summary**

- K WTA applications interact with the telephony related functions present in the mobile phones and those provided by the phone network.
- K WTA useragent is an extension of the standard WML useragent with the addition of capabilities for interfacing with mobile network service available to a mobile telephony device.
- K WTAI Libraries provides a device independent and network independent interface that is completely under the control of the content provider.
- K The Repository is a persistent storage module within the mobile terminal that may be used to eliminate the need for network access when loading and executing frequently used WTA service.

**22.5 Brain Storm**

1. Explain briefly about WTA Architecture?
2. Discuss in detail about WTAI?
3. Short Note on WTA Server?
4. Explain about WTA Services?



---

## Wireless Technologies

---

### Objectives

After completing this Lecture, you should be able to do the following

- ✔ Imode Internet components
- ✔ Bluetooth
- ✔ Bluetooth Wireless Solution Components
- ✔ Bluetooth Security.



---

## Coverage Plan

---

<b>Lecture 23</b>
23.1 Snap Shot
23.2 I - mode Network Service
23.3 I - mode Internet Components
23.4 GPRS
23.5 Bluetooth Technology Overview
23.6 Bluetooth Wireless Solutions Components
23.7 Bluetooth Security.
23.8 Bluetooth Link Management
23.9 Short Summary
23.10 Brain Storm

### 23.1 Snap Shot

In 1999, NTT DoCoMO, Japan's leading cellular phone operator, launched a service called I-Mode. I-Mode (which stands for information -mode) is a mobile phone service which offers continuous Internet access. I-mode is similar to WAP. The reason DoCoMO decided to go with I-Mode instead of waiting for WAP is simple. The Japanese were ready to access the Internet through their mobile phones. They didn't want to wait for WAP to provide them with wireless data services they needed.

An I-Mode enabled cellular phone is similar to in appearance to most cellular phone models. One feature in particular is a four-point command navigation button at the center of the phone. This allows the user to control the pointer on the display, as connect to the I-Mode service by pressing a single button. There are several companies that manufacture I-Mode cellular phones, including Panasonic, Nokia, Ericsson, and Sony. However, NTTDoCoMo's models are the most popular within the industry.

### 23.2 I-Mode Network Service

The I-Mode service uses an additional packet communication network that is built onto DoCoMo's main network. This packet data transmission technology allows for constant connectivity. Thus, users are not charged for how long they are online, since this time is unlimited. Rather, users are charged only for how much information they retrieve. There are essentially four main components that are required for the I-Mode service. They are as follows:

A cellular phone capable of voice and packet communication and with a browser installed

- A packet network
- An I-Mode server
- Information providers

Unfortunately, the I-Mode service is currently available in Japan and Hong Kong only. However, there are plans in the works to bring I-Mode to parts of Europe in the near future. It is unknown at the moment if I-Mode will make it to the United States.

### 23.3 I-Mode Internet Components

In the case of the I-Mode Internet, an Internet server contains the I-Mode phone. These phones are now the clients. There are two other factors involved in connecting to a wireless network. In order to connect a cellular network to a sever, a gateway must exist. Also, the web site must be in I-Mode format.

#### Gateways

A gateway translates wireless request from a mobile phone to the server. It also sends information from a gateway back to the mobile phone. NTTDoCoMo provides a gateway to their users; however, this is only available to those in Japan. There are other gateways on the market that allow users outside of Japan to build new mobile Internet services based on cHTML.

#### I-Mode Enabled Site

Web pages today are often written in HTML (Hypertext Markup Language), Which is too complex for mobile phones because of their slower connection speeds. An I-Mode enable web sites utilizes pages that are written in cHTML (Compact Hypertext Markup Language), which is a subset of HTML designed for

devices with slower connection speeds. Today, the I-Mode service boasts 500+ I-Mode enabled websites linked to portal page, as well as 12,000+ "unofficial" web pages created by private individuals.

### **cHTML**

CHTML is extremely similar to HTML – in fact, it is HTML. The only difference is that some of the more resource intensive areas of the code (such as tables and frames) have been taken out. Mobile devices have a slower connectivity speed. Thus, by eliminating some of the more involved portions of the code, cHTML allows I-Mode web pages to download more quickly to mobile devices. The World Wide Web Consortium (WWW.W3.org) contains a complete listing of the cHTML tags available to developers. The NTT DoCoMo site at [www.nttdocomo/ser2.htm](http://www.nttdocomo/ser2.htm) gives an outline of cHTML tags available to developers.

### **Micro-browsers**

Most I-Mode phones today utilize a micro-browser. These usually have a title bar with icons at the top of an LCD screen. These icons then allow users to access various services such as weather forecasts, transportation schedules, data searches, and news updates. Below this title bar is a text screen that displays text messages and data.

One micro-browser in particular is Compact NetFront, developed by the Japanese company Access. Compact NetFront is used as the micro-HTML browser for about 75% of all I-Mode enabled devices.

### **I-Mode Applications**

The criteria for creating an I-Mode application or an I-Mode web page are essentially the same as creating web apps and web pages with HTML. The cHTML language must be developed and then the must to be loaded to an Internet web sever utilizing FTP or some other transfer method.

Currently, the cHTML language does not support scripting language (this being a major obstacle for developers). However, NTT DoCoMo and Sun Microsystems have plans to incorporate Java Jini, and Java Card technologies into I-Mode cellular phones.

## **23.4 GPRS**

The global GPRS (General Packet Radio Service) market is now beginning to take off. The introduction of GPRS is one of the key steps in the evolution of today's GSM networks to 3G, and GSM operators around the world are upgrading their networks, with a view to launching commercial GPRS services in 2000.

Data traffic is increasing enormously, and is expected to grow 40-50 per cent this year. This growth in demand for Internet access and services has paralleled the explosion in demand for mobile communications. Users want access to the Internet while they are away from their offices and homes, and GPRS can deliver this mobile Internet functionality.

With the capability to charge per data bit sent and received, customers will be able to pay only for usage. GPRS will offer a tenfold increase in data throughput rates, from 9.6kbit/s to 115kbit/s. Using a packet data service, subscribers are always connected and always on line so services will be easy and quick to access.

GPRS will allow innovative services to be created, enabling new and previously inaccessible market segments to be addressed, increasing customer loyalty and reducing churn. Machine-to-machine and person-to-machine communications will become possible.

The next stepping stone towards 3G will be the implementation of EDGE, offering data services and applications at speeds up to 384kbit/s essentially using existing infrastructure.

### **GPRS Introduction**

GSM network operators all over the world are developing a new packet data transmission technology known as GPRS (General Packet Radio Service) that will enable users mobile networks to connect at a speed of 115 Kbit/s, a rate more than ten times faster than the current standard.

With GPRS, the data transfer rate of mobile networks will overtake even fixed networks. But there are further benefits: GPRS will substantially lower operational costs both for network operator and subscribers.

Since data will be transferred in packets, charging will be on the basis of bits sent or received, and not on the length of time connected as is the case today.

Users will be able to stay on line permanently, to make the most of the quick access to services, like high-speed e-mail delivery, web surfing or access to LANs and company intranets.

Packet data transfer opens up a whole new series of opportunities for networks without cables. New machine-to-machine and man-to-machine interfaces will be developed. For instance, vending machines could use GPRS to ask for the replenishment or repair.

GPRS is a key step in the evolution of GSM networks towards third-generation capabilities. The EDGE(Enhanced Data rates for Global Evolution) high speed modulation technique will enable data services and application at speeds of 384bit/s, using existing infrastructure.

### **Always Online**

The wireless/mobile/cellular systems of today are sometimes called 2G (2nd Generation) systems, with the first being the early analog systems. With the boom of the Internet and the massive spread of both computers and cellular phones, it is likely that we will want to get access to more than voice services and applications when we are on the go.

The Mobile Internet is something that has started over the last year, with WAP, Wireless Application Protocol (Europe, USA), Palm.net (USA) and I-Mode (Japan). What WAP does to the cell phone is what the web browser did to the computer. It enables non-Rocket-scientists to access the information anytime, anywhere and on any device. WAP is an open standard that is now rapidly spreading throughout the world. If you want to use WAP today, you will connect to the servers with a cell phone using circuit switched data. That works much like computer modems, where you dial in to an ISP (Internet service provider), do your e-mail and surf the web and then disconnect. If you want to do it again, you have to dial in again. Compare that to the connection some of us have at work or at school, where your connection to the Internet is always there, always online. The web browser starts as quickly as a word processor, and the remote Internet services are just one click away. That is exactly the functionality that GPRS adds to the cell phones and the Mobile Internet, enabling the user to be always connected, always online (without necessarily having to pay by the minute). The functionality means a tremendous change in the way we will use the cell phone and probably also the home PC.

**An upgrade to existing systems**

GPRS is not a completely new system, but rather an upgrade that empowers existing GSM and TDMA networks (US: PacBell, Voicestream, AT&T etc, Europe and Asia: Everywhere but in Japan and a few other spots). This means that you are still going to have the same functionality for voice calls and it is even possible to have simultaneous voice and data for some handsets. This smooth migration also means that you are going to enjoy the same coverage for GPRS as for present cellular networks, as opposed to building a completely new network from scratch. That way, your present phone will work in the future as well, but you will need a new handset in order to access the new features.

**An integral part of 3G wireless systems**

3rd Generation wireless systems will bring high speed multimedia services to the Mobile Internet. You will not only be able to surf the web fast, and download e-mail. You will also see a million new applications where high quality voice, text, pictures and interaction is blended.

**Applications**

The area of Mobile Internet applications is one of the most intense in the world. The operators that have bought billions worth of equipment are now asking: What should we (and our customers, the users) use it for? It is likely going to be such an important question that it will make and break players in the business. Who wants to buy a subscription if there is nothing fun or useful to do with it? Of course, there will be a huge variety of Internet content available but that is only the start of it. The Mobile Internet applications will be so much more. Imagine what could be done if your device knew roughly where you were? The possibilities for the 'Find the closest Starbucks shop' kind of applications show where that could take us. Coming to a new city and clicking on an icon to see what is happening close to your hotel that weekend (all according to your own preset interests, of course). Also, the mobile device that you will use for all these services is much more personal than any other Internet access device. The things you will have in there will reflect your interests and personality. The personalization will take the devices into things that make life easier, not more complicated. That is also in essence what GPRS does for the applications. Where you before dialed in and waited, the information will now only be a click away. Checking the weather where you are will be as easy as looking up a number in the phonebook.

Typical GPRS applications will utilize the always online functionality to its full extent. Things that reside inside your device will do things for you, when you have better things to do. Say you want to buy stocks when the Nasdaq drops below 3000, then you can have your GPRS handset vibrate whenever that happens (while you are hanging out at the beach).

In developing applications for GPRS it is necessary to understand both the new possibilities and new challenges. The bandwidth will vary and sometimes be very low, you will temporarily lose contact with the network and there will be packet loss. Wireless networks are different and you need to consider these properties when developing applications for them.

**23.5 Bluetooth Wireless Technology Overview**

Bluetooth wireless technology is a specification designed to enable wireless communication between small, mobile device. The inspiration behind this technology was the concepts eliminate the need for the proprietary cables, which are currently required to enable device connectivity. For instance, in order to

transfer images from a digital camera to a laptop PC, a cable is needed in order to connect the camera to the laptop. Each camera manufacturer and model has a different cable requirement. In fact every hand held device manufactured which allows connectivity with a PC has a different cable configuration. Imagine a scenario in which both the laptop PC and the digital camera use Bluetooth wireless technology. In this case there is no need for cables to transfer data between devices. Expanding that idea to include all hand held mobile electronic devices is, in a nutshell, the Bluetooth wireless technology vision.

In addition to eliminating the need for cables to connect devices, Bluetooth enables devices to form small, ad hoc wireless networks called piconets. These wireless connections are established using a radio transceiver embedded within each Bluetooth device. The radio operates in the 2.4 GHz Industrial, Scientific, and Medical (ISM) band which is globally available[1]. The Bluetooth Radio is designed to operate in a noisy radio environment and to provide a fast, robust, and secure connection between devices. A full duplex data exchange rate of up to 1 Mb/s may be achieved in which a Time-Division Duplex(TDD) scheme is used. Stability is ensured by implementing a frequency-hopping scheme which enables Bluetooth Radio modules to avoid interference from other signals after transmitting or receiving a data packet. Security within Bluetooth connections is implemented at the hardware layer, with the option of using one of three security levels.

#### **How does Bluetooth work?**

Bluetooth wireless technology works by using a radio transceiver, which basically acts as a short-range walkie-talkie. This radio allows the device to send short-range radio signals which look for other Bluetooth devices. When another device is found, the devices begin to communicate with each other and can exchange information.

### **23.6 Bluetooth Wireless Solution Components**

There are four major components in any Bluetooth wireless technology system: a radio unit, a baseband unit, a software stack, and application software. The radio unit is the actual radio transceiver which enables the wireless link between Bluetooth devices. The baseband unit is hardware, consisting of flash memory and a CPU, which interfaces with the radio unit and the host device electronics at the hardware level. The baseband hardware provides all required functionality to establish and maintain a Bluetooth wireless connection between devices. The software stack is essentially driver software or firmware which enables the application level software to interface with the bandband unit. The application software implements the user interface and overall functionality of the Bluetooth device.

### **23.7 Bluetooth Security**

The Bluetooth specification defines three security modes: non-secure, service-level security, and link level security. In the non-secure mode, the device does not initiate any kind of security procedure. In the service-level security mode, more flexibility in application access policies is allowed. Service-level security mode is especially useful when running several applications in parallel with differing security requirements. In link level security mode, the device sets up security procedures before the link set-up is completed. Link level security provides application with knowledge of "who" is at the other end of the link and provide authentication, authorization, and encryption service.

Authentication is a key component in any Bluetooth system which allows the user to develop a domain of trust between Bluetooth devices. Authentication services allow two devices to decide if a connection will be formed based on available identification at the hardware level. Once a connection has been established,

additional security may be applied to the data transmission using encryption. Encryption procedures are applied to an existing connection between devices while authentication procedures dictate whether or not a connection will ever be formed. The built-in Bluetooth security mechanisms are secure enough for most application. However, in the event that the built in mechanisms are not sufficient, stronger encryption schemes may be built into Bluetooth products at the software application level.

### 23.8 Bluetooth Link Management

The Link Manager(LM) is the software entity within the baseband which implements among other protocols, link setup, link authentication, and link configuration. When the LM discovers other remote LMs it communicate with them via the Link Manager Protocol (LMP). In order to perform its service provider role, the LM uses the services of the Link Controller(LC) which is the hardware entity enabling the creation of physical links with other devices.

The services provided by the Link Manager are:

- Transmitting and receiving data
- Requesting a remote device name
- Inquiring for a remote device link address (inquiry scan procedure)
- Negotiating and setting up the connection and link mode(ACL and /or SCO links)
- Authentication
- Determining the frame type on a packet-by packet basis
- Setting a device in one of the three low power modes (hold,sniff, and park)
- Ensuring the master only starts transmission in specified, regularly spaced time slots

#### Bluetooth Application Software

All device using Bluetooth wireless technology are required to support baseline interoperability feature requirements. The feature requirements are defined within the Bluetooth Profile Specification. The requirements may vary widely depending on the nature of the device. For instance, a device which provides LAN access using Bluetooth wireless technology would have far more feature requirements than a Bluetooth wireless mobile phone headset. The primary goal of the Bluetooth profile requirements is to ensure that any device displaying the Bluetooth logo will interoperate with other Bluetooth devices. All devices which use Bluetooth Wireless technology must be able to recognize each other and to discover the higher level abilities each device supports.

### 23.9 Short Summary

- K I - Mode stands for Information - mode. It is a mobile phone service which offers internet access.
- K A Gateway translates wireless request from a mobile phone to server. It also sends information back to the mobile phone.
- K Compact Netfront is an micro-browser, developed by Japanese Company Access. It is used by most of the I - mode enabled devices.
- K The EDGE high speed modulation technique will enable data services and applications at speed of 384 bit/s using existing infrastructure.

- K Bluetooth Wireless Technology is a specification designed to enable wireless communication between small, mobile device.

### **23.10 Brain Storm**

1. Explain briefly about I - mode Internet Concepts?
2. Short Note on GPRS?
3. How does Bluetooth Work?
4. Explain about Bluetooth Security?
5. Discuss about the Link Management in Bluetooth?

END



---

## Syllabus for MS(IT&EC)

### MS 3.6 Wireless Internet Architecture

#### Lecture 1 Introduction to WAP

---

Background of WAP-WAP Overview-Characteristics Features of WAP-Applications of WAP-Benefits of Adopting WAP-Limitations of WAP-Desired Characteristics of a WAP Solution-Future Developments in WAP

#### Lecture 2 WAP Communication Model

---

World Wide Web Architecture Overview -WAP Model-WAP Implemented Wireless Networks-WAP Communication Model-WAP Portals

#### Lecture 3 WAP Protocol Architecture

---

Protocol and Layers-WAP stack and WEB stack-Wireless Application Environment (WAE)-WAE Useragent-WTA-Wireless Session Protocol (WSP)- Wireless Transport Protocol (WTP)- Wireless Transport Layer Security (WTLS)-Wireless Datagram Protocol (WDP)

#### Lecture 4 Introduction to WML

---

WML Structure-Characteristics of WML -A Basic WML Card-The Header-The Body

#### Lecture 5 Style Element, Tables and Entities

---

Text formatted Elements- <head> element-Tables-WML entities

#### Lecture 6 WML Elements

---

Navigational Elements-WML Task Elements-Template Elements

#### Lecture 7 Links and Images

---

Links -Images-Interacting with Users

#### Lecture 8 WML Events

---

ontimer event-onenterforward event-onenterbackward event -onpick event-variables

#### Lecture 9 WML Script

---

WML Script Language Syntax - How to call a WMLScript from a WML page-Data types -Operators

#### Lecture10 WML Script Controls and Functions

---

Flow Control Statements-Functions-break statement-continue statement-return statement

#### Lecture11 WML Script Library

---

Libraries-Dialogs Library-Float Library-Lang Library

#### Lecture12 WML Script Library

---

String Library-URL Library-WML Browser Library

Wireless Internet Architecture

---

Lecture13 ASP and Object Model

---

ASP object Model

Lecture14 WAP MIME Types

---

MIME types -Request and Response Object.

Lecture15 Introduction to ADO

---

Cookies- ADO Object Models

Lecture16 Database Handling in WAP

---

Receiving Values from the WAP client and stored in a Database - Retrieving from the Database and send to the client.

Lecture17 Using Servlets in WAP

---

Power of Servlets-Servlet API-Basic Servlets Structure-Life Cycle of Servlets-Request and Response Headers -Servlet and Cookies

Lecture18 WAP and JSP

---

JSP Request Model-JSP Architecture - Using JDBC in JSP - JSP Error Page-Application using JSP.

Lecture19 WAP Security

---

Security concepts

Lecture20 Encryption Technologies

---

Cryptography -Keys- Ciphers-Cipher Suites-Certificates-Security in WAP-Role of SSL and WTLS in WAP Communication

Lecture21 Push Technology and WTA

---

WAP push model-Push framework-Push Proxy Gateway-Push Access Protocol-Push Over The Air Protocol

Lecture22 Introduction to WTA

---

WTA architecture-WTA Services.

Lecture23 Wireless Technologies

---

I-mode-Introduction-I-mode Internet components-GPRS -Bluetooth Wireless Solution Components-Bluetooth-Link management-Bluetooth Security.