Master's Thesis in
Computer Science

Approved
September 19, 2000

Development of WAP-services

by
Fredrik Axelsson
The Royal Institute of Technology
Kungliga Tekniska Högskolan

Examiner:	Prof. Seif Haridi Department of Teleinformatics Royal Institute of Technology
Supervisor:	Thomas Sjöland Department of Teleinformatics Royal Institute of Technology
Supervisor:	Lennart Öhman Sjöland & Thyselius Telecom AB

Abstract

This thesis is about the Wireless Application Protocol (WAP) version 1.1 and the development of services based on that protocol. It is both a study on the WAP protocol and its different parts, as well as a description of various methods to provide information to users through WAP enabled devices.

The studies of the protocol include an overview of the architecture, a description of the different layers of the protocol stack and an introduction to how the security layer works. Some advantages with WAP as well as alternatives to the protocol within the wireless domain are also included. For the future development of WAP services there is furthermore a closer look at the page describing language WML, some programming tools, environments and techniques.

Some implementations are also included within the thesis. Those are about session management with WAP, conversion of a web application into a WAP solution and a stand-alone system with a complete WAP structure on a single computer. The last-mentioned technique is useful to achieve higher security or to enable access control.

Contents

List of figures

List of tables

1 Introduction

1.1 Summary

1.1.1 WAP at a glance

The Wireless Application Protocol (WAP) is an effort to make Internet available to mobile phones and similar devices, which are communicating over a wireless network. It is designed to use as much as possible of existing technologies, with the World Wide Web as the main component. Other important factors in the protocol design are the differences between desktop computers and handheld devices. Examples of such differences are the limited power, display and input capabilities. Since WAP is using the web, the service providers do not need to buy new servers or any other hardware devices at all. Instead the company's existing web server is fully capable of delivering WAP contents after some minor configuration of the web server software.

Between the wireless domain and the web there is a gateway that converts the WAP requests into HTTP ditto. The gateway also converts the information returned by the web server back to the WAP protocol. The documents containing the information sent over the wireless domain are encoded to save bandwidth and workload on the client. The page describing language for ordinary web browsers is replaced by a simpler one more adapted to the user interface of a mobile phone. This language is based on the Extensible Markup Language (XML) and is called Wireless Markup Language, or WML.

1.1.2 Development of WAP services

Both of the telecom corporations Ericsson and Nokia offer Software Development Kits to WAP application developers. They also provide WAP gateways and mobile phones as well as browser emulators. An emulator is a handy tool for the developer to check out the result of the development before it is released for use on the real devices. Besides those tools, all the development environments used in web programming is useful for development of WAP applications as well. However, there are some major differences between the browsers used by the web and WAP clients. This makes some of the features unusable in some of the development environments.

1.2 Goals of this project

There were two primary goals with this project. The first goal was to look at the capabilities of WAP and to answer some questions. The second goal with the project was to develop a couple of applications using WAP.

1.2.1 WAP studies

When studying available literature, these where the characteristics of WAP that had to be determined:

- Background of WAP and why it was developed
- Description of WAP, i.e. how does it work
- User interactivity
- Advantages and limitations
- Alternatives
- How to develop WAP services
- Future of WAP

Since this is a very young and changing technology there is a lack of printed literature on the market. Therefore much of the background material for this report had to be found on the Internet. Besides the above-mentioned characteristics there were also some questions asked that needed answers. Those questions were:

- Is WAP a good solution for the mobile Internet?
- Is there a future for WAP, e.g. with the new network bearers?
- Is WAP something for the company Sjöland & Thyselius to concentrate resources on?

1.2.2 Implementations

The major application development was to convert a web site into a WAP site. The functionality of the web site is for Swedish citizens to calculate their future pension. Even if there might not be a need for a WAP implementation in this case, it will be a good example of the degree of reusability of advanced web applications. One of the other implementations was to realize a stand-alone system, which includes all the different parts that WAP is built upon.

- **Session management**
  Since WAP did not inherit all the functionalities of the web, there is some comfortable functionality for the developer that is missing in the development of WAP services. One of those functionalities is the management of user sessions. This is made very easily for web development, e.g. when using the Active Server Pages. In WAP however, this session management must be handled completely by the developer and this implementation is an example of how to achieve such functionality.
- **Pension calculation**
  The goal of this implementation was to provide a WAP service where people can calculate their future pension. This service was already available at the Internet through the web site http://www.pension.nu. In the web version of the pension calculations there was a client-side script that calculates a person's upcoming pension based on values entered on the web page. The script that was used for the calculations was a very large script written in the programming language JavaScript. Since the mobile phones on the market have a very limited memory and lacks a JavaScript interpreter, a different solution seemed necessary.
- **Own gateway with a modem**
  This implementation was about setting up a stand-alone WAP system. It includes a computer with a modem, a WAP gateway and a web server. This was to test the possibilities of offering WAP services without going through the Internet. The test was to be run on two different platforms, Windows NT and Linux. One practical use for this kind of service could be the need to avoid the security hazards with Internet traffic or to get faster data deliverance by avoiding the net.

1.3 Organization of this report

In the first sections, i.e. 2 and 3, the background of WAP and an architecture overview is discussed. This discussion includes the base on which WAP is built and under which conditions the protocol was developed. The foundation of WAP is to use as much of existing technologies as possible as well as to adopt the standard to the capabilities of existing user devices.

Sections 4 and 5 deals with the WAP protocol stack. It is a brief look at all the layers of the stack and their respective role. It begins with the uppermost layer, the application environment, which contains the user interface. At the bottom of the stack there is a datagram layer, which either operates above a network bearer or is replaced by the UDP protocol. The latter is the case if the bearer already supports UDP. In the session and transaction layers, different configurations are possible, i.e. connection-less mode or connection-oriented. If the connection-oriented mode is used, three levels of transactions can be selected, all with different reliability.

This part of the report ends with a look at the current bearers on the market today as well as in the future.

Advantages with WAP are discussed in Section 6, mostly in contrast to the HTTP protocol used by the web. The main advantages lies in the optimization of the numbers of datagram packets sent over the air and the power savings in the user's device.

Security is discussed in Section 8. It begins with an explanation of the encryption techniques used today and is followed by a description of how Internet security with the Secure Socket Layer, or shorter SSL, protocol is solved on the web. Since WAP is using the web as a vital part for its communication, it is also using the SSL to achieve security on the web. To solve the security issue on the wireless domain, WAP is using the Wireless Transport Layer Security (WTLS) protocol. This protocol is based on a descendant of SSL.

The markup language used to describe the contents of WAP documents is called Wireless Markup Language (WML). This language and the script language WMLScript are described in Sections 9 and 10. Most of the tags available in WML 1.1, i.e. the WML version included in version 1.1 of WAP, are described as well as some characteristics of the language. Since WML is based on XML it is very strict on how the tags are placed in the document. HTML on the other hand is a much looser language, where the browsers are very forgiving to developers who write erroneous code. WMLScript is a script language related to JavaScript but adapted to the kind of devices WAP is normally used on.

Several WAP development tools and test applications are reviewed in Sections 11 and 12. There are for instance two development kits from Nokia and Ericsson, which work very well to write static documents or the foundation of dynamic documents. If the WAP site contains documents with fluctuating data the document must be built at runtime on the web server. This can be managed with one of the server side scripts available, such as CGI or ASP. If there is a need for user sessions, e.g. on a site with multiple documents based on user related information, a session management functionality must be written. Since WAP lacks the cookie functionality a web browser uses, it is up to the developer to program the session management. This is described in Section 13, which is followed by a description of an implementation in ASP in Section 14.

The two following sections are about the implementations *Pension calculation* and *Own gateway with a modem*, which are described above. The last section is about the conclusions of this thesis and the answers and solutions to the questions and problems stated above, within the project goals section.

2 Background to WAP

2.1 Motivation

Internet has proven to be both an easy and efficient way to publish data and to offer services to millions of people. Most of the contents on Internet are designed for users running desktop computers with fast access to data and without limitations of power. The differences for the users with mobile handheld devices, such as mobile phones, to reach information on Internet are the limitations of capacity. Examples are less powerful CPUs, less memory, restricted power consumption, smaller displays and different input devices. The wireless networks also tend to have some major limitations in opposite to ordinary telephone- or broadband networks with less bandwidth, more latency, less stability in connections and that the availability is less predictable.

2.2 WAP Forum

Ericsson, Motorola, Nokia and Unwired Planet took the initiative to create a standard for development of services for the wireless community on June 26 1997. At the end of the year WAP Forum was created and the first release of the WAP specification was released in February 1998. The goal of WAP Forum is to develop a license-free standard for bringing information and services to wireless devices [1]. Among the requirements of the WAP architecture is to use existing technologies wherever possible, support as many networks as possible and to optimize for narrowband bearers. By using existing technologies the standards will reach the market faster as well as keep the prices in developing and running applications down [18].

3 Architecture overview

3.1 The World Wide Web model

The WWW model, or simply the web, used on the Internet gives a client the possibility to receive contents in a well-specified data format from web servers. The communication is handled through standard networking protocols such as HTTP and TCP/IP. To reach the content on the server the client uses addresses in a standard naming model called Uniform Resource Locator (URL) as shown in Figure 3.1 The client uses a Web Browser to view the content provided and among the formats supported are a language to describe the appearance of the content called HyperText Mark-up Language (HTML) and a script language to enhance the content functionality called JavaScript.

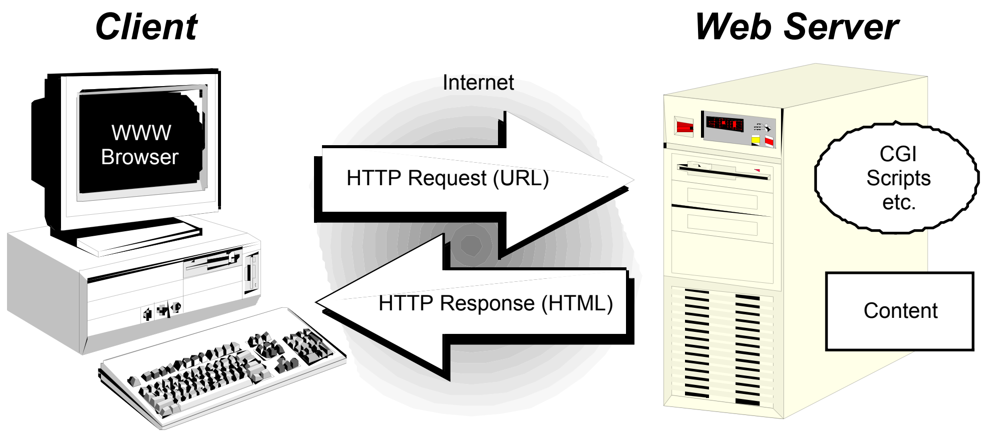


Figure 3.1: The World Wide Web model.

3.2 The WAP model

The WAP protocol is designed to use as much of existing technologies and standards as possible. A browser in the WAP device communicates with a WAP gateway (or proxy) connected to the Internet. The gateway translates requests from the WAP protocol stack to the WWW protocol stack (HTTP and TCP/IP) and vice versa. Since all communication between the gateway and the WAP client is binary encoded to reduce network traffic, the gateway also encodes and decodes all messages respectively.

When the browser sends a request the gateway decodes it to plain text and then forwards the request to the web-server containing the desired content as illustrated in Figure 3.2. In this way a content provider only needs to add a few content types to the web server to enable WAP services to be developed since the user of the WAP device is always connected to the same gateway. This leads to the fact that WAP uses the same naming model as web applications to point out content on remote servers by using URLs. The standard content formats used by WAP applications is based on WWW technology including a markup language called Wireless Markup Language (WML), calendar information, a scripting language by the name WMLScript and so forth.

When a server replies, the desired content is sent to the gateway. The gateway encodes the information into the binary form of WML it uses for the communication with the WAP device. The binary encoding compresses the tags and the header information of the WML document. Each tag in the document is replaced by a two-byte value, i.e. no more data than a single character. The textual content is not compressed but all unnecessary spaces and line breaks are removed. This saves both bandwidth on the communication channel and power on the client.

The latter since the document is much easier for the device to parse. If the content is in HTML the gateway tries to translate it to WML before the encoding.

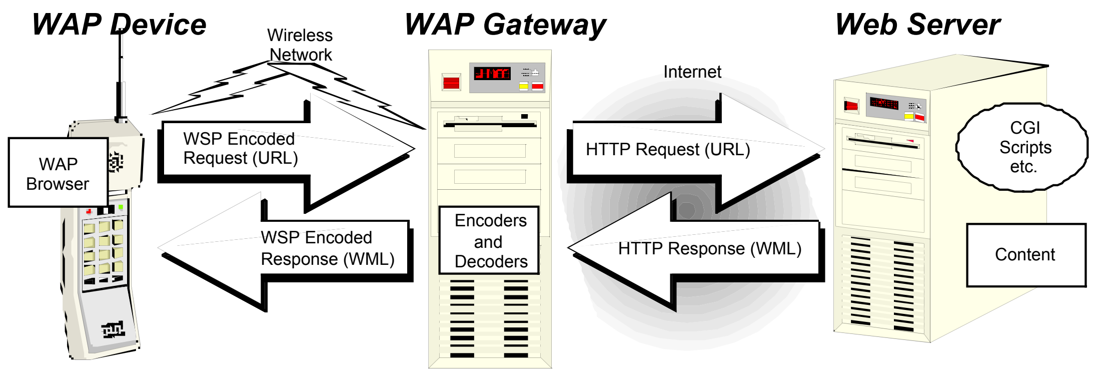


Figure 3.2: The WAP model.

4 Wireless Application Environment, WAE

The Wireless Application Environment (WAE) is the uppermost layer in the WAP protocol stack. It combines World Wide Web and mobile telephony technologies and the effort is to provide a common environment. This will enable operators and service providers to reach different wireless platforms in an efficient and useful way.

The WAE is divided in two logical layers, one for user agents and one for services and formats. The first layer is for user agents, e.g. browsers, phonebooks and message editors. In the second layer, the one with services and formats, there are different elements available such as WML, WMLScript, image formats, card and calendar formats and so forth. Those services and formats are accessible to and used by the different user agents. This logical view is assumed by the WAE but it is not necessary for an implementation to follow it, e.g. there can be a user agent that supports all the different services and functions.

4.1 Addressing model

An important service in WAE is the URL service. WAE uses the same addressing model that is used on the Internet and relies therefore on the HTTP and HTML URL semantics. Beside the URL, which identifies resources on a server that can be reached by well-known protocols, the WAE also supports Uniform Resource Identifiers (URI), which is used to locate resources that are accessed without using well-known protocols, such as wireless devices' telephony functions.

4.2 Wireless Markup Language, WML

The WML user agent is a fundamental user agent in WAE, even if an implementation of WAE is not limited to include one. It supports WML, WMLScript or both of them.

WML is a markup language derived from XML and similar to HTML used in the WWW model but it is optimized for small handheld devices with limited display and user input function. In the communication between a WAP gateway and a client, the WML code is encoded to a binary form to reduce the amount of data to transmit and therefore saves bandwidth.

A WML document, called a deck, is divided into one or more cards. A card contains the data that is displayed at once on the client's device, either to visualize data or to receive user input. The first card of a deck is automatically displayed when the deck is loaded and then the user can be allowed to continue to other cards in the deck. In this way the user can get moderately pieces of information for the limited display size on the device. At the same time the network traffic is reduced since several cards are downloaded simultaneously, which requires less requests and acknowledgements.

4.3 WMLScript

WMLScript is a lightweight script based on ECMAScript and therefore similar to JavaScript. It is used to add intelligence to the client, improve the UI, reduce the needs for roundtrips etc. WMLScript also support libraries that contain functions that extend the basic functionality. This also provides the ability to expand the language without having to change the core of WMLScript.

4.4 Wireless Telephony Application, WTA

The Wireless Telephony Application, WTA, is another user agent available in the WAE. It provides the ability to set up phone calls and enables network operators to provide advanced telephony services. Because of the nature of these services only the network operator or some service provider trusted by the network operator is allowed to present content to a WTA user-agent. For this reason there is a special WTA domain totally controlled by the network operator that provides WTA services. The separation between the domains is shown in Figure 4.1.

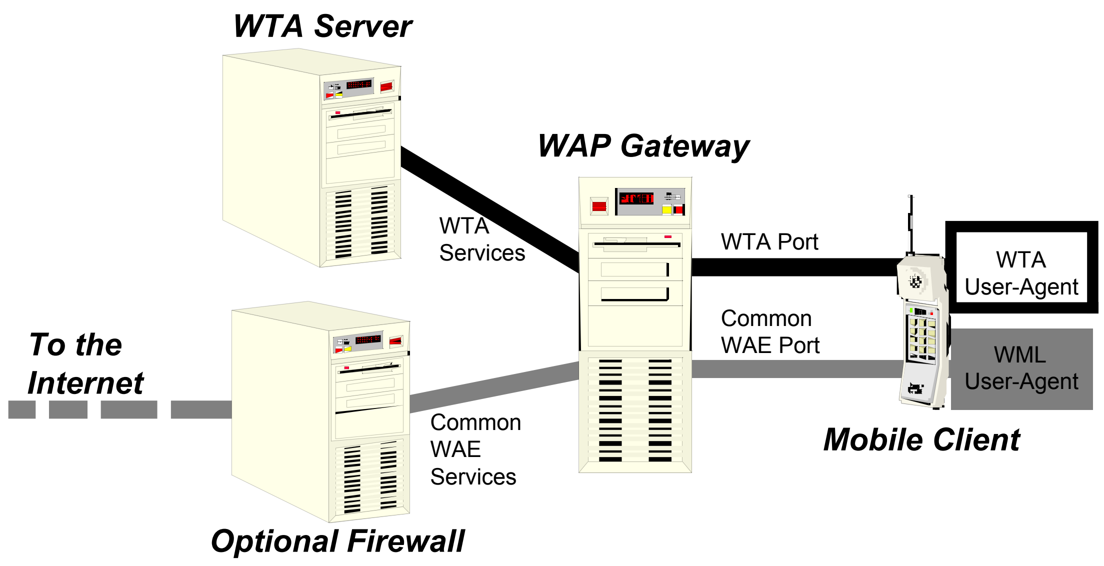


Figure 4.1: WTA access control.

The WTA server might be an ordinary web server, which could be connected to e.g. a voice mail system, that provides different services. The functionality of WTA includes:

- **Wireless Telephony Application Interface, WTAI**
  This is an interface for telephony related functions that can be invoked by WML and/or WMLScript. Some examples of such functions are call-management, handling of text messages and phonebook control. The functions are divided into three categories. First there is the network common functions that are available on all types of networks. Then it is the network specific functions are unique to a certain network type. Finally there are the public functions that can be invoked from the WML user-agent. Currently there is only one public function available and that is the ability to set up a phone call. To avoid surprises on the phone bill this function must be acknowledged by the user.

- **Repository**
  The repository stores WTA services persistently to allow access without any use of the network. This is useful for real-time handling when no delays are allowed.

- **Event handling**
  Events on the network could be incoming calls, call disconnect, call answered etc. and the WTA must handle these events to provide telephony services. As a result of an event some service in the repository can be activated or some events can be bound to some action in the WML.

- **WTA service indication**
  This is a notification using the push functions in WAP. It can be used to notify a user of voice mail etc. The service indication function sends a message and a URL to the device and the user has the ability to start some service with the use of this information.

5 The protocol stack

5.1 General

Four protocols, besides the application environment (WAE), forms the actual WAP protocol stack used for communication between a client and a WAP gateway. In Figure 5.1 the protocol stack is visualized with a comparison to the HTTP protocol stack used on the World Wide Web.

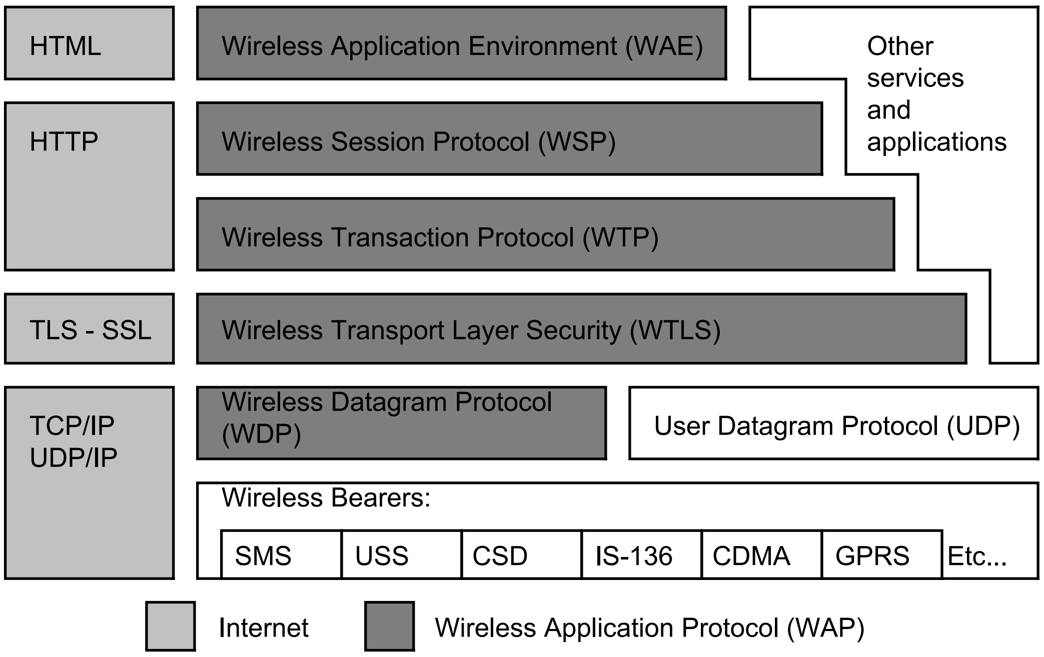

HTML
Wireless Application Environment (WAE)
Other services and applications
HTTP
Wireless Session Protocol (WSP)
Wireless Transaction Protocol (WTP)
TLS - SSL
Wireless Transport Layer Security (WTLS)
TCP/IP
UDP/IP
Wireless Datagram Protocol (WDP)
User Datagram Protocol (UDP)
Wireless Bearers:
SMS
USS
CSD
IS-136
CDMA
GPRS
Etc...
Internet
Wireless Application Protocol (WAP)

Figure 5.1: The WAP protocol stack.

The protocols can be used in four different configurations [1]:

- **Connectionless mode**
  In this mode the session protocol (WSP) runs directly on top of the datagram protocol (WDP) and offers an unreliable datagram service with no acknowledgements or resends.
- **Connectionless mode with security**
  This is the same service as the one above with the addition of the security layer (WTLS) to provide authentication, encryption etc.
- **Connection mode**
  This is the configuration where all protocol layers are involved in the communications. The session protocol (WSP) is in a mode to handle long-lived sessions, the transmission protocol (WTP) provides reliable connections with acknowledgements and, if necessary, retransmissions and the datagram protocol (WDP) provides a datagram service if the bearer doesn't support it.
- **Connection mode with security**
  Just as in the connectionless case this mode is the same as the connection mode but with the addition of security services provided by the WTLS layer.

5.2 Session layer (Wireless Session Protocol, WSP)

The session layer offers two interfaces for the WAE. A connection-oriented service that operates above the transaction layer protocol and a connectionless service that operates above a secure or nonsecure datagram service. The connection-oriented service provides a session between the client and a WAP gateway or proxy. It handles capability negotiation and communication interrupts, such as change of bearer. There is support for asynchronous requests and answers can be handled unordered. The connectionless service is basically a thin layer used by the WAE when there is no need for a reliable transaction of data. WSP is optimized for low-bandwidth bearer networks.

The wireless session protocols currently consist of services suited for browsing applications named WSP/B. WSP/B is designed to allow a WAP proxy to connect a WAP client to a standard HTTP server and provides HTTP/1.1 functionality and semantics in a compact over-the-air encoding. The service is assumed to be a long-lived session that can be suspended and resumed at a later time without any need for a new capability negotiation, which results in less traffic. With the release of WAP/1.2 a function for both reliable and unreliable data push is fully included in the protocol [6].

5.3 Transaction layer (Wireless Transaction Protocol, WTP)

WTP runs on top of a datagram service and provides a lightweight transaction-oriented protocol that is suitable for mobile phones. It operates over secure or nonsecure wireless datagram networks and is a reliable way of communication with the ability to retransmit lost messages and avoid duplication. The communication sequence is only alive during the exchange of an individual message and therefore there is no relation between different messages.

There are three classes of transactions:

- **Unreliable one-way requests**
  A message is sent and nothing is returned. This can be useful e.g. for a push service but it is not intended for applications that use a datagram service as their primary way of communication. Such applications should use WDP.
- **Reliable one-way requests**
  A message is sent and the recipient sends an acknowledgement. This is a reliable datagram service that can be used for, e.g. a reliable push service.
- **Reliable two-way request-reply transactions**
  A message is sent and the recipient replies with exactly one result message. The initiator then finally acknowledges the result message. If the recipient knows that the message processing time will exceed the initiators timer interval the recipient may send a "hold on" message to prevent the initiator to resend the original message. In this way the initiator is prevented from unnecessary retransmissions.

5.4 Security layer (Wireless Transport Layer Security, WTLS)

WTLS provides a transport layer security between the WAP client and the WAP Gateway/Proxy. It is based upon the industry standard Transport Layer Security (TLS) protocol, formerly known as Secure Socket Layer (SSL). WTLS is optimized for use over narrowband communication channels and each application can selectively enable security features.

The security layer provides data integrity to ensure that the data is unchanged and uncorrupted. It grants privacy and ensures that any intermediate parties intercepting the data stream cannot understand the data. Authentication and a protection against Denial-of-service attacks are also parts of WTLS.

5.5 Transport layer (Wireless Datagram Protocol, WDP)

WDP provides a datagram layer for different types of bearers without User Datagram Protocol (UDP) support, such as GSM SMS. This gives the upper layers a common interface to different types of wireless networks. If the bearer supports UDP the WDP layer is not needed because WAP is then given a datagram service anyway.

5.6 Bearers

The WAP protocols are designed to operate on different bearer services, such as short message, circuit-switched data and packed data. The protocols compensate for the varying levels of service. On the existing GSM networks there are two possible bearers:

5.6.1 Circuit Switched Data, CSD

This is a traditional "dial up" connection where the user connects to a WAP Gateway and stays online while browsing the pages on Internet. Since GSM is designed for voice traffic there are several drawbacks with using CSD. First of all it is the slow bearer; the best-case scenario is a connection of 14400 bps but 9600 bps is a far more common data speed. The connection lacks all sorts of immediacy too. To connect to a WAP Gateway the user has to wait for at least 10 seconds but connection times of 30 seconds is common [7].

5.6.2 Short Message Service, SMS

SMS is a text message service on GSM networks. It is useful to notify the user of things like the arrival of new voice mail or to send short notes between different users on the network. The size of the messages is limited to 160 characters, which limits the use of SMS for WAP services dramatically. Even if the transaction is small it can be both time consuming and expensive since the network operators charge the user per sent message [7][15].

5.7 Future bearers

5.7.1 General Packet Radio Service, GPRS

GPRS is a packet based communication service that is based on GSM. It will be available from the year 2000 and promises a connection speed of 56 - 114 kbps. The technique takes advantage of the unused time slots on GSM channels to transmit data. In one GSM channel there are eight time slots for voice traffic, i.e. there can be eight voice calls at the same time on the channel. When some of those slots are free those can be used for transmission of packets. The great advantage with GPRS is that the user is always on-line and has a device that is ready to start to send and receive packets at any time. In this way the time to dial a number to get a connection is eliminated but on the other hand the network operators cannot charge the user per minute but instead they will need to charge per byte. The drawback with the technique is that voice traffic is prioritized so the data speed will be very limited if there are only a few free time slots available [7][10][15].

5.7.2 Universal Mobile Telecommunications System, UMTS

The third generation of mobile phone communication standard will be UMTS. It is a packet based broadband system with a possibility to transfer data at rates of 2 megabits per second. The high speed of the data communication is possible by using a bandwidth that is 25 times bigger than the one on GSM. UMTS is based on the GSM communication standard and is planned to be up and running in the year 2002. By using packets to communicate UMTS gives the user the ability to be on-line all the time in the same way as on GPRS [7][10][15].

5.7.3 Conclusions

For WAP these new bearers will mean a more flexible and practical use of the WAP devices. The user will be able to download information much faster and will therefore get faster access to the services.

5.8 Stack configuration

In the practical use there are four different combinations of the protocol layers. The simplest way is to use the session layer right on top of a datagram protocol, i.e. WDP or UDP. This is the combination that normally is called the connection-less mode. Sometimes, though, it is desirable to achieve a more reliable way of communication. By including the transaction protocol a reliable communication channel is achieved which is called connection-oriented mode. This protocol is added between the session and the datagram layer and handles acknowledgements, message resending etc. Both of the mentioned combinations can be combined with the security layer on top of the datagram protocol, to enhance the traffic privacy or to be able to use certificates.

Since different devices can use different stack configurations they can also generate different types of requests. To be able to separate those configurations at the server, the requests are sent to different ports. Since there are four different combinations of protocol layers there are also four different ports a device can use to make a request from a server. The relationship between the stack configurations and the port numbers are shown in Table 5.1.

Table 5.1: Stack configuration and the corresponding ports.

WAP stack layers	Port
WSP WDP	9200
WSP WTP WDP	9201
WSP WTLS WDP	9202
WSP WTP WTLS WDP	9203

If there is a firewall between the WAP gateway and the client it can be necessary to open up the above mentioned ports in the firewall. An example could be a client that is connected to a modem, which resides within a Local Area Network (LAN). The gateway on the other hand is accessible somewhere on the Internet. Between the LAN and the Internet there is usually a firewall and in some cases there are just a few ports open for communication, e.g. the HTTP and POP port. Therefore it is important that none of the ports used by the client's WAP device is closed for traffic, or the user will not be able to use the device at all.

6 Advantages with WAP

Even if the new bearers mentioned above will give the client a connection speed similar to ordinary telephone modems there are still some major advantages with WAP in front of ordinary WWW communication with the TCP/IP and HTTP suites. The list below is some of the functionalities that reduce the workload and the power consumption for the client. It will give the user more operating time as well as a cheaper device, since it does not need as much computing power.

- All information, including the HTTP headers, is binary encoded by the WAP gateway. The amount of data to deliver between the client and the gateway is therefore significantly reduced in contrast to the plain text used by the HTTP protocol. The encoding also saves power on the client device since the content is easier to parse.
- Sessions can be suspended and resumed without the overhead of initial establishment. This is useful, besides saving power, to free up network resources.
- The number of packages needed by the transaction protocol is reduced, since there is only one route between the gateway and the client. Therefore the need to manage unordered packages does not exist.
- The gateway handles all the DNS services to resolve domain names used in the URLs. This means that no extra packages for name translation have to be sent over the wireless domain. However, this is not a unique advantage of WAP since it can be achieved with a HTTP proxy as well.
- From version 1.2 of the WAP protocol push functionality will be available. This means that a content provider can push information to the user whenever it is appropriate, e.g. to inform the user of changes or events.

As seen in Table 6.1 the improvements made to the protocol stack lead to significant savings in bandwidth. Here is a query from a HTTP 1.0 compatible browser compared to a query from a WAP browser. With a typical handset session with three requests and three responses less than half the number of packages is needed by the WAP protocol stack, which leads to the fact that while the HTTP 1.0 stack have 65% overhead the WAP stack only needs 14% overhead [18].

Table 6.1: Comparison between WWW and WAP bandwidth requirement.

	HTTP/TCP/IP		**WSP/WTP/UDP**
1	⇒ TCP SYN	1	⇒ **Data Request**
2	⇐ TCP SYN, ACK of SYN	2	⇐ ACK, **Reply**
3	⇒ ACK of SYN, **Data Request**	3	⇒ ACK, **Data Request**
4	⇐ ACK of Data	4	⇐ ACK, **Reply**
5	⇒ **Reply**	5	⇒ ACK, **Data Request**
6	⇐ ACK of Reply	6	⇐ ACK, **Reply**
7	⇒ **Data Request**	7	⇒ ACK
8	⇐ ACK of Data		
9	⇒ **Reply**		
10	⇐ ACK of Reply		
11	⇒ **Data Request**		
12	⇐ ACK of Data		
13	⇒ **Reply**		
14	⇐ ACK of Reply		
15	⇒ TCP FIN		
16	⇐ TCP FIN, ACK of FIN		
17	⇒ ACK of FIN		

Bold packets contain payload
Non-bold items are overhead

Furthermore the content written in WML is designed for a device with a smaller screen and a more limited input device than an ordinary computer. Since traditional web pages written in

HTML are designed for desktop computers those are not practical for small handheld devices, such as pocket computers and mobile phones. This kind of devices is called Personal Digital Assistants, or PDAs.

7 Alternatives to WAP

For the European GSM market there are no real alternatives to WAP to get mobile Internet access. The only option is the ordinary dial-up connections with a mobile phone and a laptop computer or a PDA such as PalmPilot or Cassiopeia. The drawbacks with the classical computer-and-phone concept are the clumsiness of the computer and if you are using a PDA you still have to carry two devices. A common thing for these two alternatives is the higher overhead an ordinary HTTP and TCP/IP connection results in which leads to higher power consumption etc.

Casio's PDA Cassiopeia will in a future model include a GSM telephone and will in that way eliminate the need for two devices. Casio will however not support the WAP protocol in this PDA. Instead they believe in faster mobile networks and wireless communications with traditional WEB services [9], which ought-to lead to the same problems as discussed above.

In Japan the mobile phones are operating on a network standard called PDC. This network supports packet-based communication services and has the ability to charge the user per amount of data instead of minutes. An operator, by the name Docomo, has released a very popular service called I-mode that has more than 7000 different services provided by 356 companies in Mars 2000. Many phones have color displays and future models will be able to run small Java applications that can be downloaded over the network [4][5].

8 Security

8.1 Security in general

Internet has become a more and more important part of people's lives. It is no longer only used as an information resource or an electronic mail system but also an important place to shop or manage bank and stock transactions. This leads to the need for a highly secure network where credit card numbers and bank accounts are kept secret from anyone else but the user and the service provider. Exactly the same need for security applies to the mobile use of the Internet but it has to take a slightly different approach. As mentioned before the wireless devices usually do not have the same bandwidth and computing power as the wired ones. This is therefore something the security implementation must take into consideration.

There are four different aspects a security system can address [14]. Those are:

- **Privacy**
  All messages sent between two parties must be protected from any intermediating accesses. This means that no one else except the sender and the receiver shall be able to see, access or use the information sent. The information can be addresses, credit card numbers, phone numbers or any other kind of sensitive data.
- **Integrity**
  Any changes to the content must be detected by the receiver to prevent that any other party than the sender give the receiver information. If the receiver finds out that a message has been altered between herself and the sender it often results in a resend request from the receiver.
- **Authentication**
  Ensures that the corresponding party is who it claims to be. In the real world this can be compared to the need for a drivers license or an ID. If a person for instance wants to withdraw money from a bank account the bank cashier require to see some proper identification to assure it is the owner of the account that does the withdrawal.
- **Non-repudiation**
  Guarantees that a party cannot claim it never participated in a transaction. This can be compared to a signature in the real world, e.g. if a person has signed a check the signature proves that the person and no one else approved the transaction.

There are some different techniques used to implement those security aspects and the techniques used by the security protocols will be described below.

8.1.1 Public key cryptography

This technique is based on pairs of keys and algorithms. Each individual has one private and one public key and there is one algorithm for encryption and one for decryption. To send an encrypted message to a person one uses that person's public key and the encryption algorithm. That message can then only be read if it is decrypted with the receiver's private key and the decryption algorithm. In this way the messages are hidden to all other users but the receiver. It is also possible to encrypt a message with one's own private key. This message is then only readable after decryption with the sender's public key. In this way everyone reading the message knows that it really came from the owner of the public key and not from anyone else.

8.1.2 Bulk encryption algorithms

Since public key cryptography uses very advanced algorithms to encrypt small amounts of data it is unpractical to use with larger quantities of information. A better alternative can be faster bulk encryption algorithms that use a shared secret key to exchange the data between the parties.

These algorithms are extremely difficult to decode when the shared key contains a large number of bits and this method is therefore used to encrypt most of the secure messages on the Internet.

8.1.3 Hashing algorithms

By using hashing algorithms the integrity can be provided since they do a small mathematical fingerprint of a message. If any content in the message has been altered the fingerprint will no longer match and the receiver must ask the sender to retransmit the message.

8.1.4 Digital certificates

Anyone can put up a web server, generate key pairs and falsely give oneself out as being another web site. To prevent this from happening there are digital certificates used to provide an authenticated way of distributing public and private keys. The certificates can also be used to authenticate a party so clients and servers know whom they are really communicating with.

There are two types of digital certificates: server certificates and client certificates. As the names indicate the server certificate is used to authenticate a web server and a client certificate is used to authenticate an individual user on the Internet. The certificate holds information such as the party's identity and public key. It is also encrypted with the private key of a certificate authority to verify the certificate's authenticity. Companies like VeriSign and RSA Security operate as certificate authorities and is providing a respected and independent third part resource to issue keys and certificates to their holders.

8.1.5 Digital signatures

An authorization from a user can be a message encrypted with the user's private key from the client certificate. This ensures that no one else but the specific user has sent the authorization and it can therefore be compared to a written signature in the real world.

8.2 Secure Socket Layer (SSL)

The best way to understand the security features of WAP is to look at how security is implemented on the Internet, which often uses the Secure Socket Layer (SSL) protocol. The security model used by WAP is not only based on SSL but it also uses it as an intimate part for its communication since the WAP communication model is highly integrated with the World Wide Web. Over the Internet the most common way to provide security is with the combination of SSL, digital certificates and either username and password pairs or digital signatures.

To encrypt the traffic between different parties SSL uses the public key cryptography to exchange the shared keys necessary to use the bulk algorithm. The public key cryptography is a secure and reliable way of exchanging keys during the handshake process. The bulk algorithm is used for the rest of the communication since it provides both fast and secure transmitting of large quantities of data. In this way it is ensured that the shared secret keys are kept secret during the whole conversation. The technique with encryption algorithms is used to provide the privacy aspect to the communication while hashing algorithms and digital certificates are used to grant the integrity.

When a web browser requests a secure document it first downloads a server certificate from the web server. This certificate is then validated and decrypted with the certificate authority's public key in order to get the server's public key. With this public key the browser then encrypts and sends a shared secret key to the server to be used for the rest of the communication. By following this procedure the browser and the server can rely on a secure conversation that is both private, authenticated and provides complete integrity. To achieve non-repudiation, most of the services rely on the username and password pairs used by the client to log in, but to authorize a transaction a digital signature can be requested by the server.

8.3 Wireless Transport Layer Security (WTLS)

While SSL is designed for high bandwidth communication channels with low latency and access to high processing capabilities it is by its nature unsuitable for the principles of data traffic with wireless handheld devices. If the SSL protocol were to be implemented in the WAP security model it would lead to the need of powerful and expensive devices and therefore violating the goals of WAP Forum. Instead there is a protocol developed called Wireless Transport Layer Security (WTLS). This protocol is based on the Internet standard security protocol Transport Layer Security (TLS) 1.0, which in turn is based on SSL 3.0. This provides strong Internet security over the wireless communication domain. WTLS minimizes the protocol overhead and enables more compression than SSL. Compared to TLS it also adds functionality such as datagram support, optimized handshake and dynamic refreshing. WTLS is fully optional in a WAP session and operates above the datagram layer of the WAP stack. It can be used in a connection-oriented mode as well as in a connection-less mode.

Just like its ancestors WTLS provides guaranteed privacy and prevents manipulation by any third party. It is not required to use client authentication or the non-repudiation mechanisms but it is easily implemented by using standard web development practice using usernames and passwords. If so the sender can be identified and a party cannot falsely deny having sent its messages. Just like SSL, the confidentiality is maintained using encryption and the authentication and non-repudiation functionalities by using digital certificates.

A secure connection is set up during an establishment phase where the parties are negotiating about parameter settings, key exchange and authentication. Both parties can abort the connection at any time, during the establishment phase as well as later on.

8.4 WAP security model

Since WAP is using existing technologies as much as possible there is a lot of communication over the World Wide Web domain and not just only over the wireless community. This leads to the need of two different security protocols, SSL for the web and WTLS for the wireless part. The WAP gateway becomes the link between those two parts as shown in Figure 8.1.

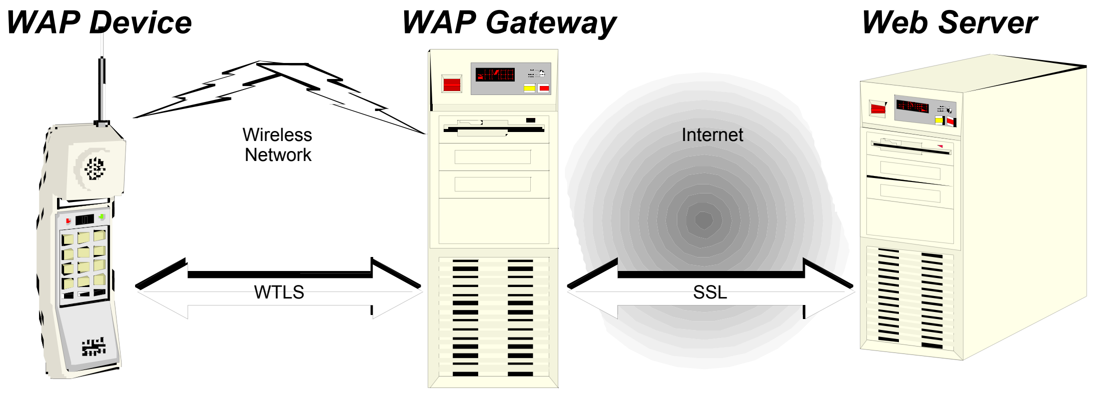


Figure 8.1: The WAP security model

Since all traffic must be decoded and re-encoded in the gateway there are some strict rules for the gateway to follow. First of all it is not allowed to store any decrypted information on secondary media. The whole conversion process has to occur in volatile memory and all information must be deleted as soon as the conversion is finished. The only access to the gateway that can be allowed is authenticated logins by an administrator within the gateways domain. This is to ensure the users and service providers that the information will still be secure and not fall into any other party's hands despite the conversion process.

8.5 Drawbacks with WTLS

The security protocols are frameworks of different configurations and algorithms. In the implementations of SSL there are a number of limited combinations of algorithms that can be used but there are no such limitations in WTLS. According to Magnus Nyström at RSA Laboratories, who is specialized on wireless security, this lack of limitations opens the system for attacks [2]. If the wrong combination of the algorithms is used, a third party can both listen to and change the contents of the transferred data without any possibilities for the communicating parties to notice it. There are warnings for the combinations that are not suitable in the WAP specification but Magnus Nyström thinks it would be better if there were limitations in which combinations that can be used. Instead, it is now up to the software developers to keep track of the dangerous combinations.

8.6 Computer viruses and WAP

Today there are small possibilities to write any severe computer viruses for WAP enabled devices. The functionalities in WAP are still quite limited and there are no uniform operating systems between different manufacturers. But in the future there might be other features in the WAP specification and if there will be similar and more powerful operating systems on the devices there is also an increasing risk of virus attacks. One possible attack could be a virus that copies itself to all contacts in the client's phonebook like the recent Melissa and Love-Letter mail viruses.

The only antivirus software manufacturer that have developed any program for WAP virus attacks up until today is the Finnish based F-Secure. According to Risto Siilasmaa on F-Secure there is a theoretical possibility to send damaging SMS code today but in the future with WAP 1.2 it will be easier to write malicious code. Later on, when there is a version 2.0, there might be an open field for all kinds of viruses [3].

It is probably the service providers that have to take care of the protection against viruses, e.g. in the WAP gateways. This is because if the clients should need to download antivirus software to their devices the whole idea with cheap PDAs or mobile phones with small memory would fall. Therefore it is more likely that virus protection will be a service rather than a product developed by the anti-virus corporations. The mobile network operators or the service providers will in turn offer this service to the customers.

9 WML

WML is a markup language intended to specify content and user interface for narrowband devices [17]. It is based on XML and is similar to HTML but with some major differences. Among other things it is specified for small devices with limited user input facilities as well as limited memory and computational resources.

WML is built up around four major functionalities:

- **Text presentation and layout**
  WML includes text and image support. This includes a variety of formatting and layout commands, such as boldface text and alignments.
- **Deck/card organization metaphor**
  All information in a WML document is organized in cards and decks. A card contains one or more units of user interaction, e.g. an input field or a paragraph of text. A user navigates through a series of cards and make choices, reads the content or moves on to another card. A deck is a collection of cards. It is the deck that can be compared to a HTML page in that an URL identifies it and it is the unit of a transmission.
- **Intercard navigation and linking**
  WML includes support for linking between cards and decks. It also provides event handling in the device that can be used for navigational purposes as well as to execute scripts.
- **String parameterization and state management**
  All WML decks can be parameterized using a state model. Variables are substituted at runtime and can be used in place of strings, which allows a more efficient use of network resources.

9.1 XML inheritance

Since WML is based on XML it inherits all the syntactic rules of XML and since all WML documents must include a Document Type Definition (DTD) they are valid XML documents (see Appendix A: for further details on XML). Here follows some major differences the XML inheritance brings in contrast to HTML.

9.1.1 Unicode character set

XML is intended for worldwide use and it is therefore not limited to the use of seven or eight bit character sets. Instead it is specified using the 16-bit Unicode character set which contain most of the worlds different characters.

9.1.2 Character entities

Because of the use of the Unicode character set a reference to a character always refer to the same logical character with respect to the document character set (Unicode) and not to the document encoding (charset). WML supports three different character entity formats:

- Named character entities, such as `&amp;` and `&lt;`
- Decimal numeric character entries, such as `&#32;`
- Hexadecimal numeric character entities, such as `&#x20;`

The correct way to write non US-ASCII characters in an XML document is by using the decimal or hexadecimal character entities with the corresponding Unicode character value. As an example it can be mentioned that the Swedish characters Å, Ä and Ö, which in HTML are written with the entities `&Aring;`, `&Auml;` and `&Ouml;`, need to be written with their corresponding Unicode value in WML. The particular named entities in WML are:

- Quotation mark: `&quot;`
- Ampersand: `&amp;`
- Apostrophe: `&apos;`
- Less than: `&lt;`
- Greater than; `&gt;`
- Non-breaking space: `&nbsp;`
- Soft hyphen: `&shy;`

9.1.3 Case sensitivity

XML, in contrast to HTML, is completely case-sensitive. That means that a tag `<CARD>` is a completely different tag than `<card>`.

9.1.4 Document header

All XML documents must begin with a header tag that indicates it is an XML document. Since WML is not only an ordinary XML document but also a valid one there has to be a DTD reference on the second line of the document. WML documents shall have a reference to the DTD even if valid XML documents in general can have the whole DTD included in the document contents. Because of these two demands, a proper WML document must begin with the following lines:

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

Of course there will be a difference in the header between different versions of the WAP specifications.

9.1.5 Elements and tags

An element is the core structure of an XML document and consequently it is the core structure for a WML document as well. Tags delimit the elements of an XML document and a tag consists of the element name enclosed with a pair of angular brackets ("<" and ">"). To separate the end tags from the start tags the element name begins with a slash ("/") if it is an end tag, e.g. the card element looks like this: `<card>…</card>`. Empty elements, such as the `line break` element, are written without an end tag but with a slash at the end of the start tag, e.g. `<br/>`.

9.1.6 Hierarchical structure

Since WML is a valid XML document it must have a DTD included. The DTD contains the proper internal state of all the elements the WML document can contain and it is important that the programmer follows the rules of the DTD. Therefore it is necessary to examine which elements can contain which before writing the code.

9.1.7 Proper nesting of elements

The elements of an XML document need to be properly nested. It is not allowed to overlap elements in XML, i.e. to write the end tags in a different order than reversed from the start tags. As an example it is not allowed to write `<card>…<do>…</card>…</do>` since the card element must contain the do element as a whole. The elements in XML can be illustrated as boxes. To achieve a proper XML documents the boxes need to be closed. A closed box can either contain another box or it can stand alongside other boxes, but it can never contain a half box.

9.2 WML elements

Here is a brief description of some of the more important elements in WML. To get a more detailed description see [12] or [17].

- **`wml` element**
  A WML document must contain one deck. This element should contain all the contents, i.e. all the other elements, of the document. A deck is limited to a maximum of 1400 bytes. This means that the document needs to be very code efficient. One way to achieve that is to use templates and variables. It is also good to try to keep as much of the information as possible on the server instead of the client's device.

- **`template` element**
  This element defines template event bindings for cards. The event bindings declared in this section applies to all the cards in the deck.

- **`card` element**
  The card element delimits which of the deck's contents will be visualized at the same time. The element can contain paragraphs as well as events. When a deck is loaded the first card of the deck is presented to the user. To access a particular card within a deck a hash mark (#) is included in the URL to separate the deck from the card, e.g. *http://wap.domain.com#card2*. Even relative URLs are allowed within a deck, e.g. *#card2*.

- **`head` element**
  The `head` element specifies information related to the deck as a whole. This information resides in sub elements of `head`, which can contain meta-data and access control structures.

- **`onevent` element**
  There can be different events in a WML document and there is the `onevent` element to catch those events. The arguments specify what kind of event to handle and the content of the element state the event action. Examples of events to handle are timers, forward- or backward entrance to cards etc.

- **`p` element**
  This element delimits a paragraph within a card. With the arguments, the programmer can control alignment as well as whether or not the browser should wrap the text.

- **`timer` element**
  The `timer` element sets up, as the name implies, a timer. The timer generates a timer event after an interval set by its attributes. The timer event can be caught by an `onevent` element and can for instance be useful to display a logotype image for a couple of seconds before a menu is displayed to the user.

- **`input` element**
  An input element can be compared to an edit box on a web page or in a dialogue box in the graphical user interface (GUI) of an operating system (OS). In WML the input element is used to input strings of text. The format of the strings can be specified, for instance if the entered string must contain alphanumeric numbers or a certain number of characters.

- **`table` element**
  The table element specifies a table with rows and data fields quite similar to the corresponding element in HTML.

- **`select` and `optgroup` elements**
  Together these elements build a list where the user either gives the possibility to select one list item, like radio buttons in an OS GUI, or multiple list items, like checkboxes.

- **`do` element**
  The `do` element provides a general mechanism for the user to perform actions on the card. The action can be e.g. to move to the previous card or to go to another destination as well as set some variables in the devices memory.

- **`anchor` and `a` elements**
  Both the `a` and the `anchor` elements defines hard links. The difference is, that while the `a` element only redirects the browser to a new location, the `anchor` element can perform actions similar to the `do` element.

- **`go`, `prev` and `postfield` elements**
  The `prev` element moves the browser back to the last location in the history list of the device while the `go` element navigates the browser to a new location. The `go` element supports the HTTP parameter passing methods GET and POST and those parameters can be set with the `postfield` element.

- **`setvar` element**
  This element is used to set variable values in the devices memory. It can be used under the `go` or the `prev` element while the user is leaving the card, e.g. to save some data the user entered in an input box. It can also be used within a `do` or `anchor` element through the `refresh` element to set variables by user actions. The variables can be substituted into formatted text in a WML document by using the dollar sign ($) operator, e.g. $(var). Because of the special function of the dollar sign there has to be two dollar signs in a row to write one in the document text, i.e. "$$". The variable names do not follow the same rules of characters that WML documents do. The names are case sensitive but instead of using the Unicode character set they must begin with a US-ASCII letter or underscore followed by zero or more letters, digits or underscores.

- **`img` element**
  Just like in HTML this element inserts an image into the document. Since WAP only supports it's own bitmap format the link should point to a wbmp-file. Wbmp is a one bit bitmap file format and it is used because more advanced graphics would demand higher bandwidth to be downloaded.

- **Style elements**
  There are several elements to format the output text. Examples of text formats available are bold, italic, underlined, big and small.

Since WML is a valid XML document there are requirements on how the elements are ordered in the hierarchal aspects. Which elements can be sub elements of others are defined in the documents type definition, i.e. the DTD, and these definitions must be followed. If the definitions are violated the byte compilation performed in the WAP gateway will fail and in that case there will be no page presented to the user. As a tool for the programmer a hierarchical view of the WML elements are presented in Figure 9.1. The elements visualized in the figure are all available elements in WML 1.1, i.e. the WML specification in WAP version 1.1. Please note that some of the relationships are required and some of them are limited to one per ancestor element. For further information it is recommended to read [12].

One problem with designing WML pages is the unlikeness between how different browsers render the WML code. Not all browsers support all tags in the WML code and the tags that two different browsers support doesn't have to appear in the same way when a user looks at the page. For instance the available WAP enabled mobile phones from Nokia and Ericsson on the market today have some major different approaches on how to interpret the WML code. While the Nokia 7110 always puts a line-break after a link or an input field the Ericsson R320 requires a line break tag to achieve the same look.

So if the programmer wishes the page to look similar on the devices there are two alternatives. There can be different pages for different devices, e.g. by writing a server side script which produces different code depending on what device the client have. Otherwise some workaround has to be done in one way or another. Since there are more than two manufacturers of mobile phones and other kinds of PDAs, the simplest solution might be to write good WML code and let the devices interpret it as they wish. Hopefully the functionality will be similar despite the rendering disagreement.

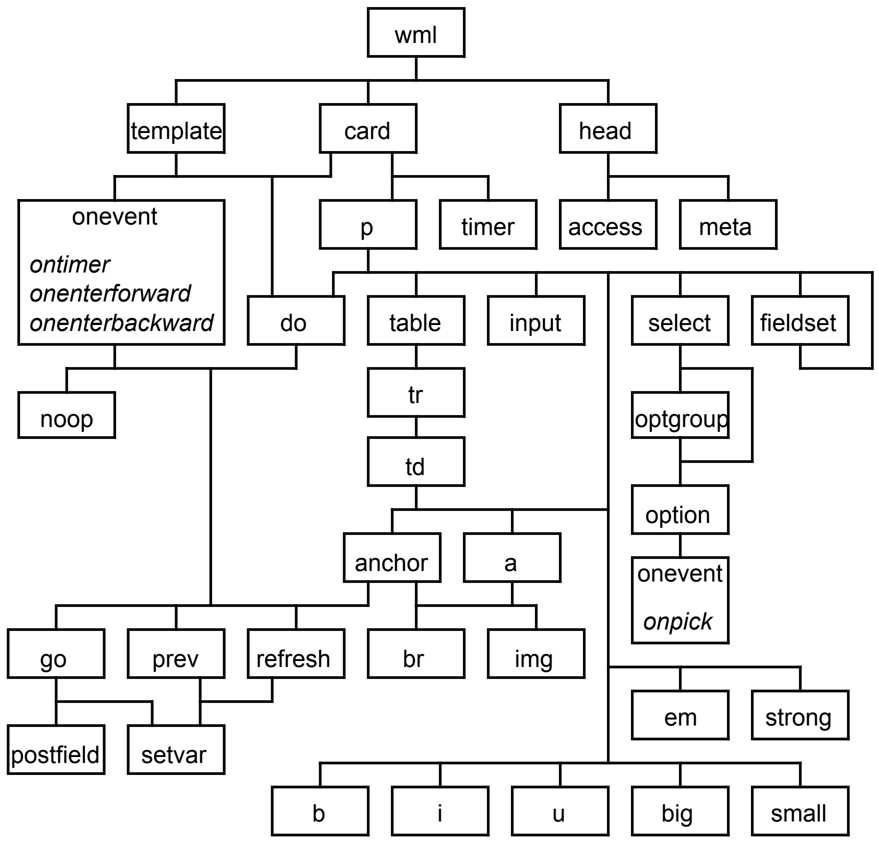


Figure 9.1: The element hierarchy of WML.

10 WMLScript

To enhance the functionalities of the WML pages there is a script language available called WMLScript. It can among other things be used to write procedures that minimize the bandwidth requirement and time-consuming server requests, e.g. by managing input errors in the device instead of at the server. Even if WMLScript is often used to provide WML documents with some sort of intelligence there is no need to write a WML document to use WMLScript since it can be used as a stand-alone tool as well. Here are some features that WML doesn't support but which can be available by using WMLScript:

- Checking validity of user input, as mentioned above.
- Accessing facilities of the user agent. On a mobile phone such facilities can be to make a phone call, send a message or to save a phone number in the address book.
- Generating dialogs and messages locally. In this way error messages, confirmations etc. can be visible to the user faster than if it has to be downloaded from a server.
- Allowing extensions to the user agent software and configuring a user agent after it has been deployed.

WMLScript is based on ECMAScript but it has been optimized for low bandwidth devices. Just like the WML documents the WMLScript code is transferred in a compiled byte code over the wireless network. This saves radio bandwidth as well as memory and processing capacity on the client's device. To get the language smaller and easier to compile into byte code as well as easier to learn many of the advanced features of ECMAScript are dropped [13].

10.1 Character set

Even though WML uses the Unicode character set, WMLScript does not. Instead the WMLScript interpreter must use only one character set, which is the native character set, for all its string operations. Transcoding between different character sets are allowed as long as the operations are performed in the native character set. In the Lang library there are functions available for character set operations.

10.2 Accessing WMLScript

A WMLScript is identified on the Internet by using URLs. It is retrieved the same way as WML- or HTML documents, i.e. over standard protocols using HTTP semantics, such as WSP. There are examples of WMLScript available that use URLs that contain white spaces. This is not recommended since there are browsers on the market that do not accept spaces in URLs and it is therefore better to avoid such URLs. For instance, the mobile phones with WAP functionality that are released from Ericsson cannot interpret URLs that contains white spaces.

In a WMLScript file there are one or more functions declared. Those functions that are accessible from outside the file are marked with the word extern to separate them from the internal functions of the file. To call a function within a WMLScript file one uses a URL to identify the file followed by a function name with parameters within parentheses. The URL and the function name are separated by a hash mark (#). This results in a URL that can, for instance, look like this:

```
http://wap.games.com/blackjack.wmls#play($(money),$(stake))
```

The dollar signs indicate, as this URL is written inside a WML document, that there is a substitution of variables. In the browser's memory there are two variables, one named *money* and one named *stake*, and when the WML document is presented to the user those variables are substituted with their actual values. In this way, when this URL is called, the arguments to the function *play* will be the actual amount of money and the stake the user really has in the game.

All WAP browsers shall support relative URLs. This means that, as long as the calling WML document is in the same location, the URL above could be written:

```
blackjack.wmls#play($(money),$(stake))
```

10.3 Script libraries

WMLScript is based on the use of libraries. Within every device that supports WMLScript there are a set of standard libraries, each with a collection of standardized functions. To call a function within a library, the library name and the function name are used, separated by a dot (.). If a user function shall call the library function `elementAt()`, within the library `String`, it can look like this:

```
function dummy(str) {
    var i = String.elementAt(str,3,” ”);
};
```

For an overview of the WMLScript libraries, please look at [13].

11 Development tools

11.1 Nokia WAP Toolkit 1.3beta

The Nokia WAP Toolkit contains several components for WAP service development. Firstly it is a telephone emulator with the same WAP-functions as the actual mobile phone Nokia 7110. A very nice feature is a location field where it is possible to see the actual URL even if the browser was redirected to another location. The emulator is also equipped with a history list, bookmarks and a list of variables in the devices memory and what values they are bound to. A drawback with the toolkit is the bugs. The version now available is 1.3beta, and it really feels like a beta. Sometimes the whole emulator device freezes but it can usually be restarted quite easily. Some of the freezes requires a whole new memory image and that operation requires some menu and dialog box operations. The browser also has a major bug in the WMLScript support that makes it almost impossible to read the text output.

Secondly the toolkit is a WML and WMLScript editor. The tags, attributes and values are presented in different colors that make it easy to get an overview of the code. There is also a compiler for both WML and WMLScript that makes it easy to check the written code for errors. WML is stricter than classic HTML, which means that an erroneous document will not be rendered at all. This is because of the WAP gateway, which compiles the source code to byte code, wont pass on any erroneous code to the browser.

The third feature in the Nokia WAP Toolkit is the bitmap editor. It can read gif- and jpeg formatted pictures and convert them to the WAP picture format wbmp. It is also a very simple drawing tool to paint lines and squares, just like in any other paint program.

Besides the features of the Toolkit there are a set of examples on how to write both WML and WMLScript code. Note though, that some of the examples included, violate the rules for URLs. This violation makes the examples incompatible with some browsers on the market (see Section 10.2).

11.2 Nokia WAP Server 1.0

This application is a complete WAP gateway with some additional functionality. It contains all the normal WAP gateway features, such as WML and WMLScript binary encoding and decoding and the possibility to connect via connectionless or connection oriented modes as well as HTML and text conversion to the WML document format.

Besides the standard WAP gateway features it provides complete logs and alarm capabilities. The server also has an access control scheme where the administrator can set up from which domains the traffic should be allowed or denied. The entire configuration is done from a control program that connects to the server and consequently it can be used from a remote site. The server also has an API of its own, for server side programming as well as to access third party systems.

To achieve secure connections over the WTLS protocol there is a Security Pack available to the Nokia WAP Server. The Security Pack is released in two different versions with either 56-bit or 128-bit encryption for the bulk encryption algorithms. For the asymmetric encryption 768-bit and 1024-bit encryption is used respectively.

11.3 Ericsson WapIDE SDK 2.1

Ericsson’s Service Development Kit consists of three different parts. There is a WAP browser, an application designer and a server toolset.

The first component in the SDK is the browser. It is a WAP phone emulator with practically the same WAP functions as the real mobile phone Ericsson R320. In difference to the real WAP terminal this browser doesn’t need a WAP gateway to reach the contents. Instead it can download the WML documents directly from a web server or it can display WML files locally. The user interface is exactly the same as with the real device since the emulator consists of a picture visualizing the real phone where the buttons can be maneuvered by clicking the mouse.

The application designer is actually an editor for WML and WMLScript files. The program organizes the files in projects to make them easy to maintain. When editing WMLScript files the application designer works as any ordinary text editor but when editing WML documents there is a menu available with automatic insertion of different tags. This tool is very handy since it only allows insertion of elements that are allowed at a certain place in the code, which makes it easier to write a code that will be correctly validated. Another feature to make the code writing easier is the colorization of the tags that makes a visual separation from the document text.

Even the application designer contains a R320 emulator but this emulator is not intended for any use over the Internet. Instead it is used to try out the code written in the text editor. When the code is written it can be compiled. This is a good way to check if there are any errors in the code, which as usual is hard to see while writing. When the code is free from errors it is ready to be run in the browser. Just like in the stand-alone browser emulator this is an exact copy of the look and feel of the real device.

As the last component of the SDK suite there is a so-called server toolset. The toolset contains several functions to develop both static and dynamic (se Section 12) WAP applications. Among the functions included are WML and WMLScript compilers, a WML code verifier that compares the code against the WML document type definition (DTD) and a Perl library with functions for creating WML source. It also contains a set of sample applications, both static WML pages and dynamic CGI applications.

11.4 Ericsson R380 Emulator

Ericsson R380 is a mobile phone with built-in PDA functionality. It differs from ordinary mobile phones with its large touch-screen display, as seen in Figure 11.1, and that it is running the EPOC operating system. The device has a lot of features including infrared communication, a notepad, built-in modem, e-mail and SMS management etc.

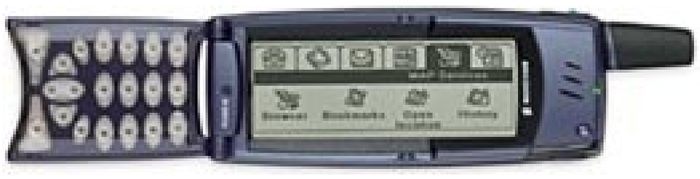


Figure 11.1: The mobile phone Ericsson R380.

The R380 emulator is quite similar to the browsers available in the previously mentioned developer toolkits from Ericsson and Nokia and just like them it has all the WAP functionality implemented in the same way as the corresponding real device. Unlike the two other emulators mentioned this is a stand-alone program with no other features but the WAP browser. The big difference, however, between this emulator and the R320 ditto is the display size. Since the device has a larger screen and since it is used lying down the proportions of the contents displayed have big differences. Therefore it can be good to check if the WML documents one has written looks and behaves well even in this device.

11.5 Ericsson Test Area

A WAP enabled mobile phone that connects through a GSM data call (ordinary telephone connection with the capability to transmit data) is connecting with the Point-to-Point Protocol (PPP). Since this is the most common way a computer connects to an Internet Service Provider (ISP) via a modem the mobile phone could consequently connect to the Internet by calling an ISP.

For this reason Ericsson has set up a free WAP gateway for test purposes on the Internet. Since not all network operators provide a WAP gateway for their customers the WAP application developers can test their applications over this gateway instead. The necessary IP address and other useful information can be found at [16].

11.6 Ericsson WAP Gateway/Proxy Demo 1.0

There is a free trial of Ericsson's WAP Gateway available for download [16]. The gateway can be used for testing the WAP technique but it is does not have the status of a real product. Since it is a demo version it has some limitations, such as a limited number of transactions per minute, but on the other hand it supports both connection less and connection oriented modes, as well as WML and WMLScript encoding and decoding, i.e. the basic building blocks of a WAP gateway.

11.7 WAPDrive Waptor 2.3

WAPDrive has a lot of development tools for WAP applications. The WML editor Waptor is an easy to use tool, free for download on WAPDrive's homepage http://www.wapdrive.net. The editor has all the normal functions a text editor usually has, including search and replace operations and the ability to manage multiple files at the same time. The extra functionality for the WML document editing is a toolbox with buttons that adds WML elements in the code and a preview field that can be helpful during the code writing.

Even if the text editor has different colors for tags, attribute and text and even if it is almost possible to write a complete WML document only by clicking the mouse it has some major drawbacks compared to the developer tools from Ericsson and Nokia. Firstly the insertion of elements in the WML code has no limitations. In the corresponding tool in Ericsson's SDK the user is only allowed to insert elements in their correct environment, and the program even asks for the parameters automatically by showing a dialog box. Secondly the preview pane lacks a lot of the functions in WML, for instance it cannot handle timers or variables.

11.8 Web servers

WAP is developed to make use of as much of existing technologies as possible and just like in the World Wide Web case the content is placed on a web server. Almost any web server on the market works perfectly as a WAP content server as long as it supports new alternative MIME types. MIME, which stands for Multi-purpose Internet Mail Extensions, is a description of the type of the content sent by the web server. Ordinary web pages written in HTML for instance are of the MIME type *text/html.* Other MIME types available describe for instance gif or jpeg picture files or just plain text file contents.

To separate the different WAP files from the ordinary documents and picture formats on the web there are some special MIME types for WAP. If the document sent by a server has the wrong content type it will not be properly interpreted or maybe not shown at all. It is therefore very important to add the MIME types shown in Table 11.1 to the web server.

11.9 Conclusions

While most of the useful WAP sites are based on dynamic information it is necessary to write some sort of server side scripts or applications as described in Section 12. For this purpose the Ericsson WapIDE SDK might be helpful to generate the script code for the use of the Common Gateway Interface (CGI). But since CGI is an old technique it might be better to reconsider the use of a more modern server side application technique as described below.

Table 11.1: WAP MIME types.

Content	MIME type	File extension
WML source	text/vnd.wap.wml	wml
Compiled WML	application/vnd.wap.wmlc	wmlc
WMLScript source	text/vnd.wap.wmlscript	wmls
Compiled WMLScript	application/vnd.wap.wmlscriptc	wmlsc
Wireless bitmaps	image/vnd.wap.wbmp	wbmp

12 Dynamic WAP applications

When a user downloads a static WAP page the page will always look the same, every time it is downloaded. In some cases this is enough, e.g. when the page is a timetable for a bus route. WMLScript can solve some of the limitations with the static content by acting on user input. For instance if a user writes some data in an input field, this can be sent to a WMLScript. Then the script will act differently depending on what data the user entered. In WMLScript it is also possible to use random values to change the course of the script, e.g. for simple games etc.

If the page a user downloads to the device is supposed to contain fluctuating data, such as stock-exchange rates, the page has to be built on the server. The page is then based upon data from a data source, e.g. a database, and built at the server after the client's request. This is also the case if the client, for example, does not want to see a timetable of one bus route but wants to know how to travel from one point to another at a certain time. In this case the user enters the time, point of departure and destination. The data is then sent as a request to the server. At the server this request is processed in some way and an answer is sent back to the client in form of a WML document. This type of document, which is built at the server and depends on either occasion or user input, is called a dynamic document or a dynamic application.

There are a lot of different techniques used to achieve dynamic content on web or WAP pages and some of them will be mentioned below. The descriptions of the techniques are mainly based on [15]. Since WAP is using the same technique as the web the descriptions below will be about web contents. This is, however, fully applicable on WAP implementations as well. Whenever a server side application can generate a HTML document it can, with some major or minor modifications, generate WML documents.

One really important thing to think about is the content type of the response document. As mentioned before WAP documents, such as WML documents, WMLScript script files, wbmp bitmap images etc., need to be of the right MIME type. Since web servers and their corresponding techniques to generate dynamic contents are usually designed to produce documents with HTML contents this is the type the documents get. Therefore it is often necessary to explicitly order the web server to put another MIME or content type on the document before it is returned to the client. If this is not done there is a big risk that the document cannot be handled correctly by the WAP gateway and consequently the information will never reach the user.

12.1 Common Gateway Interface (CGI)

On an ordinary client request from a web server, the web server analyzes the URL provided by the client and returns the appropriate document. This is the way of providing static information to a user. In the HTTP protocol there is a standard way to achieve dynamic web contents called the standard gateway interface, or shortened CGI. This is a standard way for a web server to pass a user's request to an application program. It also defines how the application returns data and how it is forwarded to the user. For instance, a user fills in a form on a web page and clicks a button to send the information to a web server. The web server, in turn, passes the information over to a small application program. When this program executes, it processes the user's data and generates a web page. The page is returned to the server and finally the server then forwards the web page back to the client. The user can then see the result of the operation in the browser.

The CGI technique is a standard way of sending data between a client, server and an application program. The programmer can be sure that the code, as long as it does not contain any system dependent operations, will work whatever operating system the server runs on. Because CGI is only an interface, a programmer can write a CGI application program in a number of programming languages. Common languages are C, C++, Java and Perl.

12.2 Active Server Pages (ASP)

An Active Server Page is a HTML (or WML in the WAP case) file with server side scripts included. The server executes those scripts when the file is requested from a client and the output of the scripts fill in the dynamic data in the HTML code. This method is considered faster and more secure than CGI since it is executed directly on the server. One security hazard that is eliminated with ASP, in contrast to CGI, is the problem with allowing another program to execute the client's request. If this request is malicious and the application program is badly written, that could lead to some major problems. Since ASP is executing within the web server the application does not have more rights on the system than the server itself.

ASP is a product of Microsoft and it is only available in the Internet Information Server (IIS), which requires a Windows machine. This platform dependent solution requires the web server to run a computer with the Windows operating system and it is not available in any UNIX environment. Though the generated files are of a standardized format any web browser on the market, independent of platform, can render the content. The scripts can be written in one of the two scripts languages VBScript or JScript and the execution can be controlled, e.g. by user input or database queries.

12.3 Java Server Pages (JSP)

In Java Server Pages there are references to small programs in the HTML code. When a page is requested the server executes those small programs to fill in the missing content in the HTML document. The technique is comparable to ASP with the big difference that in ASP the script code is included into the HTML file but in both cases it is the web server that executes the application code.

12.4 Personal Home Page (PHP)

Personal Home Page is far more known under the name PHP, the former was the name of the earliest version. PHP is similar to ASP in that it is an HTML file with the script code included but with the difference that it is platform independent. Before the page is sent to the user that requested it, the web server calls for PHP to interpret and execute the script code included in the document.

Even though PHP is platform independent it is primarily used under Linux web servers. PHP is free to use and is developed under an open-source license.

12.5 Servlet

A Servlet is a small program that is intended to run on the server. The terminology is derived from the name applet in the world of Java. While an applet is a small program that executes on the client's web browser and therefore is limited to the browser capabilities a servlet can access functionalities on the server instead. Such functionality can be database connections for instance.

A servlet can be compared to a CGI script in that the server is executing a script when the user requests a web page. The big difference between CGI and servlets is that a servlet, in the case of a Java servlet, is executed in a new thread on an already running Virtual Java Machine. Since CGI needs to start up a whole new process for each request from the clients it is a far more time consuming operation.

13 Session management

Each time a web or WAP browser requests a page on a server the server looks at the request without knowing if that client has visited the server before. If the server runs some kind of script to generate the browser contents it has no way to remember the data to next request. Sometimes the amount of information to view or collect is too big to fit in a small display of a WAP device, for instance, and then there might be a need for session management. The server, where the information resides, must then keep track of what a specific user has read or what data has been edited within that session. Sessions are also important in e.g. e-commerce sites on the Internet, on the web as well as with WAP, where the server has to know which user has put a certain product in which virtual shopping basket and so forth.

A programmer who is building web applications for Internet browsers, such as Netscape Navigator or Microsoft Internet Explorer, has a neat tool available in the cookies. A cookie is a string of text stored in a file on the client's machine and this text string is then sent to the web server on every request the client does. By sending a session ID as a cookie to the client the server will always know which client is requesting some certain information by looking at the cookie. One common method of programming web sites with session management is by using Active Server Pages (ASP) scripts. In ASP there are built in functions for, among other things, managing sessions with the use of cookies. Since it is done automatically and since the data is stored in a database managed by the server it becomes very easy for the programmer to handle sessions. Another frequent method of server side scripting is with the Common Gateway Interface (CGI), as described above. The big difference, in this case, between ASP and CGI is that in CGI scripts all session management has to be done by the programmer, both the creation of the session ID and the handling of the data to be saved. But to the programmer's help there is, just like in the ASP case, the possibility to use cookies.

The principle with cookies is that each site that wants to store information on the client's computer can do so and this leads to a rather large cookie file on the computer. Firstly such a file is not possible to store in a cheaper device with small memory size and secondly the mobile phones tend to lack any kind of secondary storage medium. Therefore, unfortunately, there is no support for cookies in WAP.

There is one technique to achieve session management without cookies that isn't that simple as the method available in ASP but not far away. When a user visits a WAP site for the first time the server, just like described above, generates some kind of unique session ID. The difference is that instead of saving it as a cookie on the client's browser it is passed on to the server with every request from the user. This can be done with the HTTP parameter passing methods GET and POST, which are supported by WAP. With the GET method the parameters are passed in the URL string. The drawback with this method is the limited length of URLs where the limit unfortunately is application dependent. To get rid of this limitation one can use the POST method instead, which sends the parameters to the server in the HTTP header.

To identify the user that owns the session, the server only needs to check the passed ID. As long as the user navigates within the same site, there are no actual drawbacks with those methods. But if the site for instance contains links to the outside world, the session will be lost if the user navigates outside the site. This leads to the problem that a site with personal data, e.g. a portal with links and user related information which can be customized by the user, will always need to ask the user to login on every visit at the site. Therefore it can be rather frustrating for the user to use the portal since there is a need to enter a username and a password just to look at the collection of links. On an ordinary website this problem can be solved with cookies. The server only needs to check if the user got a cookie that it recognizes and this cookie will then identify the user to the system. Unfortunately this is not possible in WAP due to the lack of cookies.

14 Implementation 1: Session management in ASP

14.1 Generating the WML code

ASP has a set of functions that manages sessions by writing a cookie on the client's browser. Since WAP lacks the cookie functionality there is no way that a developer can use this handy tool when programming WAP applications. Instead it is necessary to write a procedure to handle the situation. As described in a previous section this can be done by sending a unique number, called the session ID, to the client the first time it calls for a page at the site. The ID is sent to the client by writing the session ID in the WML document that is sent from the server. The client can either set the session ID in its memory for future use or it can be written in a `postfield` element so it will be sent back to the server automatically when the user clicks on a link to navigate to another location. This can be illustrated with the following WML code:

```
<anchor>
           Move on
           <go href="second.asp" method="post">
                      <postfield name="session" value="123456789"/>
           </go>
</anchor>
```

The `anchor` element specifies a link with a certain task. In this case the task is to move to another page that is specified with the `go` element. The `href` attribute specifies the URL of the destination and the `method` attribute specifies in what method the parameters of the http request will be sent. Here the method specified is POST and consequently the parameters will be sent in the HTTP header. If the method had been GET the parameters would have been sent in the URL instead.

A `go` element can include one or more `postfield` elements, which takes two arguments; the first is what name the parameter will have in the HTTP header and the second is the value of the parameter. In this case the WML document that reaches the client has a session ID that is built up by the string of numeric characters 123456789. When the user clicks on the link that says *Move on*, the session ID will be sent to the web server. The server will process the script in the file *second.asp* and provide the session ID to the script whenever it is asking for it. Of course the script that generates the code above will not contain exactly that code. If so all users would get the same session ID and there would not be much use of the session management philosophy at all. Instead the server at runtime builds pieces of the code. In ASP programming for example, the line with the `postfield` element might look like this:

```
<postfield name="session" value="<%=SessionID%>"/>
```

In ASP there is a convenient way to insert variable values in the text output that will be sent to the server. If the text between the strings <%= and %> is a proper name of a variable the string and the syntactic surrounding will be substituted with the value of the variable. Thus, if the variable *SessionID* contains the value 123456789 an execution of the script will generate the code shown in the previous example.

14.2 Writing the script code

There are a few things to think about when writing scripts to generate WAP pages that do not apply to the generation of ordinary web pages. Firstly the MIME type for a WML document is not the same as for HTML documents but instead the quite cryptic string *text/vnd.wap.wml*. A script that is generating WML code for a web server must instruct the web server to put this MIME type on the file before it is sent. The second aspect that differs WML from ordinary HTML documents is the XML specification of the file. A WML document must begin with the XML version tag on the very first line of the document. On the following line there must also be a reference to the DTD for WML documents. With all this in mind we can look a the beginning of an ASP script page:

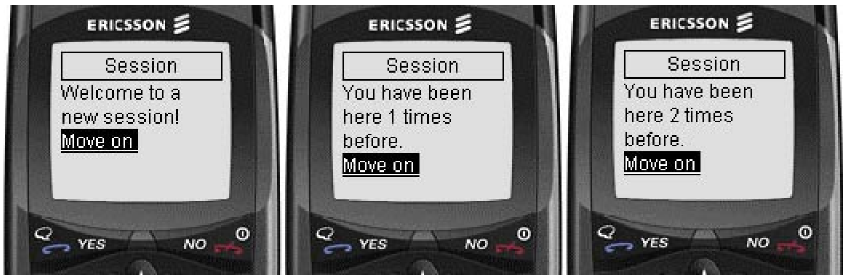
```
<%@ LANGUAGE=VBScript %>
<%
Response.ContentType = "text/vnd.wap.wml"
%><?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
```

The first line is mandatory in an ASP script and it tells the script interpreter which script language the code is written in, in this case VBScript. Everything in an ASP file is sent to the user's browser except for the script code, which is placed within the strings <% and %>. The first line of script code in this example is where the script tells the web server what kind of MIME type the document has. It is important that this line of code is stated before any output from the script, i.e. before any text that will form the WML document is written. Since the XML tag needs to be on the first line in the resulting WML code it must be on the same line as the end mark of the script code. If not, there will be a line break before the tag and the document will not be valid. This is just as important in any server side application-programming environment as in ASP.

14.3 Saving session data

If there shall be any use of the session management at all there has to be a way to save data between the user accesses. There are several ways to do that where the most common ways is to store the data in a database or to write a file on the server. ASP saves all session data in a database that makes it very easy for the developer to handle the data. The only things that have to be done are to use one command to save data and one command to retrieve data. Since WAP lacks all form of cookies and since the ASP session management is based upon the use of cookies that database cannot be used for WAP applications. In this implementation a file that contains all the data necessary to continue the session is written on the server. The file is given a name that is based on the user's session ID, which makes it easy to open the right file when the user makes new requests.

If the site navigation is based on one document, for instance a menu, there has to be a way in the script to separate ongoing users and new users. Of course this can be avoided with a welcome or log in page where the user is given a session ID. This ID then allows the user to load other pages on the site. Another way is to check for a session ID among the parameters sent from the client. If there is no session ID the client is obviously a new user. In this implementation the second case is used. Whenever a client requests a page on the site the server checks for HTTP parameters sent by the user. If there is no session ID in the HTTP header the script publishes a text that welcomes the user to the site and it also writes a new session file to disk. But if there is a session ID in the header the server instead opens the session file and reads the value written there. In this case the value indicates how many times the user has been at this page before and the server presents the value to the user before it updates the value and writes it back to the file. An example of how this can look like is presented in Figure 14.1 where the Ericsson R320 emulator is used to navigate through the implementation.



Session
Welcome to a new session!
Move on

Session
You have been here 1 times before.
Move on

Session
You have been here 2 times before.
Move on

Figure 14.1: An example of the session management implementation.

If there is a file added to the system every time a new user enters the site there will become a lot of files on the server’s hard disk after a while. Obviously there is a need for some clean up procedure, which either is included in the scripts or is a small program running regularly on the server. If it is done by the script it will be run every time there is a user request and this will lead to the drawback of slower response time and very much read and write operations on the disk. It is therefore better to write a program that runs on the server regularly, e.g. once a minute. The program will look at the latest access time on the files and from that information decide whether or not the file should be deleted. The time a session file shall be accessible since the last access, i.e. not become erased, is up to the system administrator but 20 minutes is common. If the session contains data that has a high security demand it might be better to select some shorter time.

15 Implementation 2: Calculate pension

15.1 Background

On the Internet site http://www.pension.nu there are different services available for Swedish citizens related to their upcoming pension. It contains information about how the pension system works and how to interpret the information sent home by the public authorities, i.e. the National Social Insurance Board. This information contains among other things a calculation of how much money one will get when one retires. This calculation is, however, very pessimistic and it takes nearly no consideration to increases of wage or growth of the public finances.

There is an application on this site where a user can calculate the forthcoming pension, based on the user's own knowledge about the future. The user can fill in upcoming raises of the income and see how the birth of a child will influence the pension etc. If someone also has different opinions about the growth of the public finances or own savings in pension funds, those can be changed to appropriate values. The user can now see how these altered values will influence the pension.

The first part of the application contains two fields where the user can fill in the information sent out by the authorities, as well as the current income as shown in Figure 15.1. Those values result in a just as pessimistic calculation as the one done by the authorities, with the exception for the influence of the last year's income. The second part of the application contains the parameters that the user can change to see what upcoming events etc. will lead to in the resulting pension. That page is shown in Figure 15.2.

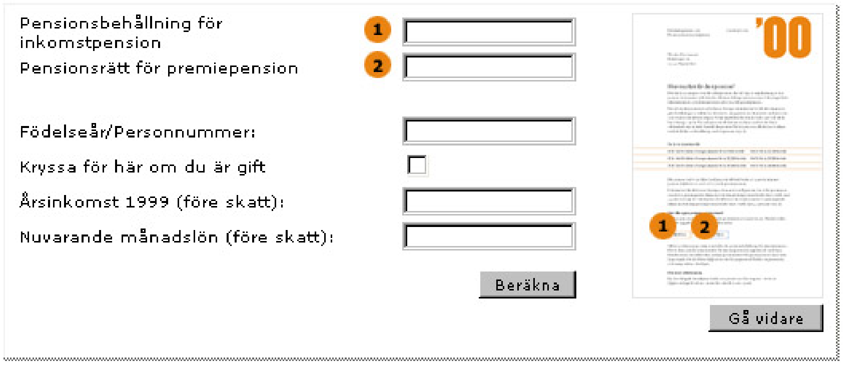
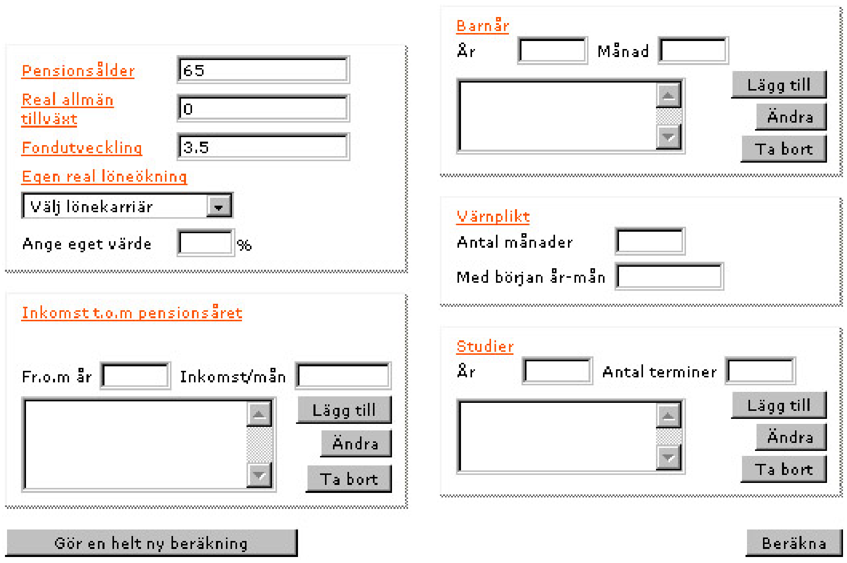

Pensionsbehållning för inkomstpension
1
Pensionsrätt för premiepension
2
Födelseår/Personnummer:
Kryssa för här om du är gift
Årsinkomst 1999 (före skatt):
Nuvarande månadslön (före skatt):
Beräkna
'00
1
2
Gå vidare

Figure 15.1: The first part of the web version of the pension calculations.

The application mentioned above is a client side script written in the programming language JavaScript and it is running in the client's browser. The script is developed at Sjöland & Thyselius on a commissioned work, ordered by the authority responsible for the web site. To prevent other service providers to use the script in their applications there are no comments in the code at all and the variables have quite cryptic names. Furthermore the code is unstructured and very large. The size of the code comes from the fact that it is rather complicated calculations to handle and there are a lot of exceptions and different factors involved. The reason why the script originally was placed on the client was to handle erroneous input by the user in an easy way. Furthermore there is in most cases no need to access data from any other information source than what the user already knows and has to fill in so therefore it is practical to let the script reside on the client.

15.2 Purpose and problems

The purpose with this implementation was to enable this script for WAP devices. Obviously there are some troubles in doing so. Firstly a mobile phone with WAP support does not understand JavaScript and secondly this large script would never fit into the memory of a device following the principles that WAP forum stated. It would also demand very long download time since the bandwidth in the mobile networks available today is too small. While the heavy calculations demand a very large script it is not possible to re-write the code in WMLScript since the gateway will not accept large quantities of data anyway. Obviously there is a need to reconsider the matter and try to see it in a different way. In the WAP case it is consequently necessary to place the script with its calculations on the server instead. In this case a modification of the script would make it possible to run on the server side and let it become a dynamic application of the kind discussed in Section 12.



Pensionsålder 65
Real allmän tillväxt 0
Fondutveckling 3.5
Egen real löneökning
Välj lönekarriär
Ange eget värde %

Inkomst t.o.m pensionsåret
Fr.o.m år Inkomst/mån

Barnår
År Månad

Värnplikt
Antal månader
Med början år-mån

Studier
År Antal terminer

Figure 15.2: The second part of the web version of the pension calculations.

Another problem with the implementation of the calculation application on WAP is the size of the input data. In the web page there are up to 15 different fields of data to be filled in by the user. These huge amounts of edit fields and selection boxes is not very handy to include in one WML document since it will force the user to scroll a very large document. It is also questionable if the WAP gateway will let the WML document through since its size will probably exceed the limit of what is acceptable. Some of the edit fields on the web page are multi-rowed. That makes it even more complicated since there are no such input fields available at all in the WAP specifications. In this case there has to be some workaround to handle this problem and it can be solved e.g. by making one WML card per web multi-row edit box as shown later on. Those factors gives the only solution to make a server side script with session handling as described in Section 13.

The script that is to be transferred from a web solution to a WAP solution is written in JavaScript. JavaScript is a script language, which despite its name has nothing in common with Java. Instead it is, just like WMLScript, derived from the scripting language ECMAScript. In a previous section about dynamic WAP applications there was a description on Active Server Pages, usually called ASP. Earlier in this text you could read that scripts for ASP applications are written in either VBScript or in JScript. This JScript is the Microsoft implementation of JavaScript and it is, except for some minor changes, exactly the same programming language as JavaScript. Hence, the client side script written in JavaScript ought to be converted to a JScript server side script running on a web server that supports ASP. Now this is, as usual, easier said than done. Firstly the client side script collects all its variables through the web page interface and secondly it operates intimately with the browser client by using cookies etc.

15.3 Session management

The first step towards a solution of the pension calculations problem is to set up a server side application in ASP. To make the application uniform all the scripts are written in JScript, the language that the calculations are written in. To separate the calculation part from the user data input part the calculation script is placed in a separate file that is called by the user when all necessary data is entered. The data input fields are divided into groups where each group forms a WAP page that fits appropriately in the display of a normal WAP device, such as Nokia 7110 or Ericsson R320. Those WAP pages are actually ASP scripts that generate WML documents, which can be sent to the client. They are scripts and not ordinary WML files because of the session management.

There has to be a session ID sent between every document to be able to track the user and the data the user filled in the input fields has to be sent back to the server where the scripts can handle them. All the documents, i.e. the pages with input fields and the calculation script, is collected under a menu which makes it easy for the user to navigate through the different input fields and fill in the required and desired data. The menu can be seen in Figure 15.3, which is how it looks on the Ericsson R320 device. When all data is entered the user simply selects the *calculate* link from the menu and the script will be executed. The script's output data is placed within a WML document and presented to the user, who is then given the options to change some input data or to make a whole new calculation.

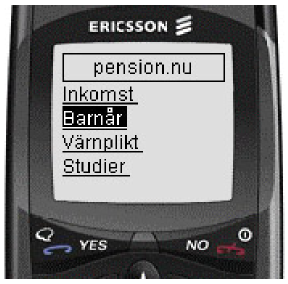


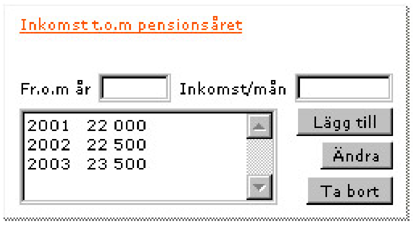
Figure 15.3: Example of the menu in the pension calculations.

To collect and save the data the user fills in the same technique is used as described in the previous section. When a user arrives for the first time a welcoming text and the navigation menu is displayed. At the same time a text file is written to the web server with the default data that the application always begins with. If the user clicks one of the links in the menu, the corresponding WML document is built on the web server by executing the script the link addresses. The script writes default values to all the input fields on the page, based on the data in the session file. The session ID is written as a parameter on the links so that the session will not be lost when the user returns to the menu. If the user changes any data and decides to save the changes the new values are written to the session file by a script, that immediately redirects the user back to the menu. Since the session ID is always sent with the requests in this way, nothing prevents several users from doing their calculations at the same time.

15.4 Special input methods

Several input fields on the web page are so called multi-row edit boxes. One example of such an edit box is shown in Figure 15.4. The data in those boxes are meant to be in a matrix form, i.e. one or more rows with two columns, and it is best to keep it in this way. Otherwise it would require some major changes in the calculation script since the script is supposed to get its data directly from the web page.

Unfortunately, as mentioned, WML have no support for multi-row edit boxes but instead a workaround was used. The user is never supposed to write any data directly into the edit box on the web page but uses instead the two ordinary input fields above to fill in a row of data. The data is then added to the edit box with the add button, in the picture named *Lägg till*. There are other buttons for the functions to change, or rather remove, the last row and to clear the whole edit box. The fact that the user never enters any data in the multi-rowed edit box leaves the possibility to build one WML document for each occurrence of those edit boxes on the web page.



2001 22 000
2002 22 500
2003 23 500

Figure 15.4: The input field for income in the web version.

On the first rows of the WML document the contents of the “edit box” are presented to the user, i.e. the data already entered. After this information there are two input fields, one for each column in the matrix just like on the web page. In the income case the user can here enter an upcoming year and the monthly salary that is expected that year. Finally there are three links available that corresponds to the buttons on the web page, i.e. one to add the entered data, one to remove the last row and one to clear the whole table. Each time the user clicks one of the links the data is saved on the server and a new WML document is generated and presented on the device. When the user is satisfied there is an *OK* link that navigates back to the menu. An example of what this can look like is presented in Figure 15.5 where different parts of the same page can be seen.

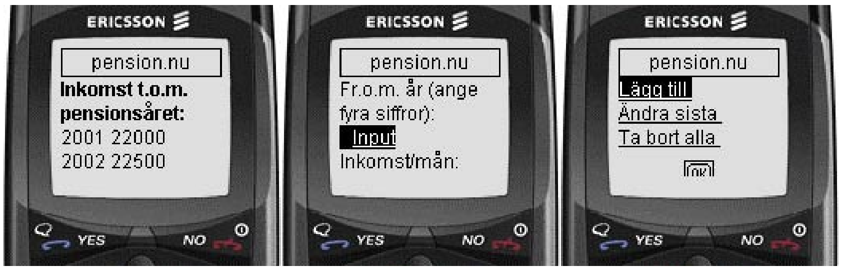


Figure 15.5: An example of the input page for future salary.

In some of the input fields the values are not integers but instead decimal numbers. On an ordinary computer keyboard this is of course no problem, but on a mobile phone keyboard it easily becomes one. Normally the keyboard on a phone has 12 keys for user input. The alphabetic characters are distributed in groups of three to four characters on each key. To enter a certain character the user browses through the letters on the appropriate key by pressing it repeatedly until the right character appears. When entering text this is the best way to handle the lack of keys but if the user is about to enter numbers it becomes a real pain. In text edit the number characters are usually placed as the fourth or even as high as the seventeenth character on a certain key.

Most of the input fields simply require integer values and a special format code has, in those cases, been set as an attribute to the input element. This format code allows nothing else but numeric characters to be entered and that makes it far easier for the user, since there is no need to browse between different characters. When entering decimal numbers, on the other hand, there has to be a decimal point somewhere. This point cannot be entered if the input field is set to only allow numeric characters. One possible way is to set a position in the format code to be a point. In this way there must be a strict number of numeric characters before the point. In this implementation this should not be any larger problem since the two values that require a decimal notation is about financial growth, which normally stays below 10%.

It could therefore be possible to let the input data be written into the form as one digit before the dot and one or more digits after. Unfortunately there is one more problem in this special case. The financial growth, though, can be a negative number and this requires an optional minus sign. Therefore there is no use of the format codes in any possible way. So the user will, after several input fields with simple numerical values, suddenly have to browse for the characters on the keys. It is not a uniform way to handle user input but unfortunately unavoidable.

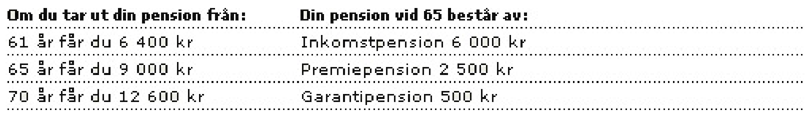
15.5 Erroneous user input

As mentioned before the calculation script in the web implementation was executing directly on the user's browser. The script could therefore handle events in the user interface. In this case an event was generated when the `user input focus` left an input field and it occurred whenever the user selected another input field to enter data. It is the field that looses the `input focus` that generates the event, which in turn executes some special code in the script. The script used those events to check user input for errors and if an error occurred it alarmed the user immediately of erroneous or missing input of values.

With the WAP implementation the script is running on the server and not on the client. It can therefore not alarm the user of missing values. The values are collected from the session file by the script and there is no way for the system to check the file before it is needed by the script, at least not in any faster way than the script itself. What can be implemented though is a WMLScript that checks the user input for erroneous values, e.g. if a value is too high or too low. The drawback with such an implementation is the need to load the script file even if the user input is valid. The positive effect is that the user does not have to call for the calculation script to find out that a bad value has been entered. The same script can certainly also check more than one WML document and therefore it will, in the end, probably save the user some loading time as long as the script is kept in the device's cache memory.

15.6 Running the calculation

At the bottom of the menu there is a link available to start a calculation, which the user can click when all necessary data is entered. When the web server gets the request it executes the modified calculation script, which now reads its data from the session file instead of the input fields on the web page. All the collected data are read from the session file and assigned to the corresponding variables in the script. In the web browser version of the script a special frame on the browser was used for output in HTML format, see Figure 15.6. The result does not only present how much money the person will get in pension every month but it also lists how different ages for the retirement affects the pension. There is also a list of what different components the pension is built up of and how much money each component generates.



Om du tar ut din pension från:	Din pension vid 65 består av:
61 år får du 6 400 kr	Inkomstpension 6 000 kr
65 år får du 9 000 kr	Premiepension 2 500 kr
70 år får du 12 600 kr	Garantipension 500 kr

Figure 15.6: The result of a calculation in the web version.

In the conversion process from a web solution to a WAP solution, there are naturally some big differences of what the output from the script can look like. Firstly the output document code has to be changed from HTML to WML so the WAP device can understand the data. Even if the WAP gateway normally does its best to convert HTML documents into WML it is always better to write all code in the correct page describing language of WAP. In the web version the output was displayed in an own frame at the bottom of the browser, but in a WML browser there are no frames available. Instead there must be a way for the user to navigate back from the result if there is something that has to be changed.

As a second step there are two links added to the script output, one to return to the menu with the same session still active and one that returns to the menu page but without any session ID. In the last case a click by the user leads to a new session. The user is once more facing the welcome text and all the pages that are linked in the menu contains the default values. This could be nice if the user wants to calculate another person's pension after the user's own calculation is finished.

Finally the web version output was ordered in a table that made it easy to read and survey the data. In WML there is a simple table format that resembles its HTML counterpart but there are several reasons why it should not be used. Not all WAP enabled devices supports the table element in WML. Nokia 7110, for instance, ignores the rows and places all table data in one column. Since the display is so small it does not matter that much since the rest of the columns with all probability would fall outside the display if it were rendered correctly. Therefore it is better to write the output in plain text and let the browsers render it, as they like. To make the text easier to read, some lines are made bold faced to separate headings from the data but even here not all the devices support the formatting. In this case the Nokia 7110 is once again a good example because the browser in that phone ignores all kinds of text formatting, such as bold, italic and underlined text. An example of the final result of a calculation in the WAP implementation is seen in Figure 15.7 where different parts of the output text are presented.

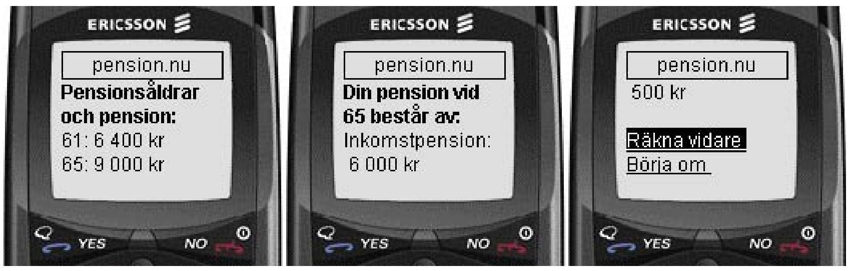


Figure 15.7: The result of a calculation in the WAP version.

In the web implementation of the calculation script there was a cookie set for almost every data item the user filled on the web page. This led to a lot of cookies on the user's browser but on the other hand it also led to the fact that the user could get back to the web page any time later and continue the calculation since all values were restored from the cookies. In this way the user could be able to change some values whenever the situation changed. Since cookies are not applicable on WAP this functionality must be removed.

The only possible way to let the user return to the server and continue the calculations is if the user could enter the session ID by hand or by using some sort of identification such as for instance a username. This would on the other hand lead to an enormous demand on hard disk space on the server if the site would be popular, a problem that is quite insignificant when each user stores the session data on her own disk. Therefore this kind of service is not possible to realize in the converted implementation. Instead it is necessary to delete the sessions when they are not used to save disk space. This leads to the need of some cleaning program running as a service under Windows NT that is deleting all files that have not been accessed the last 20 minutes or whatever time seems sufficient.

15.7 Conclusions for future development

This is a perfect example of the need to have WAP in mind when developing web services that are to be implemented in both WWW and WAP technologies. In this case the WAP implementation of the pension calculation was thought of after the web application was finished. But for future application development it would be both timesaving and providing a better result if both a WAP and a web solution where included in the project from the beginning. An important experience is that even if both WAP and WWW are technologies that are using the Internet and that they are related to each other there are some big differences in the practical implementation and usability.

16 Implementation 3: Own WAP gateway with a modem

Sometimes a service provider wants to get full control over the services it supplies, e.g. to avoid security problems with the Internet or to control to whom the services are delivered. The solution is to put up a set of modems to which the clients can connect, i.e. call up with their WAP enabled phones. When the clients are authenticated and logged in to the network they can access a private WAP gateway, which gets the contents from a web server. This procedure is called total corporate solution in Figure 16.1 below.

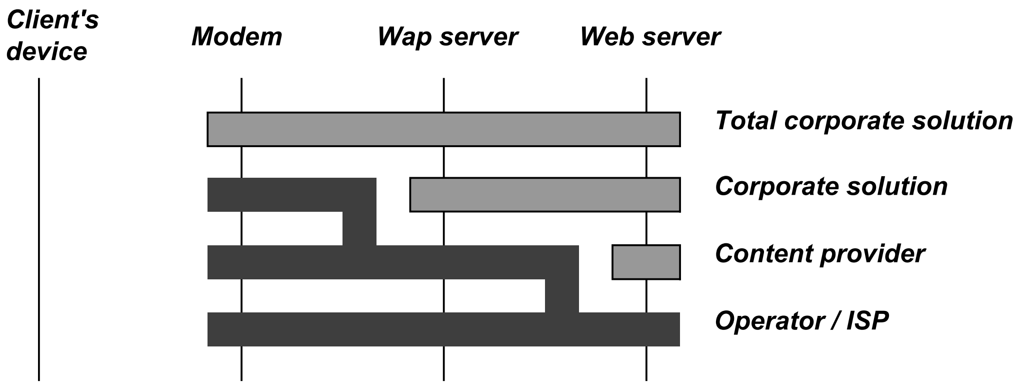


Figure 16.1: Different service provider approaches.

It can seem pretty complicated for the user to change the phone number, all the login and communication settings for each service he/she communicates with, but mostly it is not. Both the Ericsson and Nokia mobile phones released today, i.e. Ericsson R320 and Nokia 7110, have support for different communication settings. The user can set up six different modems to connect to and each of those connections has their own settings for username/password pairs for authentication, homepage and gateway address. In this way the customer can easily select the appropriate setting depending on the service needed. There can for instance be one setting for the ordinary ISP, one for bank services and one for stock trading.

Setting up a connection between a WAP enabled phone and a modem is just like setting up a connection between an ordinary computer and an ISP. The mobile phone uses the Point-to-Point Protocol (PPP) to connect to the other party, which usually is a modem attached to a server. When the connection is established the phone can be authenticated using the standard Password Authentication Protocol (PAP) where the client's username and password are sent to the server.

PAP is using plain text to send the authentication data to the server and this is usually not a problem since it is sent over a telephone connection and not over an open network like the Internet. But if the information is classified as very sensitive and there is a calculated risk of interception on the telephone line there is a possibility to use secure authentication with the Challenge-Handshake Authentication Protocol (CHAP), at least when using the Nokia 7110. CHAP uses one-way hash functions and it can also request authentication anytime during the connection. This is therefore considered a more secure way to authenticate a client.

One of the advantages with the total corporate solution is to get end-to-end security. If the connection between the client's WAP device and the WAP gateway is secured with the WTLS security protocol, an intruder not only needs to crack WTLS but also to monitor the mobile network traffic or tap the telephone line connected to the modem. If the mobile network used for instance is the GSM network the traffic already is encrypted, and consequently the intruder needs to crack that encryption first. In this way a higher security is achieved than what is possible when the traffic has to go over the Internet as well.

Another advantage with this solution is the ability to use push services, which is available in version 1.2 of the WAP protocol. Since push is not supported by HTTP the service provider has to use a corporate solution to offer such a service.

The first thing needed to set up a total corporate solution system is some kind of modem and in this implementation an ordinary 14 400 bps computer modem was used. Since GSM does not allow any faster communication than 14 400 bps it was fully sufficient for this project. For commercial use, on the other hand, some kind of modem pool with multiple lines ought to be used so that several users can be connected at the same time. The modem then needs to be connected to a server that can manage incoming calls with the use of the PPP protocol, i.e. to answer the modem when a client calls and to set up a connection between the peers.

As the second step there has to be a WAP gateway on the answering server or on its network. The user needs to know the IP number to the machine that is hosting the WAP gateway so that the telephone can reach it on the network. Finally, the web server where the contents to be sent to the clients device are to be found, must be accessible from the WAP gateway.

In this project two different implementations were made on the same machine. The hardware was a Pentium II 200 MHz PC connected to a modem as described above. A dual boot system was set up to get the ability to test the implementation in both Windows NT and Linux.

16.1 Windows NT and Nokia WAP Server

To get the machine up and running the Microsoft Windows NT Workstation operating system was installed. After the ordinary configuration procedure and the installation of appropriate modem drivers it was time to set up all servers and services required.

Windows will not automatically answer when someone calls the modem. Instead there is a service available within the Windows NT distribution that handles those events namely the Remote Access Service (RAS). After this service was installed it had to be started, which is done in the Windows control panel tool for services. The service was set in the automatic startup on boot mode so that the RAS would always be running as soon as the computer was turned on. The configuration of the service itself was done within the network settings and the service was configured to handle incoming calls with the use of the TCP/IP protocol. Also different levels of security for the user authentication process could be selected, such as PAP or CHAP, as well as the allocation of IP numbers for the connecting clients.

Before any client can log on to the computer there have to be some user accounts available. In Windows NT there is a user account manager to set up users and their rights on the local machine. Since the computer used in this case was standing completely alone without any network attached to it the use of local accounts worked fine. If the computer had been connected to some sort of Local Area Network (LAN) though, it might have been a bit more complicated since the account managing had been controlled over the network. All the accounts added to the system was given the right to connect through a dial in connection, which was necessary if the users should have any possibility to log on at all with their mobile phones.

The information delivered to the clients is normally placed on a web server, which consequently requires a WAP gateway within the domain. In this case the Nokia WAP Server 1.0 was selected. It is fully compatible with the WAP 1.1 standards and it is very easy to install on Windows NT. After the installation of the server was finished, there were two different ways to use it. Either it could be started manually on the computer, which required an administrator to log in every time the machine was started. The other way was to set it up as a service in a similar way as the RAS service mentioned above. In this case the latter seemed more convenient.

When the server was running as a service it was automatically started with Windows, just as the RAS, but there was still one thing left to do. Before the server could handle any requests the traffic had to be started because even though the server was running it did not respond to the outside world. As soon as the traffic was started the server could be used to deliver WAP contents and it was running until it was explicitly stopped the next time. This means that even if the computer had been shut down it started to handle user traffic as soon as the computer was turned on again. This state of a running server that is not handling any traffic can be useful, e.g. for configuration purposes or to check log files.

Finally, to be able to provide the client any sort of content, a web server was installed. Alongside Windows NT there is a product available to extend the features of the operating system that is called the Windows NT Option Pack. This option pack includes among other things a web server by the name Internet Information Server (IIS). This is the standard web server from Microsoft, e.g. with full support for Active Server Pages (ASP). The option pack was installed on the computer but not without problems.

Since the computer was running the Workstation version of Windows NT the installation program installed a simplified version of the web server called Personal Web Server. Unfortunately this server did not support additional MIME types that are required to provide WAP documents and scripts. Therefore the installation of the option pack was run once more, this time with the option to install the Microsoft Management Console. From this console the full features of the IIS server became available and the web server could be configured to deliver WAP contents.

After this procedure the whole chain of servers and services to achieve the complete WAP model on a single computer was fulfilled. The only thing required by the users was to add the phone number to the modem in a profile in their WAP device. In the profile they also had to enter the IP address to the gateway, which in the case of a stand-alone machine is the local host number 127.0.0.1, and a valid username and a password.

If the content provider in a commercial implementation has the ability to send SMS messages to the users it is possible, at least if they have the mobile phone Nokia 7110, to send all necessary settings via an SMS message. In this way everything will be configured automatically and the user will not have to feel that the advanced technique and the complicated settings is an obstacle in the way of using the WAP technology.

16.2 Red Hat Linux and Kannel Gateway

There is an open-source WAP gateway available for download on the Internet called Kannel. Besides the WAP functionality it is also an SMS gateway that can be used simultaneously with the WAP gateway. The project to develop the Kannel gateway was founded in June 1999 by Wapit Ltd., which is a member of WAP Forum. The latest version is 0.7.1 and as the version number indicates it is not yet complete. It does not support connection-less operation and it has no support for the security protocol WTLS. The lack of support for connection-less operation has one negative effect on the testing of the implementation. The WAP enabled mobile phone R320 from Ericsson only operates in connection-less mode, which makes it useless in this case. Fortunately the Nokia 7110 supports both connection-less and connection oriented operation and it was with that device the WAP gateway was tested.

First the Red Hat 6.2 Linux operating system was installed on the computer. In the installation process there were some very important packages that had to be included. Firstly the Kannel gateway required a library that was included within the distribution of the window manager GNOME. Therefore, even if there was no intention of using GNOME as the window manager on the system, it was important to get those libraries installed. Since there also was a need for a web server, the Apache server, which is included in the Red Hat distribution, was installed as well. When the installation process was finished some user accounts were set up before the configuration of a dial-in PPP connection were made.

In most Linux systems there is a PPP daemon available, e.g. for dial-in and dial-out connections. This daemon provides a very powerful PPP connection between two peers with a lot of options and customization features. As usual with Linux there are no nice dialog boxes with radio buttons and check boxes available but instead there are plain text files for all configuration. The configuration files for the PPP daemon, which is named pppd, in the actual distribution of Linux are placed in the /etc/ppp directory.

The first file that was edited was a file containing all attributes the pppd required. This file was named *options* and, after some experimenting, the file finally looked like this:

```
19200
lock
silent
-detach
crtscts
netmask 255.255.0.0
asyncmap 0
modem
proxyarp
auth
-chap
+pap
login
```

Some of the more important options in this file are the first option that indicates a baud rate by which the computer communicates with the modem, `auth` that indicates that the peer has to authenticate itself and `+pap` which requires the peer to use PAP for the authentication. The `lock` option provides a mutual exclusion on the connection port so that there will be no conflicts that can interrupt the traffic. Finally the `login` option makes pppd use the system password table for the authentication of the peer.

When the *options* file was finished there was a need to write a file that contained some specific options for the serial port that was connected to the modem. In this case the modem was connected to the device *ttyS1* and accordingly the file was named *options.ttyS1*. The content of this file was nothing more than one line with the hostname followed by an arbitrary name for the virtual host that is associated with the PPP dial-in line.

```
wapserver:ppp01
```

The last file located in the directory was *pap-secrets*. Normally this file can be used to set up the peers' usernames and their corresponding passwords along with the IP addresses that will be assigned to the users. In this case the authentication was made against the password list of the system and therefore this file only had to contain the IP addresses that would be handed out. Therefore the contents of the *pap-secrets* file became like this:

```
# Secrets for authentication using PAP
# client   server         secret IP addresses
*               *           ""     10.0.1.1
```

If any IP number can be assigned to the peer the number above can be substituted by a star (*) but this might lower the level of security.

By this time a functioning PPP server had been set up and it was started by the command:

```
# pppd ttyS1
```

After a short initialization the pppd program contacted the modem and put in answering mode. As soon as a WAP enabled mobile phone called the modem the PPP daemon answered the call and set up a Point-to-Point connection. When the client successfully authenticated itself over the Password Authentication Protocol the connection was ready for use. A major drawback so far was the start of the pppd program. An administrator had to log on to the computer to manually start the daemon and as soon as the user hung up the pppd exited. This is of course not acceptable in practical use and therefore there was a need to get the program running automatically. The solution was to edit the *inittab* file in the */etc* directory. This file is very important for system startup and must therefore be treated very carefully or otherwise the system might not be able to boot. To get the pppd to start automatically and to get it restarted whenever it has closed down the following line was added to the *inittab* file:

```
pd:2345:respawn:/usr/sbin/pppd ttyS1
```

`pd` is an arbitrary and unique name for this service, `2345` is the run levels where the program shall be running and `respawn` tells the system to restart the program if it exits. Since no environment is loaded at this time the complete path to the program file must be stated.

When the configuration of the dial-in service was completed it was time to set up the WAP gateway. Since this is in the world of UNIX the source code had to be compiled before any program could be executed. Fortunately the compilation phase was very easy. By running an included shell script that configured a Makefile file and then simply running the `make` command the gateway was compiled and ready to run.

Kannel is built up around three major components. The first is the bearer box that is a sort of main program for the Kannel. It is like a server that one or more WAP and SMS boxes can connect to. The other two main components are the already mentioned WAP and SMS boxes that carry the actual functionality that the gateway consists of. There are three configuration files, one for each of the three components.

Those configuration files controls how the communication between the bearer box and the other boxes is handled as well as the boxes' internal behavior. The files are very well commented and very much self-explaining. Among the features there are things like http administration and logs of events in different levels. In the configuration of the WAP box the different implementations of the WAP protocol can be set up but as mentioned before there is in this version only support for connectionless mode over the WSP/WDP combination.

The last step was to get the web server running. Since the web server option was selected in the installation process the Apache web server was already installed and running. To get a proper behavior of the web server the file */etc/httpd/conf/httpd.conf* was edited, even though most of the settings worked fine. The two lines changed in the file were the ones specifying the `DocumentRoot` and the `DirectoryIndex`. The document root is the path to the files the server uses for all files that is not owned by a system account. The default document name is set with the option `DirectoryIndex` in the configuration file. This option should include a WML document if the server is supposed to deliver such type of documents, e.g. *index.wml*. If a client, for example, requests a document with the URL *http//domainname/* and the `DocumentRoot` is set to */home/httpd/wml* and the `DirectoryIndex` is set to *index.wml* the server will respond to the request by sending the file */home/httpd/wml/index.wml* back to the client. Finally the MIME types for the WAP documents had to be entered in the system. The Apache web server uses the file /etc/mime.types and consequently the WAP types were added to this file.

17 Final conclusions

17.1 Usability

At the time of writing there are many people criticizing WAP, in the first place the differences in how the information is rendered on different devices. This is a problem since the same WML document is rendered differently on different devices. An example is the line breaks on the Nokia 7110. This device adds line breaks after some elements that make it nearly impossible to get the line spaces or the length of lines equal in the 7110 and the Ericsson R320. It is strange that Ericsson and Nokia have those disagreements since they are the most ardent advocates of the WAP technology.

There are also complaints on the small display and the limited input device a mobile phone offers. Even if this criticism is correct in many ways the persons behind the complaints often forgets what the goals of the WAP forum were. The basic idea with WAP was to get Internet access from small handheld devices, particularly mobile phones. Therefore the whole protocol stack, including the markup language WML, was developed for small displays and limited input devices, not the contrary. It is nearly impossible to get a device with the same characteristics as a desktop computer to fit in such a device that WAP is intended for.

17.2 Power consumption and user costs

When the WML content is passing through the WAP gateway it is byte encoded. This encoded data saves plenty of bandwidth on the wireless network as well as power on the client. The reduced bandwidth results in lower costs for the user, either because it is faster to download or because it is a smaller amount of data, depending on how the user is charged by the network operator.

Many people claims that WAP will be unnecessary with the arrival of new network bearers, such as GPRS and UMTS, since the network bandwidth will increase. It is true that it will became possible to use a HTTP stack over the network bearer when the bandwidth is similar to ordinary modems, but there are still some advantages with WAP. Firstly WAP does not require as much packet overhead as the HTTP protocol. Furthermore the encoded data is easier for the device to render. In the HTTP case the data is delivered in plain text, which enforces the device to analyze the HTML document to find tags etc. Both those factors leads to lower power consumption and the latter also to cheaper processors since the workload are smaller with the WAP technology.

17.3 The WAP gateway

The user depends very much upon the WAP gateway. If the gateway cannot handle the traffic load or if it is down there will be no connection to the Internet at all. An alternative to the use of a gateway could be to let the service providers provide byte encoded data directly to the client. In this way the need for a gateway would disappear while the advantages with the bandwidth savings would remain. Of course the gateway fills other needs, or another strategy had been chosen by WAP forum.

Thanks to the use of the gateway, sessions can be suspended and resumed without the overhead of initial establishment. The gateway also handles all the DNS services to resolve domain names used in the URLs. Those functions save power at the client as well as network resources.

17.4 Security

The WAP technology is using both the web and the WAP protocol for its communication over the Internet and the mobile networks. A secure connection between a client and a server is therefore also using two different protocols. Over the web connection the Secure Socket Layer (SSL) is used and the WAP connection uses the Wireless Transport Layer Security (WTLS) protocol. The link between those two protocols is the WAP gateway, which decrypts the data from one security protocol to encrypt it with the other security protocol. This opens for a security hole since the data is not encrypted within the WAP gateway. It is therefore very important that the gateway is managed by a reliable part and that the gateway is protected against all kinds of intruders.

The positive effect of using a standard protocol on the web is that a content provider does not need to learn a new security protocol. For instance, if a bank wants to put up WAP services, they can use the same web server and the same SSL software as with their web solution. In this way it will be both easy and cost effective for the bank to develop new services. At the same time bandwidth is saved on the wireless network thanks to the use of WTLS.

17.5 User sessions

A WAP browser lacks the feature of cookies. Cookies are a way for web servers to save data on web browsers. The server can, for example, use the data to identify a client by looking at a cookie set by the server at an earlier time. This method to identify a client cannot be used by WAP since WAP browsers do not support cookies. Therefore, a user is required to log on to all sites with personal information, which makes e.g. personalized starting points much more inconvenient with a WAP browser than with a web ditto.

17.6 Service provider

It is cheap for a service provider to put up WAP services since all information is placed on the already existing web servers. The server is ready to deliver WAP contents after some minor configuration of the web server software. This is possible since WAP is developed to use as much of existing technologies as possible and therefore uses the HTTP protocol between the WAP gateway and the web servers.

17.7 Reasons to use or not to use WAP

In the comparison between WAP and HTTP there are several advantages with WAP, such as the limited bandwidth requirement, the lower power consumption and the lower demand for computing power. Those factors give the user a cheaper device, more stand-by time and faster connections. The mark up language is developed for small displays and limited input devices. Some people claim that HTML is a better alternative than WML since the developers already know HTML and the WAP enabled devices interprets WML differently. Instead, a use of HTML would require a limited version of the language, similar to the language used by the I-mode system in Japan. This is caused by the limitations of the user interface. Since the developer however needs to learn a new language and a new technique to develop services there are no advantages of using HTML over WML. Furthermore are the disagreements of how the pages should be rendered not a matter of which markup language that are used, but rather a question of how it is implemented. WAP is therefore a good platform for the development of mobile Internet services, at least in contrast to the existing alternatives today. It is also the only platform that is supported by the manufacturers of mobile phones. This is an argument that really speaks in advantage of WAP.

A drawback of WAP is the focusing on the technology rather than the needs of the users. The upcoming push functionality is very much focused on stock trading and similar activities but there are no efforts to make WAP a powerful platform for instance to games. A very large group of mobile phone users today are the teenagers and young adults. They do not only frequently use the phone to call their friends but they also use the SMS text message function more than any other user group. Obviously there is a category of users that is not thought of in the development of WAP, even if it is a group with relatively high purchasing power. A platform more suitable for live chat, messages and online gaming would have increased the potential group of users.

17.8 Development of services

It is much easier to develop applications for the web and WAP at the same time than to convert web applications for WAP afterwards. If the script used in the pension calculations (Section 15) had been separated from the functionalities of the web, it would have been much easier to convert the script for the WAP implementation.

If the application is a server side script or program, a good idea could be to use different interfaces between the server and the application. There would be one interface per technology and a request would use the corresponding interface of the technology the requesting client uses. The output data from the application can in this way be formatted differently according to the client. It will also become quite easy to add new technologies to the application without the need of rewriting the application program.

Appendix A: XML

XML is a markup language similar to HTML but without the limitation of being a visualization language for the web. It is rather a general markup language that can be used for different applications for information exchange in some specific domain of human activities [8]. XML is based on SGML that was introduced in 1969 and standardized in 1986. The first W3C recommendation on XML was released in February 1998 and is called XML 1.0. The markup language uses tags to delimit and describe data in a hierarchical way. It is written in ASCII text, which make it platform independent and therefore easy to use on the Internet. XML is, unlike HTML, strictly case sensitive. Internationalization is an important part of the language and therefore it uses 16-bit Unicode character-set so the markup can be written in any language.

There are two widely spread APIs named Document Object Model (DOM) and Simple API for XML (SAX). DOM is a W3C Recommendation that lets the programmer create and modify XML documents as program objects. It presents the entire XML document in a tree view and therefore the entire document is loaded into memory as once. SAX, on the other hand, is not any product of a standardization organization but the result of a number of XML developers who needed an efficient API to develop XML. Instead of the tree view of the complete document used by DOM it offers events as it parses the document, step by step.

Further functionality is transformation of XML documents, connection to databases, server-to-server communication, e-business, etc. Even if XML is not about styling there are some styling languages available, such as Cascading Style Sheets (CSS) and Extensible Stylesheet Language (XSL).

Document Type Definition, DTD

The Document Type Definition is the description of a XML document. It gives the document a vocabulary; a definition of the special needs and structures. A DTD is necessary to create a valid XML document in contrast to a well-formed one. Well-formed XML conforms to the basic syntax of XML but there are no other syntactic rules. Those rules are instead provided by the DTD, which is used by many parsers to validate a XML document. The DTD can be placed inside the document or in an external file reached by a URI or a URL. The advantage with an external definition file is that the same file can be used by several XML documents using the same vocabulary.

There are four types of declarations that build up a DTD: elements, attributes, entities and notations. An element is the core of XML, i.e. the markup tag, the attributes are parameters to the elements, entities are reusable contents and notations are non-parsed content.

Four different types of elements are available. The simplest element type is empty, which means that it contains neither text nor child elements. An empty element can have attributes though. The second type is the element, or in different words element-only. This type has no text but can have child elements. Elements with both text contents and child elements are called mixed. Finally there is a type that is open to any content that doesn't violate the XML syntax. Those elements are of the type `any` but should be used with restriction since they reduce the level of strictness. Further there are rules available that defines order and numbers of child elements as well as if they are needed or optional.

An element has a list of attributes and an attribute has a name, a type and a default value. The default value can be, as the name indicates, a value or a notation that the attribute is required, fixed or optional. Even attributes have different types. The basic type is the character data type that is simple text with no markup contained. Other possible types of attributes are id:s that gives the element a unique identity, name tokens, enumerations or entities. There are also techniques for conditional sections in the DTD to provide further functionality and code reuse.

Entities are, as mentioned earlier, reusable contents. It can be compared with the define statement in the c programming language as it replaces an entity reference with some contents.

Parameter entities are a type of entities that makes a set of reusable attributes for elements with some or all parameters in common.

A notation, finally, is some content that falls outside the scope of XML and therefore is not to be parsed. It can be e.g. some binary content such as an image.

Appendix B: Abbreviations

API	Application Program Interface
ASP	Active Server Pages
CGI	Common Gateway Interface
CHAP	Challenge-Handshake Authentication Protocol
CSS	Cascading Style Sheets
DNS	Domain Name System
DOM	Document Object Model
DTD	Document Type Definition
ISP	Internet Service Provider
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IIS	Internet Information Server
IP	Internet Protocol
ISP	Internet Service Provider
JSP	Java Server Pages
LAN	Local Area Network
OS	Operating System
PAP	Password Authentication Protocol
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PHP	Personal Home Page
POP	Post Office Protocol
PPP	Point-to-Point Protocol
RAS	Remote Access Service
SAX	Simple API for XML
SDK	Software Developer's Kit or Service Development Kit
SMS	Short Message Service
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security Protocol, formerly known as SSL
UI	User Interface
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

WAE	Wireless Application Environment
WAP	Wireless Application Protocol
WDP	Wireless Datagram Protocol
WML	Wireless Markup Language
WSP	Wireless Session Protocol
WSP/B	WSP services suited for browsing applications
WTA	Wireless Telephony Application
WTAI	Wireless Telephony Application Interface
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol
WWW	World Wide Web
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

Appendix C: References

[1] AU-System Radio AB. 1999. WAP White Paper. Available: http://www.wapguide.com/ 28 February 2000.

[2] Computer Sweden. April 17, 2000. Expert varnar för waps säkerhetsprotokoll. IDG AB.

[3] Computer Sweden. April 17, 2000. Virus för wap hejdas i servern. IDG AB.

[4] Computer Sweden. March 1, 2000. Japan ett år före på mobilt internet. IDG AB.

[5] Computer Sweden. March 22, 2000. Fem miljoner kunder i japansk portal. IDG AB.

[6] Datateknik 3.0 No. 3. February 24 2000. Ett steg kvar till Wap-telefoni på riktigt. E+T Förlag AB.

[7] Mobile Lifestreams Ltd. 1999-2000. An Introduction to WAP. Available: http://www.mobilewap.com/ February 28, 2000.

[8] Mohr, Stephen (Editor). 2000. Professional XML. ISBN 1-861003-11-0, Wrox Press Ltd.

[9] Nätverk & Kommunikation, No 3. February 22, 2000. Casio hoppar over hypade WAP. IDG AB.

[10] Nätverk & Kommunikation, No 3. February 22, 2000. Mobil och bärbar uppkoppling. IDG AB.

[11] Nokia Corporation. 1999. Service developers guide for the Nokia 7110. Available: http://www.forum.nokia.com February 7, 2000.

[12] Nokia Corporation. 1999. WML reference version 1.1. Available: http://www.forum.nokia.com February 7, 2000

[13] Nokia Corporation. 1999. WMLScript reference version 1.1. Available: http://www.forum.nokia.com February 7, 2000

[14] Phone.com, Inc. 2000. Understanding security on the wireless Internet. Available: http://www.phone.com May 20, 2000

[15] TechTarget.com Inc. 1996-2000. Whatis?com. Available: http://whatis.com March 10, 2000.

[16] Telefonaktiebolaget LM Ericsson. 1994-2000. Ericsson Developers' Zone. Available: http://www.ericsson.com/developerszone June 14, 2000

[17] Wireless Application Protocol Forum, Ltd. 1999. Official Wireless Application Protocol. ISBN 0-471-32755-7, John Wiley & Sons, Inc.

[18] Wireless Application Protocol Forum, Ltd. 1999. WAP White Paper. Available: http://www.wapforum.org February 7 2000.