

## **4 Wireless Application Protocol**

W. Kou

ISN National Key Laboratory, Xidian University, P.R. China

### **4.1 Introduction**

The wireless application protocol (WAP) is a suite of emerging standards to enable mobile Internet applications. The WAP standards have been created as a result of the WAP Forum that was formed in June 1997 by Ericsson, Motorola, and Nokia. The WAP Forum is designed to assist the convergence of two fast-growing network technologies, namely, wireless communications and the Internet. The convergence is based on rapidly increasing numbers of mobile phone users and the dramatic effect of e-business over the Internet. The combination of these two technologies will have a big impact on current e-business practice, and it will create huge market potential.

In this chapter, a detailed introduction to WAP is presented, including the application environment and various protocols. The security aspect in the present Internet environment is dealt with in Sect. 4.3.

### **4.2 Wireless Application Protocol**

#### **4.2.1 Overview**

The WAP standards consist of a variety of architecture components, including an application environment, scripting and markup languages, network protocols, and security features. These components and features together define how wireless data handsets communicate over the wireless network, and how content and services are delivered. With the WAP standards, a wireless data handset can establish a connection to a WAP-compliant wireless infrastructure, request and receive the content and services, and present them to the end user. This WAP-compliant wireless infrastructure may include the handset, the server side infrastructure, such as the proxy server (WAP gateway), the Web server, the application server, and the network operator (telecommunication company). The WAP architecture is shown in Fig. 4.1.

The WAP architecture can also be presented through the WAP protocol stack shown in Fig. 4.2. The WAP protocol stack covers the complete picture from

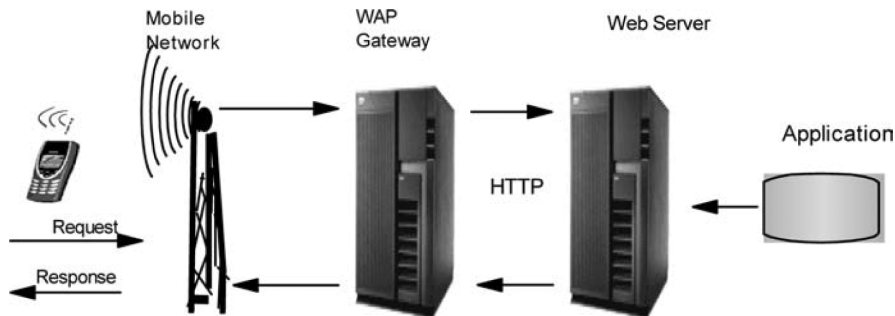


Fig. 4.1. The WAP architecture

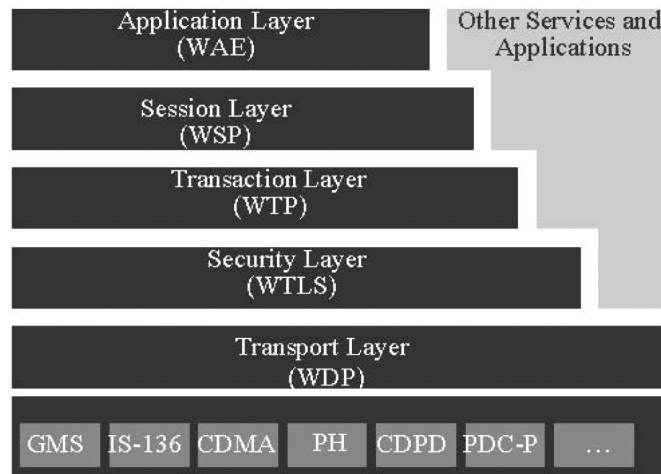


Fig. 4.2. The WAP protocol stack

bearers to applications. The bearers are the various wireless networks that WAP currently supports. The transport layer is an interface common to the underlying wireless network, and it provides a constant service to the upper layers in the WAP stack, such that the bearer services are transparent to the upper layers. In other words, with the transport layer, the specific network characteristics can be masked. The security layer provides security for the transport layer, based on the industry standard protocol and the transport layer security (TLS) protocol. The transaction layer provides a lightweight transaction-oriented protocol for mobile thin clients. The session layer provides the application layer with the capability to select connection-oriented or connectionless services. The application layer deals with a general-purpose environment for applications.

The WAP protocols in Fig. 4.2 include wireless application environment (WAE), wireless session protocol (WSP), wireless transaction protocol (WTP),

wireless transport layer security (WTLS), and wireless datagram protocol (WDP). In Sects. 4.2.2–4.2.6, we discuss these protocols with special focus on WAE.

#### 4.2.2 Wireless Application Environment

WAE consists of a set of standards that collectively define a group of formats for wireless applications and downloadable content. WAE specifies an application framework for wireless devices, such as cellular phones, pagers, and PDAs. WAE has two logical layers, namely, user-agent layer and format-and-service layer. The components of the user-agent layer include browsers, phone books, message editors, and other items on the user device side, such as wireless telephony application (WTA) agent. The components of the format-and-service layer include common elements and formats accessible to the user agents, such as WML, WMLScript, and WAP binary XML content format (WBXML).

A WAP microbrowser has the following capabilities:

- Submission of requests to the server
- Reception of responses from the server
- Conversion of and parse the data
- Interpretation from WML and WMLScript files
- Ability to interact with the appropriate WAP layer
- Local cache and variable management
- Wireless session protocol processing
- Effective management of local hardware resources, such as RAM, ROM, small screen, and input and output

##### *Wireless Markup Language*

Wireless markup language (WML) is a language based on the extensible markup language (XML). WML is optimized for small screens and limited memory capacity, and for content intended for lightweight, wireless devices such as mobile phones and personal digital assistants (PDAs).

A WML document is called deck. A page of a WML document is called card. A deck consists of one or more cards. Each deck is identified by an individual URL address, similar to an HTML page. A WML deck requires a browser that will format the deck for the benefit of the user. The browser determines the final shape of the deck. Sometimes, people use the analogy of HTML to explain WML. In the analogy, a WML deck corresponds to an HTML page. However, there are differences between a WML deck and an HTML page. While each HTML file is a single viewable page, a WML deck may contain multiple cards, each of which is a separate viewable entity. WML files are stored as static text files on a server. During the transmission from the server to the browser, the WML files are encoded in binary format by the wireless connection gateway and then sent to the browser. This is also different from HTML, where there is no need for such an encoding process.

WML contains commands for navigation in decks. Each WML command has two core attributes, namely, id and class. The id is the attribute for an individual name to the elements inside a deck, while the class is the attribute that links the element to one or several groups. A WML deck, at its most basic level, is constructed from a set of elements. Elements are identified by tags, which are enclosed in angular brackets. Each element must include a start tag (`<el_tag>`) and an end tag (`</el_tag>`). The content is included between the start and end tags. An empty element that has no content can be abbreviated by a single tag (`<el_tag/>`).

Because WML is based on the XML language, a WML document must follow the XML rule to contain the XML-specified document type definition (DTD) at the beginning of the WML code, which is referred to as deck header or document prolog, as follows:

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

A deck is defined by the `<wml>` and `</wml>` tags that are required in every WML document. Within a deck, each card is defined by the `<card>` and `</card>` tags. Both `<wml>...</wml>` and `<card>...</card>` are formatting commands. The `<wml>...</wml>` commands summarize the deck. The `<card>...</card>` commands summarize the text, images, input fields, and any other objects of a card in the deck.

Cards are the basic units of WML, defining an interaction between a mobile device and the user. Each card may contain three different groups of elements: content elements (such as text, tables, and images), tasks and events (such as `<onevent>`, `<timer>`, and `<do>`), and data entry (such as `<input>` and `<select>`).

### **WMLScript**

WMLScript is a simple scripting language based on ECMAScript (ECMA-262 standard) with modifications to better support low-bandwidth communication and thin clients. WMLScript is part of the WAP application layer.

WMLScript complements the WML by adding simple formatting capabilities to make the user interfaces more readable, for example, the capabilities of checking the validity of user input and generating messages and dialog locally to reduce the need for expensive round-trip to show alerts. These capabilities are not supported by WML as the content of WML is static. WMLScript provides programmable functionality that can be used over narrowband communication links in clients with limited capabilities. With WMLScript, more advanced user interface functions can be supported and intelligence can be added to the client. WMLScript also provides access to the device and its peripheral functionality, and reduces the amount of bandwidth that is needed for sending data back and forth between the server and the client.

WMLScript is similar to JavaScript. For example, WMLScript includes a number of operators such as assignment and arithmetic operators, which are similar to those in JavaScript. However, there are major differences between WMLScript and JavaScript. First, WML contains references to the URL address of a

WMLScript function, whereas JavaScript functions are normally embedded in the HTML code. Second, WMLScript must be compiled into binary WMLScript code prior to its execution in a WAP device, while there is no such requirement for JavaScript.

Although WMLScript is based on ECMAScript as mentioned earlier, there are differences between WMLScript and ECMAScript. First, like JavaScript, ECMAScript is not encoded in a binary form while WMLScript has to be. Second, to form WMLScript, many advanced features of the ECMAScript language have been dropped to make WMLScript smaller and easier to compile into binary WMLScript code.

WMLScript syntactically resembles C language. It has basic types, variables, expressions, and statements. Unlike C, WMLScript cannot be used to write stand-alone applications. There is no built-in support for reading and writing files. Because it is an interpreted language, scripts or functions can run only in the presence of an interpreter, which is supplied as part of the WAP user agent. WMLScript is a weakly typed and object-based language, in which variables must be declared before they can be used in expression. In WMLScript, there is no main program or routine. Functions are created to perform specific tasks and they are invoked through a WML call. When a WMLScript function is invoked, the WAP gateway accesses the source code, compiles it into binary WMLScript code, and then sends the execution function to the WAP user agent. WMLScript code is written in normal text files with the file extension “wmls.”

Each WMLScript file contains at least one function. Each function is composed of statements that perform the appropriate processing. The structure of a WMLScript function is as follows:

```
extern function function_xyz (parameter list)
{
  // start of the statements
  statement_1;
  statement_2;
  statement_n;
}
// end of the statements
```

With this structure and the file extension “xwmls,” a simple WMLScript example to set a day of the week, which is included in the file named “setday.xwmls,” is listed as follows:

```
extern function SetDay(givenDay)
{
  if (givenDay > 0 && givenDay <=7) {
    var newDay=givenDay;
  }
  else {
    newDay=1;
  }
  return newDay;
}
```

To invoke a WMLScript function, a reference to the WMLScript function must be included in a WML document. The call will be routed from the WAP browser

through the WAP gateway to the server. The server then sends the binary WMLScript code to the WAP browser. The WAP browser has an interpreter, which is able to execute WMLScript programs in their binary format. Using our example, the reference to the WMLScript can be as simple as follows:

```
<do type="ACCEPT" label="Set Day">
  <!--Calling the WMLScript function: -->
  <go href="setday.xmls#SetDay($(givenDay))"/>
</do>
```

### ***Wireless Telephony Application Interface and Wireless Telephony Applications***

One of the major mobile services is voice. How can we set up a call or receive an incoming call using a WAP-enabled mobile device? This is the problem that wireless telephony application interface (WTAI) addresses. WTAI is designed to allow wireless network operators access the telephony features of WAP device. Through either a WML deck/card or WMLScript, using the WTAI function libraries, a mobile phone call can be set up and an incoming call can be received. In addition, text messages can be sent or received, and phonebook entries can be manipulated on the WAP device.

Wireless telephony application (WTA) is a collection of telephony-specific extensions for call and feature control mechanisms that make advanced mobile network services available to the mobile users. It provides a bridge between wireless telephony and data. The WTA applications can use the privileged WTAI.

From the architecture point of view, a WTA server communicates with the WAP gateway to deliver and manage telephony services; on the client side, there is a WTA framework, which has three components as follows:

1. *User agent*. This agent supports the WTAI libraries, renders WML, and executes WMLScripts.
2. *Repository*. It provides persistent client-side storage for wireless telephony applications.
3. *Event handling*. This deals with incoming-call and call-connected events to be delivered to a wireless telephony application for processing, which may also invoke WMLScript library interfaces to initiate and control telephony operations.

Wireless telephony supports in WAP make WAP suitable for creating mobile applications through voice services. The compact form, encryption, and error handling capabilities of WAP enable critical wireless payment transactions.

### ***WBXML***

WAP binary XML content format (WBXML) is defined in the binary XML content format specification in the WAP standard set. This format is a compact binary representation of the XML. The main purpose is to reduce the transmission size of XML documents on narrowband communication channels.

A binary XML document is composed of a sequence of elements and each element may have zero or more attributes. The element structure of XML is preserved while the format encodes the parsed physical form of an XML document.

This allows user agents to skip elements and data that are not understood. In terms of encoding, a tokenized structure is used to encode an XML document. The network byte order is big-endian, that is, the most significant byte is transmitted first. Within a byte, bit-order is also big-endian, namely, the most significant bit first.

#### 4.2.3 Wireless Session Protocol

WSP is a protocol family in the WAP architecture, which provides the WAP application layer with a consistent interface for session services. WSP establishes a session between the client and the WAP gateway to provide content transfer: the client makes a request, and then the server answers with a reply through the WAP gateway. WSP supports the efficient operation of a WAP microbrowser running on the client device with limited capacity and communicating over a low-bandwidth wireless network. The WSP browsing applications are based on the HTTP 1.1 standard, and incorporated with additional features that are not included in the HTTP protocol, for example, the connection to the server will not be lost when a mobile user is moving, resulting in a change from one base station to another. The other additional features that WSP supports include:

- *Binary encoding.* Given the low bandwidth of the wireless network, the efficient binary encoding of the content to be transferred is necessary for mobile Internet applications.
- *Data push functionality.* Data push functionality is not supported in the HTTP protocol. A push is what is performed when a WSP server transfers the data to a mobile client without a preceding request from the client. WSP supports three push mechanisms for data transfer, namely, a confirmed data push within an existing session context, a non-confirmed data push within an existing session context, and a nonconfirmed data push without an existing session context.
- *Capability negotiation:* Mobile clients and servers can negotiate various parameters for the session establishments, for example, maximum outstanding requests and protocol options.
- *Session suspend/resume.* It allows a mobile user to switch off and on the mobile device and to continue operation at the exact point where the device was switched off.

WSP offers two different services, namely, the connection-oriented service and the connectionless service. The connection-oriented service has the full capabilities of WSP. It operates on top of the wireless transaction protocol (WTP), supports session establishment, method invocation, push messages, suspend, resume and session termination. The connectionless service is suitable for those situations where high reliability is not required, or the overhead of session establishment and release can be avoided. It supports only basic request-reply and push, and does not rely on WTP.



#### 4.2.4 Wireless Transaction Protocol

The wireless transaction protocol operates on top of a secure or insecure datagram service. WTP introduces the notion of a transaction that is defined as a request with its response. This transaction model is well suited for Web content requests and responses. It does not handle stream-based applications (such as telnet) well.

WTP is responsible for delivering the improved reliability over datagram service between the mobile device and the server by transmitting acknowledge messages to confirm the receipt of data and by retransmitting data that have not been acknowledged within a suitable timeout period. WTP supports an abort function through a primitive error handling. If an error occurs, such as the connection being broken down, the transaction is aborted.

WTP is message oriented and it provides three different types of transaction services, namely, unreliable one-way, reliable one-way, and reliable two-way. The transaction type is set by the initiator and is contained in the service request message sent to the responder. The unreliable one-way transactions are stateless and cannot be aborted. The responder does not acknowledge the message from the initiator. The reliable one-way transactions provide a reliable datagram service that enables the applications to provide reliable push service. The reliable two-way transactions provide the reliable request/response transaction services.

#### 4.2.5 Wireless Transport Layer Security

The wireless transport layer security (WTLS) protocol is a security protocol based on the *transport layer security protocol* (TLS) [10] (see Sect. 4.5). TLS is a derivative of the *secure sockets layer* (SSL), a widely used security protocol for Internet applications and payment over the Internet. WTLS has been optimized for the wireless communication environment. It operates above the transport protocol layer.

WTLS is flexible due to its modular design. Depending on the required security level, we can decide whether WTLS is to be used or not. WTLS provides data integrity, data confidentiality, authentication, and denial-of-service protection. Data integrity is to ensure that data sent between a mobile station and a wireless application server are unchanged and uncorrupted. Data confidentiality is to ensure that data transmitted between the mobile station and the wireless application server are private to the sender and the receiver, and one not going to be understood by any hackers. Authentication is to check the identity of the mobile station and the wireless application server. Denial-of-service protection is to prevent the upper protocol layers from the denial-of-service attacks by detecting and rejecting data that are replayed or not successfully verified.

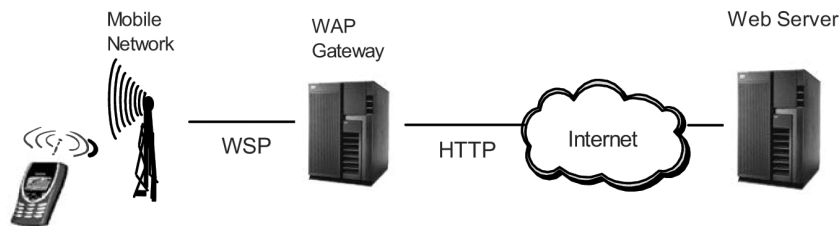


#### 4.2.6 Wireless Datagram Protocol

The wireless datagram protocol (WDP) in the WAP architecture specifies how different existing bearer services should be used to provide a consistent service to the upper layers. WDP is used to hide the differences among the underlying bearer networks. WDP layer operates above the bearer services and provides a consistent interface to the WTLS layer.

Different bearers have different characteristics. The bearer services include short message, circuit-switched data, and packet data services. Since WAP is designed to operate over the bearer services, and since the bearers offer different types of quality of service with respect to throughput, error rate, and delays, the WDP is designed to adapt the transport layer to specific features of the underlying bearers. The adaptation results in a family of protocols in the WDP layer, dealing with each supported bearer network protocol. When a message is transmitted through WAP stack, depending on the underlying bearer network, a different WDP protocol may be used. For example, for an IP bearer, the user datagram protocol (UDP) must be adopted as the WDP protocol, and for a short message service (SMS) bearer, the use of the source and destination port numbers becomes mandatory.

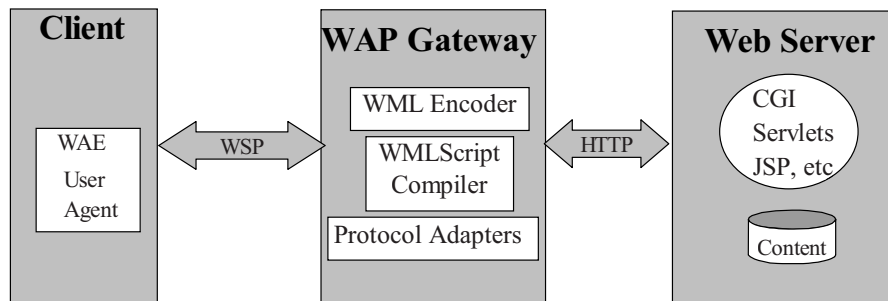
#### 4.2.7 Gateway



**Fig. 4.3.** WAP gateway

A WAP gateway (shown in Fig. 4.3) is a proxy server that sits between the mobile network and the Internet. The purpose of this proxy server is to translate between HTTP and WSP. The reason for the translation is that the Web server connected to the Internet understands only the HTTP protocol, while the WAP-enabled mobile client understands only the WSP. The WAP gateway also converts an HTML file into a WML document that is designed for small-screen devices. In addition, the WAP gateway compiles the WML page into binary WML, which is more suitable for the mobile client. The WAP gateway is transparent to both the mobile client and the Web server.

Fig. 4.4 shows the WAP model using the WAP gateway. How the WAP gateway processes a typical request for a document can be illustrated as follows:



**Fig. 4.4.** WAP model

1. The mobile user makes a request for a specific document using the WAP phone.
2. The WAE user agent on the WAP phone encodes the request and sends it to the WAP gateway.
3. The WAP gateway decodes and parses the encoded request.
4. The WAP gateway sends an HTTP request for the document.
5. The Web server answers with a response to the WAP gateway.
6. The WAP gateway parses and encodes the response.
7. If the content type is WML, then the gateway compiles it into binary WML.
8. The WAP gateway sends the encoded response to the WAP phone.
9. The WAE user agent on the WAP phone interprets and presents the document to the mobile user.

### 4.3 Wireless Application Security

Wireless application security is becoming increasingly important as transaction-based mobile commerce applications (such as mobile payment, banking, and buying stock via cellular phones or other handheld devices) take off.

The basic security needs for mobile commerce are similar to those for electronic commerce over the wired Internet, such as authentication, confidentiality, nonrepudiation, and data integrity. However, implementing them in the wireless world is more difficult than in the wired world. This is simply because of the limitations that wireless have, including limited bandwidth, high latency, and unstable connections. In addition, the limited battery and processing power that the wireless devices have also make the sophisticated security algorithms difficult to run on these devices.

As discussed in Sect. 4.2, WAP specifies an SSL-like security protocol, namely, wireless transport layer security (WTLS). However, there are some drawbacks in WTLS. First, WTLS provides only security protection from the mobile

client to the WAP gateway where the wireless communication ends. In the wired Internet environment, when a Web client (Web browser) starts an SSL session with Web server, the Web client and Web server are communicated directly, and the end-to-end security protection is provided through the SSL session. This means that when one sends a credit card number over SSL, only the receiving Web server will be able to receive it. The situation is different in the WTLS. The credit card number will be securely protected between the mobile device and the WAP gateway. It will be in the clear form at the WAP gateway. Then, an SSL session will be established between the WAP gateway and the Web server for securely transmitting the credit card number over the Internet. This means that there is no end-to-end security protection for the wireless transactions since there is a potential security hole in the WAP gateway. Second, the CCITT X509 certificate is too large for the mobile phones, and the limitations of the processing power and battery for the wireless devices make it difficult to perform the sophisticated computation of the public-key encryption. In summary, WAP security has two issues: (1) there is no end-to-end security protection and (2) there is a lack of certificates for mobile devices.

Research is being done on these two security issues. As a result, simplified certificates have been defined for mobile devices. The research on how to use currently available mobile devices to perform the computation of public-key encryption is ongoing. For example, elliptic curve cryptography (ECC) requires far fewer resources and it looks very promising for wide deployment to CPU-starved wireless devices.

## 4.4 Summary

The convergence of wireless technologies and the e-business over the Internet has led to emerging and fast growth of wireless e-business, including mobile commerce. As a result, wireless e-business has attracted increasing attention of academic researchers and business leaders. Being able to conduct e-business anywhere and anytime is becoming a reality. However, because of the limitations that wireless has, conducting e-business in the wireless world is more difficult than in the wired world. Understanding the wireless application protocol that the wireless e-business relies on is important for developing and deploying wireless e-business. In this chapter, our discussion was focused on wireless application protocol and related wireless security.

## 4.5 Appendix

### 4.5.1 Overview of the Transport Layer Security

The transport layer security (TLS) [10] is a protocol that provides privacy and data integrity between two communicating applications. The TLS is application protocol

independent, that is, higher-level protocols can layer on top of the TLS protocol transparently. The TLS protocol is composed of two layers:

1. *TLS record protocol*. This protocol provides connection security and is used for encapsulation of various higher-level protocols, such as the TLS handshake protocol discussed here. It has the following two basic properties
  - *The connection is private*. Data encryption is used for ensuring the communication privacy and is based on symmetric cryptographic algorithms, such as DES or RC4. The keys for symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (e.g., the TLS handshake protocol). The record protocol can also be used without encryption.
  - *The connection is reliable*. A message integrity check based on a keyed MAC is used for protecting message transport. Secure hash functions, such as SHA and MD5, are used for MAC computations. In such cases, another protocol uses the record protocol and negotiates security parameters, and the record protocol can operate without a MAC.
2. *TLS handshake protocol*. This protocol allows the server and client to authenticate each other, and negotiate an encryption algorithm and cryptographic keys. It has the following three basic properties
  - The authentication between the server and client can be based on a public-key cryptographic algorithm, such as RSA or DSS. Although the authentication can be mutual, the mutual authentication is optional. Generally speaking, one-way authentication is required.
  - It is secure for the negotiation of a shared secret between the server and client.
  - The negotiation is reliable.

Because the TSL is a derivative of SSL, the actual handshake exchanges are similar to that of SSL. Description of the main SSL exchanges can be found later in this book.

### Acknowledgments

This work is supported in part by NSFC grant 90304008 from the Nature Science Foundation of China and the Doctoral Program Foundation grant 2004071001 from the Ministry of Education of China.

### References

1. WAP. <http://www.ini.cmu.edu/netbil>.
2. Wireless Application Protocol Forum Ltd (1999) Official Wireless Application Protocol. Wiley, New York.

3. S. Mann, S. Sbihi (2000) The Wireless Application Protocol. Wiley, New York.
4. S. Singhal, et al. (2001) The Wireless Application Protocol. Addison-Wesley, New York.
5. J. Schiller (2000) Mobile Communications. Addison-Wesley, New York.
6. U. Hansmann, et al. (2001) Pervasive Computing Handbook. Springer, Berlin Heidelberg New York.
7. C. Sharma (2001) Wireless Internet Enterprise Applications. Wiley, New York.
8. Y.B. Lin, I. Chlamtac (2001) Wireless and Mobile Network Architectures. Wiley, New York.
9. Dornan (2001) The Essential Guide to Wireless Communications Applications. Prentice-Hall, New York.
10. T. Dierks, C. Allen (1999) The TLS Protocol Version 1.0. <http://www.ietf.org/rfc/rfc2246.txt>.



<http://www.springer.com/978-3-540-30449-4>

Enabling Technologies for Wireless E-Business

Kou, W.; Yesha, Y. (Eds.)

2006, X, 387 p. 141 illus., Hardcover

ISBN: 978-3-540-30449-4