Winwap Technologies Oy

Client WAP Stack Library

Technical information

WAP stack version 2.6
WAP specification version: 2.0
Document dated: 5 Jan 2007

Notice of Confidentiality

Revision history

Date	Author	Description
9-Feb-2004	S Markelov	Initial versions of the document.
1-Jun-2004	S Markelov	New section "Push OTA and Push OTA-WSP protocol implementation".
22-Jun-2004	S Markelov	Gray colored table headers and PDF hyperlinks in the contents.
7-Jul-2004	S Markelov	Updated the "Minimum requirements" section.
14-Oct-2004	S Markelov	Updated to WAP Stack version 2.6 (with WP-HTTP protocol).
9-Nov-2004	S Markelov	New platform: MontaVista 2.4.20 on MIPS based boards.
25-Nov-2004	Maria Sandell	English spell checked.
8-Dec-2005	S Markelov	RSA key exchange algorithm is supported.

Introduction

This document lists a set of functions that are implemented in the WAP Stack Library.

Contents

1 Summary

1.1 Supported platforms

This table lists a set of platforms that the WAP Stack implementation is available for. The table also lists a set of compilers that can be used for compilation of the WAP Stack Library.

Platform	Compiler used	Processor (CPU)
Windows 95/98/ME	MS Visual C++ 6.0	Intel x486
Windows NT4/2000/XP	MS Visual C++ 6.0	Intel x486
Windows 2000/XP	MS Visual C++ 7.0	Intel Pentium II
Windows CE 2.11/3.0	MS eMbedded Visual C++ 3.0	ARM, SH3, MIPS
Pocket PC 3.0	MS eMbedded Visual C++ 3.0	ARM, SH3, SH4, MIPS
Pocket PC 2002	MS eMbedded Visual C++ 3.0	ARM
Windows CE 4.0	MS eMbedded Visual C++ 4.0	ARM, ARMV4I, ARMV4T
Pocket PC 2003	MS eMbedded Visual C++ 4.0	ARMV4
RedHat Linux 7.3	GNU GCC 2.96	Intel x386
RedHat Linux 9	GNU GCC 3.2	Intel x386
MontaVista 2.4.20	GNU GCC 2.96	MIPS
SUSE Linux 8.1	GNU GCC 3.2	Intel x386
Solaris 5.8	Sun ONE Studio 8, C/C++ 5.5	SUNW, UltraSPARC-II
HP-UX B.11.11	HP ANSI C++ B3910B A.03.45	2.0 PA8500

The source codes are almost only using functions from the POSIX standard. It requires POSIX threads, timers (sleep functions) and sockets. The source code is written using clear ANSI C and standard C++ without any compiler specific extensions. It is therefore possible to port the WAP Stack Library source code to any platform with an available C/C++ compiler.

1.2 WAP Stack features

This table lists a set of general features that are implemented in the WAP Stack Library and are or interest for a developer of client application. Detailed descriptions of the different implementations of the WAP protocols are available in the following sections.

Component	Feature		Status
Content Type	WML 1.1 / 1.2 / 1.2.1 text and binary		yes
	WMLScript 1.1 / 1.2 / 1.2.1 text and binary		yes
	WML Char-sets	US ASCII	yes
		UTF-8	yes
		UCS-2	yes
		ISO 8859-1 (Latin)	yes
		ISO 8859-2 to 10 (other European)	yes
		ISO 8859-13 to 15 (other European)	yes
		SHIFT_JIS (Japanese)	yes

Component	Feature		Status
WSP 1.0 (5 July 2001)	Connection-oriented session establishment		yes
	Connectionless session establishment		yes
	Header encoding / decoding	Version 1.2	yes
		Version 1.3	yes
		Version 1.4	yes
	Session Management	Connect/Disconnect	yes
		Redirect	yes
	Method Invocation Facility	Get, Post	yes
	Push & Confirmed Push Facilities		yes
	Capability Negotiation support		yes
WTP 0.0 (10-Jul-2001)	Segmentation and Re-assembly (SAR)		yes
	Run-time tuning of SAR parameters		yes
	Initiator	Class 0[1]	yes
		Class 1[1]	yes
		Class 2	yes
	Responder	Class 0	yes
		Class 1	yes
		Class 2	no
WTLS (06-April-2001)[2]	Key exchange algorithms	Diffie-Hellman 768	yes
		Diffie-Hellman 512	yes
	Bulk encryption algorithms	DES (56- and 40-bits effective keys)	yes
		3DES EDE (168-bits effective key)	yes
		NULL	yes
	MAC		yes
	Session Management	Full handshake (anonymous handshake)	yes
		Handshake reliability over datagrams	yes
	API for using external WTLS library		no
WP-HTTS (29-Mar-2001)	HTTP Client	GET	yes
		POST	yes
		CONNECT	yes
	HTTP Server	GET	yes
		HEAD	yes
		POST	yes
		OPTIONS	yes
WAP 2.0 / MMS	MMS Sending		yes
	MMS Retrieval		yes

Class 0 and Class 1 WTP Initiator transactions are implemented but currently not provided by

[1]Not available via API
[2]Not implemented under Windows CE 2.11/3.0 and Pocket PC 3.0

the API.

1.3 Minimum requirements

The table below describes the approximate system requirements for the WAP Stack Library. The "Source" column indicates the approximate total size of the C and C++ code. The "Size" column indicates the size of the WAP Stack shared library, which is a DLL for Windows, .so file for Linux and Solaris and .sl file for HP-UX. The "RAM" column indicates the approximate RAM usage with a sample program, making several GET-requests to different WAP resources.

Platform	Source	Size	w/o HTTP	w/o HTTP, WTLS	RAM
Windows 95/98/ME/NT4/XP/2000	820 KB	175 KB	135 KB	–	2.2 MB
Windows CE 2.11/3.0 (ARM)	552 KB	–	120 KB	–	620 KB
Pocket PC 3.0/2000 (ARM)	552 KB	–	120 KB	–	620 KB
Windows CE 4.0/.NET (ARM)	820 KB	–	378 KB	–	1.2 MB
SuSe Linux 8.1	820 KB	1172 KB	993 KB	200 KB	1.4 MB
MontaVista Linux 2.4.20	–	–	–	346 KB[3]	780 KB
Solaris	820 KB	–	1187 KB	–	–
HP-UX	820 KB	–	1624 KB	–	–

[3]For the target board the standard C++ library implementation is available only for the static linkage

2 WSP protocol implementation

2.1 WSP Client/Server Mode

This table lists a set of modes that are implemented in the WSP protocol. The table structure is similar to the table "D.1 Client/Server Mode" in "Appendix D Static Conformance Requirement" of the WAP-230-WSP-20010705-a specification. The "Status" column indicates if the mode is implemented or not.

Item	Function	Reference	Status
WSP-C-001	Device Mode	Section 6, 7 and 8	✓
WSP-CL-C-001	Connectionless	Section 6, 7 and 8	✓
WSP-CO-C-001	Connection-Oriented	Section 6, 7 and 8	✓

2.2 WSP Connection-Oriented Client

This table lists a set of functions that are implemented in the WSP Connection-Oriented Client. The table structure is similar to the table "D.2 Connection-Oriented Client" in "Appendix D Static Conformance Requirement" of the WAP-230-WSP-20010705-a specification. The "Status" column indicates if the function is implemented or not.

Item	Function	Reference	Status
WSP-CO-C-002	Connect PDU	6.3.3.1 6.3.4 7.1.2.1 7.1.5 7.1.6.1 8.2.2.1	✓
WSP-CO-C-003	ConnectReply PDU	7.1.2.1 7.1.5 7.1.6.1 8.2.2.2	✓
WSP-CO-C-004	Redirect PDU	7.1.2.1 7.1.6.1 8.2.2.3	✓
WSP-CO-C-005	Capability Negotiation - Connect PDU	6.3.2 6.3.3.1 7.1.5 7.1.6.1 8.2.2.1 8.3	✓
WSP-CO-C-006	Capability Negotiation - ConnectReply PDU	7.1.5 7.1.6.1 8.2.2.2 8.3	✓

Item	Function	Reference	Status
WSP-CO-C-007	Disconnect PDU	6.3.3.2 7.1.2.1 7.1.5 7.1.6.1 8.2.2.4	✓
WSP-CO-C-008	Suspend PDU	6.3.3.3 7.1.2.2 7.1.5 7.1.6.1 8.2.5.1	✓
WSP-CO-C-009	Resume PDU	6.3.3.4 7.1.2.2 7.1.5 7.1.6.1 8.2.5.2	✓
WSP-CO-C-010	Push PDU	6.3.3.9 7.1.2.4 8.2.4.1	✓
WSP-CO-C-011	ConfirmedPush PDU	6.3.3.10–11 6.3.4 7.1.2.5 7.1.5 7.1.6.3 8.2.4.1	✓
WSP-CO-C-012	Ack. Headers	6.3.3.7 7.1.6.2–3 8.2.4.2	✓
WSP-CO-C-013	Extended Methods	6.3.2.2 8.3.2.4	
WSP-CO-C-014	Default Header Code Page Encoding	8.4 Table 39	✓
WSP-CO-C-015	Extended Header Code Page Encoding	8.3.2.5	
WSP-CO-C-016	Aliases	6.3.2.2 8.3.2.6	
WSP-CO-C-017	Method GET - Get PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.1	✓
WSP-CO-C-018	Method GET - Reply PDU	6.3.3.6–8	✓

Item	Function	Reference	Status
		6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.3	
WSP-CO-C-019	Method GET - Data Fragment PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.4	✓
WSP-CO-C-020	Method POST - Post PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.2	✓
WSP-CO-C-021	Method POST - Reply PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.3	✓
WSP-CO-C-022	Method POST - Data Fragment PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.4	✓
WSP-CO-C-023	Method DELETE - Get PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.1	
WSP-CO-C-024	Method DELETE - Reply PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.3	

Item	Function	Reference	Status
WSP-CO-C-025	Method HEAD - Get PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.1	
WSP-CO-C-026	Method HEAD - Reply PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.3	
WSP-CO-C-027	Method OPTIONS - Get PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.1	
WSP-CO-C-028	Method OPTIONS - Reply PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.3	
WSP-CO-C-029	Method TRACE - Get PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.1	
WSP-CO-C-030	Method TRACE - Reply PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.3	
WSP-CO-C-031	Method PUT - Post PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2	

Item	Function	Reference	Status
		8.2.3.2	
WSP-CO-C-032	Method PUT - Reply PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.3	
WSP-CO-C-033	Method PUT - Data Fragment PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.4	
WSP-CO-C-034	Multipart Data - Post PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.2	✓
WSP-CO-C-035	Multipart Data - Reply PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.3	✓
WSP-CO-C-036	Multipart Data - Fragment PDU	6.3.3.6–8 6.3.4 7.1.2.3 7.1.5 7.1.6.2 8.2.3.4	✓
WSP-CO-C-037	Method Abort	6.3.3.2–3 6.3.3.8	✓
WSP-CO-C-038	Push Abort	6.3.3.10–11	✓
WSP-CO-C-039	Encoding Version Framework	8.4.1 8.4.2.70	✓

2.3 WSP Connectionless Client

This table lists a set of functions that are implemented in the WSP Connectionless Client. The table structure is similar to the table "D.3 Connectionless Client" in "Appendix D Static Conformance Requirement" of the WAP-230-WSP-20010705-a specification. The "Status" column indicates if

the function is implemented or not.

Item	Function	Reference	Status
WSP-CL-C-002	Push PDU	6.4.2.3 6.3.4 7.2 8.2.4.1	✓
WSP-CL-C-003	Header Encoding Default page	8.4 Table 39	✓
WSP-CL-C-004	Method GET - Get PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.1	✓
WSP-CL-C-005	Method GET - Reply PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.3	✓
WSP-CL-C-006	Method POST - Post PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.2	✓
WSP-CL-C-007	Method POST - Reply PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.3	✓
WSP-CL-C-008	Method DELETE - Get PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.1	
WSP-CL-C-009	Method DELETE - Reply PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.3	
WSP-CL-C-010	Method HEAD - Get PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.1	
WSP-CL-C-011	Method HEAD - Reply PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.3	
WSP-CL-C-012	Method OPTIONS - Get PDU	6.4.2.1–2 6.4.3	

Item	Function	Reference	Status
		7.2 8.2.3.1	
WSP-CL-C-013	Method OPTIONS - Reply PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.3	
WSP-CL-C-014	Method TRACE - Get PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.1	
WSP-CL-C-015	Method TRACE - Reply PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.3	
WSP-CL-C-016	Method PUT - Post PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.2	
WSP-CL-C-017	Method PUT - Reply PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.3	
WSP-CL-C-018	Multipart Data - Post PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.2	✓
WSP-CL-C-019	Multipart Data - Reply PDU	6.4.2.1–2 6.4.3 7.2 8.2.3.3	✓
WSP-CL-C-020	Encoding Version Framework	8.4.1 8.4.2.70	✓

3 WTP protocol implementation

3.1 WTP Client

This table lists a set of functions that are implemented in the WTP. The table structure is similar to the table in "Appendix C. Static Conformance Requirements" of the WAP-224-WTP-20010710-a specification. The "Status" column indicates if the function is implemented or not.

Item	Function	Reference	Status
WTP-C-001	Transaction Class 0 Initiator	6.1.3	✓
WTP-C-002	Transaction Class 0 Responder	6.1.3	✓
WTP-C-003	Transaction Class 1 Initiator	6.2.4	✓
WTP-C-004	Transaction Class 1 Responder	6.2.4	✓
WTP-C-005	Transaction Class 2 Initiator	6.3.4	✓
WTP-C-006	Transaction Class 2 Responder	6.3.4	
WTP-C-007	User Acknowledgment	7.3	✓
WTP-C-008	Concatenation	4.1, 7.5	✓
WTP-C-009	Separation	7.5	✓
WTP-C-010	Retransmission until Acknowledgment	7.2	✓
WTP-C-011	Transaction Abort	4.6, 7.7, 7.12	✓
WTP-C-012	Error Handling	7.12	✓
WTP-C-013	Information in Last Acknowledgment	7.4, 7.10	
WTP-C-014	Asynchronous Transactions	7.6	✓
WTP-C-015	Initiator response to TID Verification	7.1.5.2	✓
WTP-C-016	Initiation of TID Verification by Responder	7.1.5.2, 7.8.1	✓
WTP-C-017	Error Transport Information Item	7.10, 8.4.2	
WTP-C-018	Info Transport Information Item	7.10, 8.4.3	
WTP-C-019	Option Transport Information Item	7.10, 8.4.4	
WTP-C-020	PSN Transport Information Item	7.10, 8.4.5	✓
WTP-C-021	Segmentation and Re -assembly with Selective Retransmission and Packet Groups	7.14	✓
WTP-C-022	Reliable transaction	7	✓
WTP-C-023	Extended Segmentation and Re-assembly with Selective Retransmission and Packet Groups	7.15	
WTP-C-024	Frame Boundary Transport Information Item	7.15.2, 8.4.7	
WTP-C-025	SDU Boundary Transport Information Item	7.15.2, 8.4.6	
WTP-C-026	Support sliding window with ESAR	7.15.3, 7.15.4	

4 WTLS protocol implementation

4.1 WTLS Class

The WTLS implementation supports class 1 mandatory features. The current version of the WTLS specification covers all features in class 1.

Feature	Availability
Public-key exchange	✓
Server certificates	
Client certificates	
Shared-secret handshake	
Compression	-
Encryption	✓
MAC	✓
Smart card interface	-

4.2 WTLS protocol requirements

The common requirements set by wireless mobile networks taken into account by the WAP Stack are described below.

Datagram transport protocol Both datagram and connection oriented transport layer protocols are supported. The WAP Stack is able to cope with lost, duplicated, or out of order datagrams without breaking the connection state.

Slow interactions The WAP Stack takes into account that round-trip times with some bearers (eg, SMS) can be long. For example, sending a query and receiving a response might require more than 10 seconds. This is taken into account in the protocol design.

Low transfer rate The slown speed of some bearers is a major constraint. Therefore, the amount of overhead is kept to a minimum. For example, with SMS the effective transfer rate may be lower than 100 bit/s.

Limited processing power The processing power of many mobile terminals is quite limited. This is taken into account when implementations of cryptographic algorithm are chosen.

Limited memory capacity The memory capacity of most mobile terminals is very modest. Therefore, the number of cryptographic algorithms used is minimal and small-sized algorithms are chosen. The RAM requirements are set as low as possible.

Restrictions on exporting and using cryptography International restrictions and rules for using, exporting, and importing cryptography are taken into account. This means that the WAP Stack tries to achieve the best permitted security level according to the legislation of each area. For example, in many cases, strong authentication can be used although strong encryption is prohibited.

4.3 WTLS Client

This table lists a set of functions that are implemented in the WTLS protocol. The table structure is similar to the table "E.2 WTLS Client Options" in "Appendix E Static Conformance Requirement" of the WAP-261-WTLS-20010406-a specification. The "Status" column indicates if the function is implemented or not.

Item	Function	Subfunction	Reference	Status
WTLS-C-001	Session management	Full handshake (eg, needed for the anonymous handshake)	10.3	✓
WTLS-C-002		Abbreviated handshake (ie, resume)	10.3	
WTLS-C-003		Optimized public key handshake	10.3	
WTLS-C-004		Session sharing (multiple connections)	11.1.4	
WTLS-C-005		Record concatenation for handshake messages	10.4	✓
WTLS-C-006		Handshake reliability over datagrams	10.4	✓
WTLS-C-007		Start negotiation after a cleartext Hello Request	10.5.1.1	
WTLS-C-010	Record protocol	Explicit sequence numbering	9.2.3.1	✓
WTLS-C-011		Implicit sequence numbering	9.2.3.1	✓
WTLS-C-012		Duplicate removal	9.2.3.1	✓
WTLS-C-013		Key refresh	Appendix B.3	✓
WTLS-C-020	Alerting	Critical alerts (close connection)	10.2	✓
WTLS-C-021		Fatal alerts (close connection, invalidate session if not in cleartext)	10.2	✓
WTLS-C-022		Checking of checksums	10.2	✓
WTLS-C-025	Change Cipher Spec		10.1	✓
WTLS-C-026	Application Data		9.2	✓
WTLS-C-030	Anonymous handshaking options; at least one supported.		Appendix A	✓
WTLS-C-031		DH_ANON	Appendix A	
WTLS-C-032		DH_ANON_768	Appendix A	✓
WTLS-C-033		DH_ANON_512	Appendix A	✓
WTLS-C-034		RSA_ANON	Appendix A	✓
WTLS-C-035		RSA_ANON_768	Appendix A	✓
WTLS-C-036		RSA_ANON_512	Appendix A	✓
WTLS-C-037		ECDH_ANON	Appendix A	
WTLS-C-038		ECDH_ANON_131	Appendix A	
WTLS-C-049		ECDH_ANON_113	Appendix A	
WTLS-C-060	Non-anonymous (server authenticated) handshake options; at least one supported.		Appendix A	
WTLS-C-061		RSA	Appendix A	
WTLS-C-062		RSA_768	Appendix A	
WTLS-C-063		RSA_512	Appendix A	
WTLS-C-064		ECDH_ECDSA	Appendix A	

Item	Function	Subfunction	Reference	Status
WTLS-C-070	Client authentication options; at least one supported.		Appendix A	
WTLS-C-071		RSA	Appendix A	
WTLS-C-072		ECDH_ECDSA	Appendix A	
WTLS-C-080	Shared secret handshake		Appendix A	
WTLS-C-090	NULL key exchange		Appendix A	✓
WTLS-C-100	Data encryption options; at least one supported.		Appendix A	✓
WTLS-C-101		RC5_CBC	Appendix A	✓
WTLS-C-102		RC5_CBC_56	Appendix A	✓
WTLS-C-103		DES_CBC	Appendix A	✓
WTLS-C-104		3DES_CBC_EDE	Appendix A	✓
WTLS-C-105		IDEA_CBC	Appendix A	✓
WTLS-C-106		IDEA_CBC_56	Appendix A	✓
WTLS-C-107		RC5_CBC_64	Appendix A	✓
WTLS-C-108		IDEA_CBC_64	Appendix A	✓
WTLS-C-120	NULL encryption		Appendix A	✓
WTLS-C-130	MAC options; at least one supported.		Appendix A	✓
WTLS-C-131		SHA	Appendix A	✓
WTLS-C-132		SHA_80	Appendix A	✓
WTLS-C-133		SHA_40	Appendix A	✓
WTLS-C-134		(Algorithm removed)	N/A	✓
WTLS-C-135		MD5	Appendix A	✓
WTLS-C-136		MD5_80	Appendix A	✓
WTLS-C-137		MD5_40	Appendix A	✓
WTLS-C-140		NULL MAC (SHA_0)	Appendix A	✓
WTLS-C-141	NULL compression		Appendix A	✓
WTLS-C-151	Predefined Diffie-Hellman parameters	Parameters 1	Appendix A	✓
WTLS-C-152		Parameters 2	Appendix A	✓
WTLS-C-165	ECC basic curves; if ECC is used, at least one MUST be supported. Verification SHOULD be supported with all basic curves that have field size not less than 160 bits	Curve 5 (163 bits)	Appendix A	
WTLS-C-167		Curve 7 (160 bits)	Appendix A	
WTLS-C-161	ECC non-basic curves	Curve 1 (113 bits)	Appendix A	
WTLS-C-163		Curve 3 (163 bits)	Appendix A	
WTLS-C-164		Curve 4 (113 bits)	Appendix A	
WTLS-C-166		Curve 6 (112 bits)	Appendix A	
WTLS-C-168		Curve 8 (112 bits)	Appendix A	
WTLS-C-169		Curve 9 (160 bits)	Appendix A	
WTLS-C-170		Curve 10 (233 bits)	Appendix A	
WTLS-C-171		Curve 11 (233 bits)	Appendix A	
WTLS-C-172		Curve 12 (224 bits)	Appendix A	
WTLS-C-180	ECC point compression		11.1.3	

Item	Function	Subfunction	Reference	Status
WTLS-C-191	Verification of certificates; WTLS certificate verification MUST be supported if nonanonymous handshake is supported	WTLS certificate	10.5.2	
WTLS-C-192		X.509 certificate	10.5.2	
WTLS-C-193		X9.68 certificate	10.5.2	
WTLS-C-200	GMT UNIX time support		10.5.1	✓
WTLS-C-210	Reject non-root CA WTLS certificate if $T = ca$ is not present		10.5.2	
WTLS-C-220	Use of WIM [WAP WIM]		[WAP WIM]	
WTLS-C-230	Not accept NULL key exchange unless sent it		Appendix B.1	

5 WP-HTTP protocol implementation

5.1 WAP-Terminal

This table lists a set of functions that are implemented in the WP-HTTP protocol. The table structure is similar to the table in "Appendix C. Static Conformance Requirements" of the WAP-229-HTTP-20010329-a specification. The "Status" column indicates if the function is implemented or not.

Item	Function	Reference	Status
HTTP-C-C001	Support for HTTP Client	6.1.1	✓
HTTP-C-C002	Support for TLS	6.4.1	✓
HTTP-C-S001	Support for HTTP Server	6.1.2	✓
HTTP-C-C003	Support for GET Method	6.1.1	✓
HTTP-C-C004	Support for POST Method	6.1.1	✓
HTTP-C-C005	Support for CONNECT Method	6.1.1, 6.4.1	✓
HTTP-C-C006	Support for 'deflate' content decoding	6.3.1	✓
HTTP-C-S002	Support for GET	6.1.2	✓
HTTP-C-S003	Support for POST	6.1.2	✓
HTTP-C-S004	Support for HEAD	6.1.2	✓
HTTP-C-S005	Support for OPTIONS	6.1.2	✓
HTTP-C-S006	Support for 'deflate' content encoding	6.3.1	

6 Push OTA and Push OTA-WSP protocol implementation

6.1 Push OTA and Push WSP Client

This table lists a set of functions that are implemented in the Push OTA and Push OTA-WSP protocol. The table structure is similar to the table in "Appendix A. Static Conformance Requirements (Normative)" of the WAP-235-PushOTA-20010425-a specification. The "Status" column indicates whether the function is implemented or not.

Item	Function	Reference	Status
OTA-CL-C-001	Connectionless Push	5, 6.2.1	✓
OTA-CL-C-002	Non-secure Port for connectionless push	6.2.1	✓
OTA-CL-C-003	Secure Port for WTLS for connectionless push	6.2.1	
OTA-CO-C-001	Connection-oriented push	5	✓
OTA-CO-C-002	Connection-Oriented Push using OTA-WSP	6.2.2	✓
OTA-WSP-C-001	Connection-oriented Confirmed Push	6.2.2	✓
OTA-WSP-C-002	Connection-oriented Unconfirmed Push	6.2.2	✓
OTA-WSP-C-003	Use non-secure transport service	6.2.2	✓
OTA-WSP-C-004	Use secure transport service with WTLS	6.2.2	✓
OTA-WSP-C-005	SIA/SIR	8, 8.2, 8.4	
OTA-WSP-C-006	Application Addressing	6.2.3	
OTA-WSP-C-007	Application Dispatching	6.3.1	
OTA-WSP-C-008	Initiator Authentication	6.2.4	
OTA-WSP-C-009	Bearer Selection	6.2.6	
OTA-WSP-C-010	Bearer Control	6.2.6	
OTA-WSP-C-011	Security Considerations	8.3	