WAP WTAI

Version 08-Nov-1999

Wireless Application Protocol
Wireless Telephony Application Interface Specification

Disclaimer:

This document is subject to change without notice.

Contents

1 SCOPE 4

2 DOCUMENT STATUS 5

2.1 COPYRIGHT NOTICE 5
2.2 ERRATA 5
2.3 COMMENTS 5

3 REFERENCES 6

3.1 NORMATIVE REFERENCES 6
3.2 INFORMATIVE REFERENCES 6

4 DEFINITIONS AND ABBREVIATIONS 7

4.1 DEFINITIONS 7
4.2 ABBREVIATIONS 8

5 WTA BACKGROUND 9

5.1 WTAI LIBRARIES 9
5.2 EVENT HANDLING 9

6 WTA INTERFACE 10

6.1 WTAI FUNCTION LIBRARIES 10
6.2 WTAI API DELIMITERS 10
6.3 WTAI URI SCHEME 11
6.4 WTAI FUNCTION DEFINITION FORMAT 11
6.5 TELEPHONE NUMBERS 12

7 PUBLIC WTAI 13

7.1 MAKE CALL 13
7.2 SEND DTMF TONES 14
7.3 ADD PHONEBOOK ENTRY 15

8 NETWORK COMMON WTA 16

8.1 NETWORK EVENTS 16
8.2 VOICE CALL CONTROL 18
8.2.1 Setup Call 18
8.2.2 Accept Call 19
8.2.3 Release Call 20
8.2.4 Send DTMF Tones 20
8.2.5 Call Status 21
8.2.6 List Call 22
8.3 NETWORK TEXT 23
8.3.1 Send Text 23
8.3.2 Read Text 24
8.3.3 Remove Text 24
8.3.4 GetFieldValue 25
8.4 PHONEBOOK 27
8.4.1 Write Phonebook Entry 28
8.4.2 Read Phonebook Entry 29
8.4.3 Remove Phonebook Entry 30
8.4.4 GetFieldValue 30
8.4.5 Change Phonebook Entry 31

8.5 CALL LOGS 32
8.5.1 Last Dialled Numbers *32*
8.5.2 Missed Calls *33*
8.5.3 Received Calls *34*
8.5.4 GetFieldValue *35*
8.6 MISCELLANEOUS 36
8.6.1 Indication *36*
8.6.2 Terminate WTA User Agent *37*
8.6.3 Protect WTA User Agent Context *37*

APPENDIX A WTAI URI AND WMLSCRIPT FUNCTION LIBRARIES 38

APPENDIX B WTAI PREDEFINED ERROR CODES 40

APPENDIX C EXAMPLES USING WTAI 41

APPENDIX D PREDEFINED FIELD NAMES 43

APPENDIX E STATIC CONFORMANCE REQUIREMENTS 44

E.1 CLIENT FEATURES 44
E.2 SERVER FEATURES 47

1 Scope

Wireless Application Protocol (WAP) is a result of continuous work to define an industry wide specification for developing applications that operate over wireless communication networks. The scope for the WAP Forum is to define a set of specifications to be used by service applications. The wireless market is growing very quickly, and reaching new customers and services. To enable operators and manufacturers to meet the challenges in advanced services, differentiation and fast/flexible service creation WAP defines a set of protocols in transport, session and application layers. For additional information on the WAP architecture, refer to "*Wireless Application Protocol Architecture Specification"* [WAP].

This document outlines the extensions to the WAP Application Environment (WAE) to support Wireless Telephony Applications. The specifics of the Wireless Telephony Applications are introduced in the form of an interface. The acronym WTAI is used in the document to denote the Wireless Telephony Application Interface. For maximum benefit, the reader should be somewhat familiar with WML [WML] and WMLScript [WMLScript].

2 Document Status

This document is available online in the following formats:

- PDF format at http://www.wapforum.org/.

2.1 Copyright Notice

2.2 Errata

Known problems associated with this document are published at http://www.wapforum.org/

2.3 Comments

Comments regarding this document can be submitted to the WAP Forum in the manner published at http://www.wapforum.org/

3 References

The following section describes references relevant to this document.

3.1 Normative references

[ISO8601] “Data elements and interchange formats - Information interchange - Representation of dates and times”, International Organization For Standardization (ISO), 15-June-1988

“Data elements and interchange formats - Information interchange - Representation of dates and times, Technical Corrigendum 1”, International Organization For Standardization (ISO) -Technical Committee ISO/TC 154, 01-May-1991

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997. URL: ftp://ds.internic.net/rfc/rfc2119.txt

[RFC1630] "Uniform Resource Identifiers (URI)", T. Berners-Lee, et al., June 1994. URL: ftp://ds.internic.net/rfc/rfc1630.txt

[WAE] "Wireless Application Environment Specification", WAP Forum, 1998. URL: http://www.wapforum.org/

[WAP] "Wireless Application Protocol Architecture Specification", WAP Forum, 1998. URL: http://www.wapforum.org/

[WML] "Wireless Markup Language", WAP Forum, 1999. URL: http://www.wapforum.org/

[WMLScript] "WMLScript Language Specification", WAP Forum, 1999. URL: http://www.wapforum.org/

[WSP] "Wireless Session Protocol Specification", WAP Forum, 1999. URL: http://www.wapforum.org/

[WTA] "Wireless Telephony Application Specification", WAP Forum, 1999. URL: http://www.wapforum.org/

[XML] "Extensible Markup Language (XML), W3C Proposed Recommendation 10-February-1998, REC-xml-19980210", T. Bray, et al, February 10, 1998. URL: http://www.w3.org/TR/REC-xml

3.2 Informative references

[RFC1738] "Uniform Resource Locators (URL)", T. Berners-Lee, et al., December 1994. URL: ftp://ds.internic.net/rfc/rfc1738.txt

4 Definitions and abbreviations

The following section describes definitions and abbreviations common to this document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

4.1 Definitions

The following are terms and conventions used throughout this specification.

Card - a navigable part of a WML document (deck). May contain information to present on the screen, instructions for gathering user input, etc.

Client - a device (or application) that initiates a request for connection with a server.

Content - synonym for resources.

Deck - a WML document. May contain WMLScript.

Device - a device is a network entity that is capable of sending and receiving packets of information and has a unique device address. A device can act as both a client and a server within a given context or across multiple contexts. For example, a device can service a number of clients (as a server) while being a client to another server.

Server - a device (or application) that passively waits for connection requests from one or more clients. A server may accept or reject a connection request from a client.

User - a user is a person who interacts with a user-agent to view, hear, or otherwise use a rendered content.

User Agent - a user-agent (or content interpreter) is any software or device that interprets WML, WMLScript or resources. This may include textual browsers, voice browsers, search engines, etc.

WML - the Wireless Markup Language is a hypertext markup language used to represent information for delivery to a narrowband device, eg a phone.

WMLScript - a scripting language used to program the mobile device. WMLScript is an extended subset of the JavaScript™ scripting language.

4.2 Abbreviations

For the purposes of this specification, the following abbreviations apply.

API	Application Programming Interface
CGI	Common Gateway Interface
DCS	Digital Communications System
DTMF	Dual Tone Multi-Frequency
GSM	Global System for Mobile Communication
OS	Operating System
PCS	Personal Communications System
PDC	Personal Digital Cellular
RFC	Request For Comments
URI	Uniform Resource Identifier [RFC1630]
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAE	Wireless Application Environment [WAE]
WAP	Wireless Application Protocol [WAP]
WTA	Wireless Telephony Applications [WTA]
WTAI	Wireless Telephony Applications Interface [WTAI]
WWW	World Wide Web

5 WTA Background

The WAP WTAI features provide the means to create Telephony Applications, using a WTA user-agent with the appropriate WTAI function libraries. A typical example is to set-up a mobile originated call using the WTAI functions accessible from either a WML deck/card or WMLScript.

The application model for WTA is based on a WTA user-agent, executing WML and WMLScript. The WTA user-agent uses the WTAI function libraries to make function calls related to network services. The WTA user-agent is able to receive WTA events from the mobile network and pushed content, like WML decks and WTA events, from the WTA server. WTA events and WTAI functions make it possible to interact and handle resources, for call control etc., in the mobile network.

The WTA server can invoke applications dynamically using content push with WML and WMLScript.

5.1 WTAI Libraries

The WTAI features are partitioned into a collection of WTAI Function libraries. The type of function and its availability determines where the different functions are specified. The WTAI function libraries are accessible from both WML, using URL's, or from WMLScript using the scripting function libraries.

These functions may initiate an interaction between the mobile and the network. The function then typically terminates independently from the started network procedure. So any result delivered by the function call will not reflect the outcome of this procedure, which itself may result in events.

Example: A "user busy" condition is not reported by the return value of the "Setup Call" Function but is delivered by the "call cleared" event.

Network Common WTAI The most common features that are available in all networks. They are only accessible from the WTA user-agent. Examples of functions are call setup and answer incoming call.

Network Specific WTAI Features that are only available in certain types of networks. Operator specific features may also reside in this set.

Public WTAI Simple features that are available to third party applications executing using the standard WAE user-agent.

5.2 Event Handling

WTA event is one method that can be used to convey the change of state, in the WTA server or the mobile network. It's in the nature of network events to arise asynchronously from applications in the WTA user-agent and without any assumable order.

The WTA user-agent can be setup to act on a WTA event. WTA events must be mapped into URL's, indicating the content that must be loaded to handle the WTA event, either using the WTA event table, or can be handled dynamically, from within the WTA user-agent context using temporary event binding.

See more details on the event handling in the WTA Specification [WTA].

6 WTA Interface

6.1 WTAI Function Libraries

The WTAI functions are divided into libraries depending on type of function. A function library can also be specific to a certain type of network, and then a "well-known" network name is included in the name of the library. The WTAI specification defines the set of predefined WTAI function libraries for public and network common WTAI, listed below. Network specific WTAI function libraries are specified as addenda to the WTAI specification.

Table 1, Public WTAI Function Libraries

Function Library	*Name*	*Description of Library*
Public WTAI	“wp”	Public available WTAI functions.

Table 2, Network Common WTAI Function Libraries

Function Library	*Name*	*Description of Library*
Voice Call Control	“vc”	Voice Call Control library. Handles call setup and control of device during an ongoing call
Network Text	“nt”	Network Text library. Sending and retrieval of network text.
Phonebook	“pb”	Phonebook library. Manages the entries in the device phonebook.
Call Logs	“cl”	Call Logs library. Used for accessing different kinds of call logs in the device.
Miscellaneous	“ms”	Handling of miscellaneous features. An Example is logical indications.

6.2 WTAI API Delimiters

All parameters are assumed to be of type string, unless otherwise specified. The WTAI functions are accessed using the WTAI URI scheme, a CGI like style parameter scheme, or by using the defined WMLScript calls.

Notations used for the WTAI syntax:

- < > Angle brackets denotes an enumerated parameter
- [ ] Square brackets denote an optional section.
- | Vertical bar denotes a pair of mutually exclusive options
- ( )* Repeat none or multiple times
- *( ) Repeat one or multiple times

Specification of parameters:

A general rule is to always specify all input and output parameters unless otherwise stated. The WTA user-agent should not fail if a result parameter is not specified. The recommended procedure in this instance is to discard the result.

6.3 WTAI URI Scheme

Access to the WTAI function libraries from WML can be handled through URI "calls" using the dedicated WTAI URI encoding scheme. Using a predefined reference to the specific WTAI function library together with the actual function name forms the WTAI URI. The WTAI URI library identifier can be used to identify the library. An example of a predefined library is "WTAVoiceCall", specifying the common call control features.

A set of "well-known" network library names will be used to specify the WTAI URI's for the network specific features.

WTAI Functions are named using URI's. URI's are defined in [RFC1630]. The character set used to specify URI's is also defined in [RFC1630]. Consequently characters such as space, used in a WTAI URI, must be escaped, see [RFC1630] for more details on escaping.

```
wtai://<library>/<function> (; <parameter>)* [! <result>]
```

Table 3, WTAI URI scheme

<library>	Name that identifies the type of function, ie Voice Call Control uses the library name "vc".
<function>	Function identifier within specific library. An example is "ac" for the function "Accept Call" residing in the library "Network Common WTA".
<parameter>	Zero or more parameters to be sent to the function. Delimiter between subsequent parameters must be a semicolon ";".
<result>	Start of the result data section is indicated by an exclamation mark "!". Result is zero or more names of variables that will be set in the WTA user-agent context as a result from the function call. Delimiter between subsequent result data must be a semicolon ";".

6.4 WTAI Function Definition Format

Description	
This is where the function is described.	
URI:	The URI form of the function.
WMLScript:	The WMLScript form for the function.
Function ID:	The number of the function in its library.
Parameters:	Describes the identified parameters.
Output:	Describes the output of the function. See Appendix B for WTAI predefined error codes.
Examples:	Gives an example for function use as a URI Method. And an example of the function as a Script.
Associated Events:	This section lists WTA events that may occur after the function call has been initiated. An example is the function "Accept Call". An associated event in this case would be "Call Cleared".
Notes: Extra notes that may be helpful.	

6.5 Telephone Numbers

The following BNF specifies the formats for phone numbers and DTMF digits that are valid in WTAI function calls (URI and script API), if not stated otherwise in the specification of the function:

```
phone-number = international-phone-number / national-phone-number
international-phone-number = "+" 1*phonedigit
national-phone-number = 1*phonedigit
phonedigit = DIGIT
DIGIT = <any US-ASCII digit "0".."9">

dtmf-seq = 1*dtmf-digit
dtmf-digit = "*" / "#" / "A" / "B" / "C" / "D" / "," / DIGIT
```

7 Public WTAI

The Public WTAI functions are available to applications not originating from the network WTA server, ie third party server applications. The handling of public applications differs from the network WTAI in that the user must be able to cancel any specific operation before it is carried out. An example of an application, from a third party service provider, could be a “Phone number Guide” to customer services. The listed numbers are in fact “identifiers”, URI’s, that call the “Public WTAI” function “makeCall”.

Name: WTAPublic

Library ID: 512

Description: This library contains a public function that presents a number that can be dialled.

7.1 Make Call

Description	
This function is used to initiate a mobile originated call using the specified *number*. This number must be displayed to the user prior to place the call. The user must explicitly acknowledge the operation. The *Make Call* function can be used from within any application, not only WTA, to present the user with a number that can be dialled.	
URI:	wtai://wp/mc ; <number> [!<result>]
WMLScript:	makeCall(number);
Function ID:	0
Parameters:	<number> = String: Destination number to call. May use any valid telephony number characters and digits.
Output:	<result> = Integer: Zero if successful or a negative value in case of failure, the WTAI error code.
Examples:	URI: wtai://wp/mc; 5554367 WMLScript: WTAPublic.makeCall("5554367");
Associated Events:	-
Notes: The call must be terminated using the standard MMI.	

7.2 Send DTMF Tones

Description	
Send DTMF tone sequence through an active voice connection. The user must explicitly or implicitly acknowledge the operation. For instance, an acknowledgement made once for the public Make Call function could also be valid for all calls of the function Send DTMF Tones during that call. Or, acknowledgement can be made for each call of the Send DTMF function. This is implementation dependant.	
URI:	wtai://wp/sd ; <dtmf> [! <result>]
WMLScript:	sendDTMF(dtmf);
Function ID:	1
Parameters:	<dtmf> = String: Any valid sequence of standard DTMF characters.
Output:	<result> = Integer: Zero if successful or a negative value in case of failure, the WTAI error code.
Examples:	URI: wtai://wp/sd; 555*1234 WMLScript: WTAPublic.sendDTMF ("555*1234”);
Associated Events:	-
Notes: Like for the Make Call public function, the call must be terminated using the standard MMI.	

7.3 Add Phonebook Entry

Description	
Add a new entry to the phonebook to the next available entry. The name and number to be added must be displayed to the user. The user has to confirm ALL changes made to the phonebook. In case of unsuccessful operation, the output contains a negative number identifying the WTAI error code.	
URI:	wtai://wp/ap;<number>;<name> [! <result>]
WMLScript:	addPBEntry(number, name);
Function ID:	2
Parameters:	<number> = String: Phone number to be stored. <name> = String: Name that will be associated with the phone number.
Output:	<result> = String: The return value is "0" (zero) if successful or a value below zero indicating the WTAI error code. Predefined relevant errors are (see error codes values in Appendix B, table 5) : • "illegal number of parameters" • "out of memory"
Examples:	URI: wtai://wp/ap; 5551234; GENE; WMLScript: WTAPublic.addPBEntry("5554367", "EINSTEIN");
Associated Events:	-
Notes: -	

8 Network Common WTAI

Functions defined in this chapter apply to all types of mobile networks that WAP is intended to. Network specific functions are defined in addenda to this document.

8.1 Network Events

WTAI specifies the names of the WTA events that map to the mobile networks, native events. These mobile network events convey the state of services in the mobile network. They may be handled by the active context or can be used to start the WTA user-agent with a new context.

Table 4, Predefined network events

Event	*Parameters*	*Description*
cc/ic	id, callerID	Incoming Call indication. An incoming call has reached the user-agent and may be picked up from the application using the WTAI function “Accept Call”. < id>: Identity generated by the user-agent itself, to be used with subsequent call control operations. <callerID>: Contains the number of the calling party if available to the user-agent. Otherwise an empty string will be returned.
cc/cl	id, result	Call Cleared. The connected call, or the call that has been placed but not yet connected, is disconnected (independent of reason). <id>: The identity of the call that has been cleared. <result>: The result indicates why the call is cleared. See Appendix B, "WTAI predefined error codes".
cc/co	id, callerID	Call Connected. The called party has lifted the handset or accepted the incoming call. <id>: The identity of the call that has been answered or for which a notification has been received by the called party. <callerID>: Contains the number of the answering party if available to the user-agent. Otherwise an empty string will be returned.
cc/oc	id, callerID	Outgoing Call indication. An outgoing call is being setup by a client application e.g. using the WTAI function "Setup Call". <id>: Identity generated by the user-agent itself, to be used with subsequent call control operations. <callerID>: Contains the number of the called party. Otherwise an empty string will be returned.

cc/cc	id	Alert indication. An outgoing call is now ringing at the B-Party. <id>: Identity of the call.
cc/dtmf	resultstring	DTMF sent <resultstring> = String: DTMF sequence sent.
nt/it	id, sender	Incoming Network Text indication. The client has received a Network Text message. <id>: The identity of the received network text message generated by the user-agent itself. <sender>: Contains address information about the sending party if available to the user-agent. Otherwise an empty string will be returned.
nt/st	Text-ID	Network text sent <Text-ID> : Internal identity of the text sent.
ms/ns	camping, networkName, notCamping Cause	Network Status indication. The value of one, or more, of the defined parameters has changed. Other possible causes of this event could be hand over, change in network location, change of network etc. <camping> = BOOL: TRUE if the phone is camping on the network and is able to make and receive calls, otherwise FALSE. <networkName> = String: The string that is presented as standby information to the end user, identifying the network that the phone is camping on. <notCampingCause> = String: 0 = No network found. 1 = Only forbidden networks found.

8.2 Voice Call Control

During a call, the following WTAI functions can be used, where applicable, to control the operation of available call control features such as accept call and release call.

Name: WTAVoiceCall

Library ID: 513

Description: This library contains functions that are related to voice call control, common for all "well-known" networks.

8.2.1 Setup Call

Description	
Set-up a mobile originated voice call to the specified number. The *mode* parameter indicates how the call should be handled if the context in the WTA user-agent terminates. There are two modes, “drop” and “keep”. “Drop” means that the OS will release the call if the context should be restarted. “Keep” makes it possible to maintain the call even after the current context has terminated.	
URI:	wtai://vc/sc ; <number> ; <mode> [! <result>]
WMLScript:	setup(number, mode);
Function ID:	0
Parameters:	<number> = String: Destination number to call. May use any valid telephony number characters and digits. <mode>: 0 = drop, Drop Call when current context is removed. 1 = keep, Keep Call after current context is removed.
Output:	<result> = String: The return value is the identity of the created call or a negative number in case of failure, the WTAI error code.
Examples:	URI: wtai://vc/sc; 5554367;1 WMLScript: WTAVoiceCall.setup("5554367", 0);
Associated Events:	cc/cl, Call Cleared cc/co, Call Connected
Notes: -	

8.2.2 Accept Call

Description	
Accepts an incoming voice call or waiting call and lifts the handset. The “id” is the ordinal number assigned by the in-device call handler, and will be returned if the call is carried out. If the call, for some reason, can not be carried out the return value contains an error code. Any party or the network can terminate a call. When a call is terminated the Call Cleared event will be generated and may be detected by the application. The *mode* parameter indicates how the call should be handled if the context in the WTA user-agent terminates. There are two modes, “drop” and “keep”. “Drop” means that the OS will release the call if the context should be restarted. “Keep” makes it possible to maintain the call even after the current context has terminated.	
URI:	wtai://vc/ac;<id>; <mode> [! <result>]
WMLScript:	accept(id, mode);
Function ID:	1
Parameters:	<id> = String: The identity of the call to be accepted. <mode> = String: 0 = drop, Drop Call when current context is removed. 1 = keep, Keep Call after current context is removed.
Output:	<result> = String: The return value is the identity of the created call or a negative number in case of failure, the WTAI error code.
Examples:	URI: wtai://vc/ac; 1;1 WMLScript: WTAVoiceCall.accept ("1”, 0);
Associated Events:	cc/cl, Call Cleared cc/co, Call Connected
Notes: -.	

8.2.3 Release Call

Description	
Release the specified voice call. Calls involved in a multiparty group can be released using the call identity.	
URI:	wtai://vc/rc;<id> [! <result>]
WMLScript:	release(id);
Function ID:	2
Parameters:	<id> = String: The identity of the call to be released.
Output:	<result> = String: The return value is the identity of the released call or a negative number in case of failure, the WTAI error code.
Examples:	URI: wtai://vc/rc; 1 WMLScript: WTAVoiceCall.release ("1”);
Associated Events:	cc/cl, Call Cleared
Notes: -	

8.2.4 Send DTMF Tones

Description	
Send DTMF tone sequence through the specified voice call. If the call succeeds the integer value zero is returned. In case of unsuccessful outcome an error code will be returned.	
URI:	wtai://vc/sd;<id>;<dtmf> [! <result>]
WMLScript:	sendDTMF(id, dtmf);
Function ID:	3
Parameters:	<id> = String: The identity of the call on which to send the DTMF tones <dtmf> = String: Any valid sequence of standard DTMF characters
Output:	<result> = String: Integer value below zero indicates unsuccessful execution.
Examples:	URI: wtai://vc/sd; 2;555*1234 WMLScript: WTAVoiceCall.sendDTMF ("2","555*1234”);
Associated Events:	cc/dtmf, DTMF sent
Notes: -	

8.2.5 Call Status

Description	
Retrieves parameters associated with a specific call. In case of unsuccessful operation, the output contains a negative number identifying the WTAI error code.	
URI:	wtai://vc/cs;<id>;<field> [! <result>]
WMLScript:	callStatus (id, field);
Function ID:	4
Parameters:	<id> = String: Identity of the call. <field> = String: Name of field that is used to retrieve parameters associated with the call. Predefined fields are: ❑ *'number':* The number of the other party. This field is mandatory. ❑ *'name':* The name of the other party. In case no name exists the number value is returned. This field is optional. ❑ *'duration', 'durationHMS':* The duration of the call up to the current state of the call. If (at a specific state) no duration info is possible to return an empty value will be returned. Use of duration implies that a device specific time representation may be used (may depend on user preferences; 12/24 hour timekeeping system, time-zone etc.). Use of durationHMS is based on the [ISO8601] format. The following representation MUST be used: `HHH` = 3 digit hour, Total number of hours ("000" … "999"*) `MM` = 2 digit minute ("00" ... "59") `SS` = 2 digit second ("00" ... "59") Both fields are optional. *Example*: durationHMS="2505010" means 250 hours 50 minutes and 10 seconds.
	*) Editors note: Three digits for hours should be sufficient for the foreseeable future. However it is recommended that a WAP service parse (using appropriate WMLScript functions) in the order right to left. (I.e. begin with seconds). In case a device is able to return more than three (3) digits for the hours all remaining digits (up to the leftmost position) should be treated as part of the total number of hours.
	❑ *'state':* `The` state of the call. The value of the state field can be one of the following`:` *active* = an active call *hold* = the call is on hold *waiting* = the incoming call is not answered yet *connecting* = the outgoing call is in the connecting phase *disconnecting* = the call is in the disconnecting phase *released* = the call has been released This field is mandatory.

	❑ *'mode':* The mode of the call as defined when the call was setup using the functions Setup Call or Accept Call. The value of the mode field can be either 'keep' or 'drop'. This field is mandatory.
Output:	<result> = String: String value associated with the requested field. Value below zero indicates unsuccessful execution.
Examples:	URI: wtai://vc/cs;5;name; WMLScript: WTAVoiceCall.callStatus("5", "name");
Associated Events:	-
Notes: If the value of a supported field can not be retrieved, the empty string ("") is returned. If the field is not supported the WTAI error code (see Appendix B, Table 5) is returned.	

8.2.6 List Call

Description	
Returns the identities of the calls currently handled in the device and available to the WTA user-agent. The function is called repeatedly to retrieve information of all calls.	
URI:	wtai://vc/lc;<id> [! <result>]
WMLScript:	listCall (id)
Function ID:	5
Parameters:	<id> = String: Call list identity of the call. The id should index calls in the order of creation. If the id has the value of "0", or if the id is empty (""), the oldest call is returned, if id has the value of "1", the second oldest call is returned, and so on.
Output:	<callid> = String: The returned value is the identity of the call. This value can for example be used as input parameter to the function Call Status. In case there is no call matching the requested id, <callid> contains the WTAI error code (see Appendix B, Table 5).
Examples:	WMLScript: WTAVoiceCall.listCall(0);
Associated Events:	-
Notes: -	

8.3 Network Text

The Network Text WTAI function library handles sending and retrieval of text messages from the network text application in the device. Using the *read* function, entries in the network text application are retrieved as "*structs*", a formatted character string containing *fields* with associated data. Data from each field can then be retrieved using the *GetFieldValue* function. There are six types of "well-known" field names required for a minimal implementation:

- *text* Contains the `actual body of the network text message`
- *tstamp* Timestamp – Contains `the (local) time when the text message was sent`
- *tstamp_off* The offset from GMT to the timezone of *tstamp* in multiples of ¼ hours
- *tstamp_rec* Contains the local time when the text message was received in the device
- *address* Contains `the originating address (or destination address)`
- *status* Contains information about the status (unread, read, written, sent)

The available “network text” functions are send, read, remove and getFieldValue.

Name: WTANetText

Library ID: 514

Description: This library contains functions that handles sending and retrieval of network text.

8.3.1 Send Text

Description	
Sends a network text message, if feature is available in the network, to a destination identified by number.	
URI:	-
WMLScript:	send (number, text);
Function ID:	0
Parameters:	<number> = String: Destination number. Any valid telephony characters and digits. <text> = String: Network text data structure, the text to send
Output:	<result> = String: Integer value below zero indicates unsuccessful execution.
Examples:	WMLScript: WTANetText.send ("5554567”, “WAP Forum”);
Associated Events:	nt/st, Network text sent
Notes: -	

8.3.2 Read Text

Description	
`Returns` the network text data that may be stored in the device. Data is retrieved in the form of a field encoded character string. Use the GetFieldValue to extract values for any specific fields.	
URI:	-
WMLScript:	read (id);
Function ID:	1
Parameters:	<id> = String: Identity of network text to read. The id parameter should index messages sorted in a chronological order. If id has the value of "0", or if id is empty (“”), the most recent message is returned, if id has the value of "1", the second most recent message is returned, and so on.
Output:	<struct> = String: The name of variable to receive the network text data structure. Integer value below zero indicates unsuccessful execution.
Examples:	WMLScript: WTANetText.read (3);
Associated Events:	-
Notes: -	

8.3.3 Remove Text

Description	
Removes a network text message identified by id. If no record can be identified an error code will be returned.	
URI:	-
WMLScript:	remove (id);
Function ID:	2
Parameters:	<id> = String: Identity of network text message to be deleted.
Output:	<result> = String: Integer value below zero indicates unsuccessful execution.
Examples:	WMLScript: WTANetText.remove (3);
Associated Events:	-
Notes: -	

8.3.4 GetFieldValue

Description	
Retrieves the value, from the string <struct>, identified by “field”.	
URI:	-
WMLScript:	getFieldValue(struct,field);
Function ID:	3
Parameters:	<struct> = String: Formatted character string containing the fields with the associated data. <field> = String: Name of field containing the value that will be retrieved from <struct>.
Output:	<result> = String: String value associated with the requested field. Value below zero indicates unsuccessful execution. Encoded message structure. There are six types of predefined fields: ❑ *text:* A string containing the body of the network text message. This field is mandatory. ❑ *tstamp:* A string containing information about when the text message was sent. The *tstamp* value MUST be represented in the local time of the (originating) relaying network text element (e.g. GSM SMSC). This implies that if the receiving device cannot handle time zones, the default time zone is that of the home network. The following reduced representation based on [ISO8601] MUST be used: `YYYYMMDDHHMMSS` Where: `YYYY` = 4 digit year (“0000” ... “9999”) `MM` = 2 digit month (“01”=January, “02”=February ... “12”=December) `DD` = 2 digit day (“01”, “02” ... “31”) `HH` = 2 digit hour, 24-hour timekeeping system (“00” ... “23”) `MM` = 2 digit minute (“00” ... “59”) `SS` = 2 digit second (“00” ... “59”) This field is mandatory. *Example*: tstamp=“`19990430064500`” means 6:45 in the morning on the 30th of April 1999. ❑ *tstamp_off:* An integer containing tstamp’s offset from Co-ordinated Universal Time (GMT) in multiples of 15 minutes This field is mandatory unless unsupported by the device ❑ *tstamp_rec:* A string containing information about when the text message was received (or in the case of mobile originated Network Text, when it was sent). The tstamp_rec value is represented in the local time of the device. The format is the same as that of

	tstamp. This field is optional ❑ *address:* A string containing the originating address (or in the case of mobile originated network text, the destination address). This field is mandatory unless unsupported by the device ❑ *status:* A string containing the status of the message. The following values have been pre-defined: *unread*= a received message that has not been read *read* = a received message that has been read *written* = a message that has been written but not sent *sent* = a message that has been sent This field is mandatory unless unsupported by the device. In case there is no message matching the requested id, <struct> contains the WTAI error code (see Appendix B, Table 5).
	Editor's Note: Regarding "semi-mandatory" fields. The timestamp of a GSM SMS includes a field indicating the time-zone offset from GMT. Therefore, all devices that handle GSM SMS – not *all* GSM devices! – are mandated to convey this information through the *tstamp_off* field to a WTANetText caller.
Examples:	WMLScript: WTANetText.getFieldValue($struct,"text");
Associated Events:	-
Notes: If the value of a supported field can not be retrieved, the empty string ("") is returned. If the field is not supported the WTAI error code (see Appendix B, Table 5) is returned.	

8.4 Phonebook

The Phonebook WTAI function library handles requests for operations towards the phonebook application. The requested operations can be used for storage and retrieval of phonebook entries. It is also possible to search the phonebook for a certain number, name or identity. The Phonebook in general will be specified with an extensible format regarding available fields in order to facilitate “contacts”, (”address info”), applications.

Using the *read* function, phonebook entries are retrieved as "*structs*", a formatted character string containing *fields* with associated data. Data from each field can then be retrieved using the *GetFieldValue* function. There are three types of "well-known" field names required for a minimal implementation:

- *name* Contains the entry's name. This type is based on the vCard FN name type. Example: Albert A. Einstein.
- *number* Contains the entry's telephone number. The telephone number format is described in section 6.5.
- *id* Contains the entry's id

The Phonebook functions are write, read, remove and getFieldValue.

Name: WTAPhoneBook

Library ID: 515

Description: This library contains functions that handle operation towards the phonebook application[1], such as storage and retrieval of phonebook entries.

[1] *Existence of a phonebook application is implementation dependent and is not within the scope of WAP to define*

8.4.1 Write Phonebook Entry

Description	
Writes a new entry to the phonebook. Any previous phonebook entry with the same identity will be overwritten. If no identity is specified (“”) the next available phonebook entry will be used and the new identity is returned. In case of unsuccessful operation, the output contains a negative number identifying the WTAI error code.	
URI:	-
WMLScript:	write(id, number, name);
Function ID:	0
Parameters:	<id> = String: Identity of the phonebook entry. <number> = String: Phone number to be stored <name> = String: Name that will be associated with the phone number.
Output:	<result> = String: Phonebook entry identity. A value below zero indicates an error.
Examples:	WMLScript: WTAPhoneBook.write("2", "5554367", "EINSTEIN");
Associated Events:	-
Notes: -	

8.4.2 Read Phonebook Entry

Description	
Returns phonebook entries containing a value matched using the specified field. The function can also return sequential entries following the last match. Entries should be returned sorted ascending, using the selected field as the sort-key, if possible. There are three predefined fields: 1. The identity of an entry (="id"). E.g. "0", "1", "2" … 2. The phone number (="number"). E.g. "+46555418466"" 3. The name of an entry (="name"). E.g. "Albert Einstein"	
URI:	-
WMLScript:	read(field, value); Field is a string entity that identifies the name of a field containing the data indicated by value.
Function ID:	1
Parameters:	<field> = string: Predefined fields are: “id”, “number” and “name”. When using “id” or “number” then any preceding or trailing spaces (“ “) in the value-parameter should be ignored. Additional fields may be used. If a specific field is not supported an empty string (“”) shall be returned. <value> = string: Value is the actual data that is used when searching for a matching phonebook entry. The phonebook field containing the value is defined by the field parameter. Usage: read (“”, “”) = Reset the phonebook search mode and removes any previous search criteria. This function call returns an empty string (“”). read (field, “”) = Read next entry following the last match or the first entry in case the phonebook search mode have been previously reset. The function returns an empty string (“”) when all entries have been searched. Entries with the value of the field set to an empty string (“”) are returned (in the sorting sequence) after any non-empty values. read (field, value) = Read the entry with the matching value or the next ascending value. Matching is done from the start of the string. I.e.: Value = “alb” may match with “albert” but not directly with “einstein, albert”. Matching is not case sensitive.
Output:	<struct> = String: Encoded message structure or an empty string in case "read (“”, “”)" is used or no match could be found.
Examples:	WMLScript: WTAPhoneBook.read("name", "Albert");
Associated Events:	-
Notes: When searching the phonebook the sorting is implied by the selected field type. An implementation not supporting sorting is assumed to return entries in the ascending order using the identity field.	

8.4.3 Remove Phonebook Entry

Description	
Removes a phonebook entry. If the call succeeds then the result variables contains a zero. If the function fails then a negative number will be returned indicating the WTAI error code.	
URI:	-
WMLScript:	remove(id);
Function ID:	2
Parameters:	<id> = String: Identity of the phonebook entry.
Output:	<result> = String: Zero if successful. Integer value below zero indicates unsuccessful execution.
Examples:	WMLScript: WTAPhoneBook.remove("2");
Associated Events:	-
Notes: -	

8.4.4 GetFieldValue

Description	
Retrieves a value, from the string <struct>, identified by “field”.	
URI:	-
WMLScript:	getFieldValue(struct,field);
Function ID:	3
Parameters:	<struct> = String: Formatted character string containing the fields with the associated data. <field> = String: Name of field containing the value that will be retrieved from <struct>.
Output:	<result> = String: String value associated with the requested field.
Examples:	WMLScript: WTAPhoneBook.getFieldValue($struct,”name”);
Associated Events:	-
Notes: If the field does not exist in <struct> then the result contains an empty string.	

8.4.5 Change Phonebook Entry

Description	
Writes a change to an existing phonebook entry. Any previously stored value in the identified phonebook entry and field will be overwritten. In case of unsuccessful operation, the output contains a negative number identifying the WTAI error code.	
URI:	-
WMLScript:	change(id, field, value);
Function ID:	4
Parameters:	<id> = String: Identity of the phonebook entry. <field> = String: Name of field (part of a phonebook entry) to be changed. <value> = String: Value to be written to the specified field. Predefined fields are: ❑ *"name", "number":* Use of name implies that a “name” is stored in a format transparent to the device specific phonebook storage (no distinction between family name and first name). number implies that a “number” is stored in a format transparent to the device specific phonebook storage (no distinction between parts of a number e.g. area code).
Output:	<result> = String: Phonebook entry identity (handle). A value below zero indicates an error.
Examples:	WMLScript: WTAPhoneBook.change("5", "name", "Cathal Kennedy");
Associated Events:	-
Notes: If the field is not supported the WTAI error code (see Appendix B, Table 5) is returned.	

8.5 Call Logs

The functions specified in the following sections make it possible to access different types of call logs in the device. The call logs may hold other information than just the phone number, for example information about when the call was made/received.

Name: WTACallLog

Library ID: 519

Description: This library contains functions that enables access to call logs, common for all "well-known" networks.

8.5.1 Last Dialled Numbers

Description	
Returns entries from the “last dialled numbers” log. The GetFieldValue function is used to extract the value of a specific field from an entry.	
URI:	-
WMLScript:	dialled (id);
Function ID:	0
Parameters:	<id> = string: Identity of the entry to be returned. A value of “0” returns the entry corresponding to the last dialled number, a value of “1” returns the entry corresponding to the second last dialled number, and so on. If id is empty (“”), the entry corresponding to the last dialled number is returned.
Output:	<struct> = String: Encoded message structure. There are two types of predefined field names: ❑ *number:* A string containing the phone number without blanks (“ ”). If the phone number can not be provided from a non-empty entry in the log for any reason, the field should contain an empty string (“”). This field is mandatory. ❑ *timestamp:* A string containing information about when the entry was written to the log (if supported by the device). This field is optional. In case the end of the log is reached, this <struct> has the value of a WTAI error code "Id not found" (-1, see Appendix B, Table 5).
Examples:	WMLScript: WTACallLog.dialled("1");
Associated Events:	-
Notes:	

8.5.2 Missed Calls

Description	
Returns entries from the “missed calls” log. The GetFieldValue function is used to extract the value of a specific field from an entry.	
URI:	-
WMLScript:	missed (id);
Function ID:	1
Parameters:	<id> = string: Identity of the entry to be returned. A value of “0” returns the entry corresponding to the last missed call, a value of “1” returns the entry corresponding to the second last missed call, and so on. If id is empty (“”), the entry corresponding to the last missed call is returned.
Output:	<struct> = String: Encoded message structure. There are three types of predefined field names: ❑ *number:* A string containing the phone number without blanks (“ ”). If the phone number can not be provided from a non-empty entry in the log for any reason, the field should contain an empty string (“”). This field is mandatory. ❑ *timestamp:* A string containing information about when the entry was written to the log (if supported by the device). This field is optional. ❑ *class:* If the phone number is not provided by the *number* field (the *number* field contains an empty string), this field should contain information about the reason (provided by the device). This field is optional. In case the end of the log is reached, this <struct> has the value of a WTAI error code "Id not found" (-1, see Appendix B, Table 5).
Examples:	WMLScript: WTACallLog.missed ("");
Associated Events:	-
Notes:	

8.5.3 Received Calls

Description	
Returns entries from the “received calls” log. The GetFieldValue function is used to extract the value of a specific field from an entry.	
URI:	-
WMLScript:	received (id);
Function ID:	2
Parameters:	<id> = string: Identity of the entry to be returned. A value of “0” returns the entry corresponding to the last received call, a value of “1” returns the entry corresponding to the second last received call, and so on. If id is empty (“”), the entry corresponding to the last received call is returned.
Output:	<struct> = String: Encoded message structure. There are three types of predefined field names: ❑ *number:* A string containing the phone number without blanks (“ ”). If the phone number can not be provided from a non-empty entry in the log for any reason, the field should contain an empty string (“”). This field is mandatory. ❑ *timestamp:* A string containing information about when the entry was written to the log (if supported by the device). This field is optional. ❑ *class:* If the phone number is not provided by the *number* field (the *number* field contains an empty string), this field should contain information about the reason (provided by the device). This field is optional. In case the end of the log is reached, this <struct> has the value of a WTAI error code "Id not found" (-1, see Appendix B, Table 5).
Examples:	WMLScript: WTACallLog.received("0");
Associated Events:	-
Notes:	

8.5.4 GetFieldValue

Description	
Retrieves the value, from the string <struct>, identified by “field”.	
URI:	-
WMLScript:	getFieldValue(struct,field);
Function ID:	3
Parameters:	<struct> = String: Formatted character string containing the fields with the associated data. <field> = String: Name of field containing the value that will be retrieved from <struct>.
Output:	<result> = String: String value associated with the requested field. Value below zero indicates unsuccessful execution.
Examples:	WMLScript: WTACallLog.getFieldValue($struct,”number”);
Associated Events:	-
Notes: If the field does not exist in <struct> then the result contains an empty string.	

8.6 Miscellaneous

Various utility functions used with the WTA user-agent.

Name: WTAMisc

Library ID: 516

Description: This library contains functions for controlling logical device features like indications.

8.6.1 Indication

Description	
Turns logical indication on or off. The appearance in the MMI is implementation dependent. An example would be a logical indication that can be visual and/or audible. An indication can also be set to show for example the number of email messages.	
URI:	-
WMLScript:	indication(type, operation, count);
Function ID:	0
Parameters:	<type> = String: 0 = incoming speech call 1 = incoming data call 2 = incoming fax call 3 = call waiting 4 = received text 5 = voice mail notification 6 = fax notification 7 = e-mail notification 8-15 = extra notifications <operation> = String: 1 = Set. Activates the selected indication, i.e. starts ringing, animating etc. 2 = Reset. Changes the indicator back to the state it was before the set function was called or in case there are no previous set operation the default status for the indication will be set instead. <count> = String: The number of new text, voice mails etc.
Output:	<result> = Integer: Zero if successful or a negative value in case of failure, the WTAI error code.
Examples:	WMLScript: WTAMisc.indication(5, 1, 3);
Associated Events:	-
Notes: Count is not mandatory to show by the WTA user-agent, how count is used depends on the implementation.	

8.6.2 Terminate WTA User Agent

Description	
This function removes the content and terminates the context for the WTA user-agent.	
URI:	wtai://ms/ec [!<result>]
WMLScript:	endcontext;
Function ID:	1
Parameters:	-
Output:	<result> = Integer: Zero if successful or a negative value in case of failure, the WTAI error code.
Examples:	URI: wtai://ms/ec WMLScript: WTAMisc.endcontext;
Associated Events:	-
Notes: The *newcontext* attribute defined in [WML] is used when the context only needs to be cleared.	

8.6.3 Protect WTA User Agent Context

Description	
This function protects the WTA user-agent context from being interrupted by other means except for the end-user. Default is that the context is not protected. For reading the current protection status the function is called without the mode parameter.	
URI:	-
WMLScript:	protected (mode);
Function ID:	2
Parameters:	<mode> = String: 0 = Do not protect context. 1 = Protect context. Leaving this field empty (“”) results in only reading the current protection status.
Output:	<result> = String 0 = Context is not protected. 1 = Context is protected. Integer value below zero indicates unsuccessful execution.
Examples:	WMLScript: WTAMisc.protected(1);
Associated Events:	-
Notes: -	

Appendix A WTAI URI and WMLScript Function Libraries

In the tables below, the URI and WMLScript Function Libraries Calls are summarised. The arguments have been left out in order to increase readability. The figures in the column named "Lib/Func ID" denote the *Library* and *Function ID*s.

Public WTA

Public WTAI

Lib/Func ID	*URI*	*Script call*	*Description*
512.0	`wtai://wp/mc`	`WTAPublic.makeCall`	Make a call
512.1	`wtai://wp/sd`	`WTAPublic.sendDTMF`	Send DTMF Tones
512.2	`wtai://wp/ap`	`WTAPublic.addPBEntry`	Add a new phonebook entry

Network Common WTA

Voice Call Control

Lib/Func ID	*URI*	*WMLScript call*	*Description*
513.0	`wtai://vc/sc`	`WTAVoiceCall.setup`	Setup a new call
513.1	`wtai://vc/ac`	`WTAVoiceCall.accept`	Accept an incoming call
513.2	`wtai://vc/rc`	`WTAVoiceCall.release`	Release a call
513.3	`wtai://vc/sd`	`WTAVoiceCall.sendDTMF`	Send DTMF Tones
513.4	`wtai://vc/cs`	`WTAVoiceCall.callStatus`	Retrieve parameters for a specific call
513.5	`wtai://vc/lc`	`WTAVoiceCall.listCall`	Retrieve identities for all calls

Network Text

Lib/Func ID	*URI*	*WMLScript call*	*Description*
514.0	-	`WTANetText.send`	Send network text
514.1	-	`WTANetText.read`	Read network text
514.2	-	`WTANetText.remove`	Remove network text
514.3	-	`WTANetText.getFieldValue`	Get Field Value

Phonebook

Lib/Func ID	*URI*	*WMLScript call*	*Description*
515.0	-	`WTAPhoneBook.write`	Write phonebook entry
515.1	-	`WTAPhoneBook.read`	Read phonebook entry
515.2	-	`WTAPhoneBook.remove`	Remove phonebook entry
515.3	-	`WTAPhoneBook.getFieldValue`	Get Field Value
515.4	-	`WTAPhoneBook.change`	Change an existing phonebook entry

Call Logs

Lib/Func ID	*URI*	*WMLScript call*	*Description*
519.0	-	`WTACallLog.dialled`	Read "last dialled numbers" log
519.1	-	`WTACallLog.missed`	Read "missed calls" log
519.2	-	`WTACallLog.received`	Read "received calls" log
519.3	-	`WTACallLog.getFieldValue`	Get Field Value

Miscellaneous

Lib/Func ID	*URI*	*WMLScript call*	*Description*
516.0	-	`WTAMisc.indication`	Logical Indications
516.1	`wtai://ms/ec`	`WTAMisc.endcontext`	Terminates user-agent context
516.2	-	`WTAMisc.protected`	Sets/reads context protection mode

Appendix B WTAI predefined error codes

Functions in the WTAI function library may return a result code indicating the outcome of a function call. In most cases a positive integer indicates a successful outcome. WTAI defines a set of error codes, non-positive result codes, which can be returned by the WTAI functions. Note! Not all codes are used by all functions. Codes in the range -1 to -63 are reserved for WTA standard library functions. Network specific WTA must use codes in the range –64 to –127.

Table 5, WTAI predefined error codes

Error code	Description
-1	Id not found. Function could not be completed.
-2	Illegal number of parameters, function could not be resolved due to missing parameters.
-3	Service not available or non-existent function.
-4	Service temporarily unavailable.
-5	Called party is busy.
-6	Network is busy.
-7	No answer, ie call setup timed out.
-8	Unknown.
-9	Out of memory
-10 to -63	Reserved for future use by WTA standard library functions.
-64 to –127	Network specific error codes

Appendix C Examples using WTAI

WTAI functions can be called in either of the following two ways. First a WTAI function can be called as a URL call. The second way a WTAI function can be performed is via a Script. The two examples show how a simple problem could be solved using either WML or WMLScript.

Here is an example of a WTAI function as a URL call:

```
<WML>
  <CARD>
    <DO TYPE="ACCEPT" TASK="GO" URL="#eFood"/>
    Welcome!
  </CARD>

  <CARD NAME="eFood">
    <DO TYPE="ACCEPT" TASK="GO" URL="wtai://wp/mc;$FoodNum"/>
    Choose Food:
     <SELECT KEY="FoodNum">
      <OPTION VALUE="5556789">Pizza</OPTION>
      <OPTION VALUE="5551234">Chinese</OPTION>
      <OPTION VALUE="5553344">Sandwich</OPTION>
      <OPTION VALUE="5551122">Burger</OPTION>
     <SELECT>
  </CARD>
</WML>
```

Here is an example of a WTAI function as a Script call:

WMLSCRIPT:

```
  function CallFood(N) {
    var i = wtaVoiceCall.setup(N;1);
    if (i >= 0) {
     // Call is good, show call is done
      Browser.setVar("Msg", "Called");
      Browser.setVar("Nmbr", N);
    }
    else {
   // Call failed, we could tell user why
   Browser.setVar("Msg", "Error");
      Browser.setVar("Nmbr", $i);
    }
    Browser.go("displayMsg");
  }
```

```
<WML>
  <CARD>
     <DO TYPE="ACCEPT" TASK="GO" URL="/script#CallFood($FoodNum)"/>
     Choose Food:
    <SELECT KEY="FoodNum">
      <OPTION VALUE="5556789">Pizza</OPTION>
      <OPTION VALUE="5551234">Chinese</OPTION>
      <OPTION VALUE="5553344">Sandwich</OPTION>
      <OPTION VALUE="5551122">Burger</OPTION>
    <SELECT>
  </CARD>
  <CARD NAME="displayMsg">
    Call Status: $Msg $Nmbr
  </CARD>
</WML>
```

Notice the capability of error checking and reporting in the Script example.

Appendix D Predefined field names

Some WTAI functions use encoded structures with predefined fields. For information, these field names are summarised in the table below.

Field Name	***Description***
id	The identity of an entry in an in-device data store.
number	A phone number.
name	A name often stored together with a number in a data store entry.
address	The originating address of a network text message.
text	The text in a network text message.
timestamp	Information about when a call log entry has been changed or a network text message has been received or sent.
duration	Duration of a call using a device specific time representation.
durationHMS	Duration of a call based on the [ISO8601] format.
costValue	A device specific cost value that may or may not be related to any known currency.
costCurrency	A device specific currency type used with the costValue.
status	The status of a network text message (unread/read/written/sent).
state	The state of a call (active/hold/waiting/connecting/disconnecting/released).
mode	The mode of a call (keep/drop).
class	The reason why a number in a log for Missed Calls could not be retrieved.

Appendix E Static Conformance Requirements

This static conformance clause defines a minimum set of features that should be implemented to ensure that WTA could interact with the mobile network. A feature can be optional or mandatory. Although a function is mandatory it may not work, e.g. if the corresponding feature is not implemented in the mobile or in the network or if the user has no subscription for this feature.

E.1 Client features

E.1.1 Public WTAI Functions

Item	Function	Reference	Status
WTAI_P_C001	Make Call	7.1	M
WTAI_P_C002	Send DTMF Tones	7.2	M
WTAI_P_C003	Add a new phonebook entry	7.3	M

E.1.2 Network Common WTAI Functions

E.1.2.1 Network Events

Item	Function	Reference	Status
WTAI_CEV_C001	Incoming Call Indication (cc/ic)	8.1	M
WTAI_CEV_C002	Call Cleared (cc/cl)	8.1	M
WTAI_CEV_C003	Call Connected (cc/co)	8.1	M
WTAI_CEV_C004	Outgoing call indication (cc/oc)	8.1	M
WTAI_CEV_C005	Connecting call indication (cc/cc)	8.1	M
WTAI_CEV_C006	DTMF sent (cc/dtmf)	8.1	M
WTAI_CEV_C007	Incoming network text indication (nt/it)	8.1	M
WTAI_CEV_C008	Network text sent (nt/st)	8.1	M
WTAI_CEV_C009	Network status indication (ms/ns)	8.1	M

E.1.2.2 Voice Call Control

Item	Function	Reference	Status
WTAI_CVC_C001	Setup Call	8.2.1	M
WTAI_CVC_C002	Accept Call	8.2.2	M
WTAI_CVC_C003	Release Call	8.2.3	M
WTAI_CVC_C004	Send DTMF Tones	8.2.4	M
WTAI_CVC_C005	Call status	8.2.5	M
WTAI_CVC_C006	List call	8.2.6	M

E.1.2.3 Network Text

Item	Function	Reference	Status
WTAI_CNT_C001	Send Text	8.3.1	M
WTAI_CNT_C002	Read Text	8.3.2	M
WTAI_CNT_C003	Remove Text	8.3.3	M
WTAI_CNT_C004	GetFieldValue	8.3.4	M

E.1.2.4 Phonebook

Item	Function	Reference	Status
WTAI_CPB_C001	Write Phonebook Entry	8.4.1	M
WTAI_CPB_C002	Read Phonebook Entry	8.4.2	M
WTAI_CPB_C003	Remove Phonebook Entry	8.4.3	M
WTAI_CPB_C004	GetFieldValue	8.4.4	M
WTAI_CPB_C005	Change phonebook entry	8.4.5	M

E.1.2.5 Call Logs

Item	Function	Reference	Status
WTAI_CCL_C001	Last Dialed Numbers	8.5.1	M
WTAI_CCL_C002	Missed Calls	8.5.2	M
WTAI_CCL_C003	Received Calls	8.5.3	M
WTAI_CCL_C004	GetFieldValue	8.5.4	M

E.1.2.6 Miscellaneous

Item	Function	Reference	Status
WTAI_CM_C001	Indication	8.6.1	M
WTAI_CM_C002	Terminate WTA User Agent	8.6.2	M
WTAI_CM_C003	Protect WTA User Agent Context	8.6.3	M

E.1.3 WMLScript Bytecode Interpreter Capabilities

Item	Function	Reference	Status
WTAI_INT_C001	Supports Public WTAI library identifier	A	M
WTAI_INT_C002	Supports Public WTAI functions identifiers	A	M
WTAI_INT_C003	Supports Voice Call Control library identifier	A	M
WTAI_INT_C004	Supports Voice Call Control function identifiers	A	M
WTAI_INT_C005	Supports Network Text library identifier	A	M
WTAI_INT_C006	Supports Network Text function identifiers	A	M
WTAI_INT_C007	Supports Phonebook library identifier	A	M
WTAI_INT_C008	Supports Phonebook function identifiers	A	M
WTAI_INT_C009	Supports Call Logs library identifier	A	M
WTAI_INT_C010	Supports Call Logs function identifiers	A	M
WTAI_INT_C011	Supports Miscellaneous library identifier	A	M
WTAI_INT_C012	Supports Miscellaneous function identifiers	A	M

E.2 Server features

E.2.1 WMLScript Encoder Capabilities

Item	Function	Reference	Status
WTAI_ENC_S001	Supports Public WTAI library identifier	A	M
WTAI_ENC_S002	Supports Public WTAI functions identifiers	A	M
WTAI_ENC_S003	Supports Voice Call Control library identifier	A	M
WTAI_ENC_S004	Supports Voice Call Control function identifiers	A	M
WTAI_ENC_S005	Supports Network Text library identifier	A	M
WTAI_ENC_S006	Supports Network Text function identifiers	A	M
WTAI_ENC_S007	Supports Phonebook library identifier	A	M
WTAI_ENC_S008	Supports Phonebook function identifiers	A	M
WTAI_ENC_S009	Supports Call Logs library identifier	A	M
WTAI_ENC_S010	Supports Call Logs function identifiers	A	M
WTAI_ENC_S011	Supports Miscellaneous library identifier	A	M
WTAI_ENC_S012	Supports Miscellaneous function identifiers	A	M