WAP WTAI

Version 31-May-1999

Wireless Application Protocol
Wireless Telephony Application Interface Specification

Disclaimer:

This document is subject to change without notice.

Contents

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

[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, 1998. URL: http://www.wapforum.org/

[WMLScript] "WMLScript Language Specification", WAP Forum, 1998. URL: http://www.wapforum.org/

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

[WTA] "Wireless Telephony Application Specification", WAP Forum, 1998. 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
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.

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

Public WTA 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. 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 WTA function libraries for public and network common WTA, listed below. Network specific WTA function libraries are specified as addenda to the WTAI specification.

Table 1, Public WTA Function Libraries

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

Table 2, Network Common WTA 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. In addition, all parameters are mandatory, unless marked as optional. 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 is 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> (; <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.	

7 Public WTA

The Public WTA 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 WTA 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 WTA" 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*. 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>
WMLScript:	makeCall(number);
Function ID:	0
Parameters:	<number> = String: Destination number to call. May use any valid telephony number characters and digits.
Output:	-
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:	-
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.	

8 Network Common WTA

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 call control events

Event	*Parameters*	*Description*
cc/ic	id, callerID	Incoming Call indication. An incoming call has reached the user-agent 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 received notification of an 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.

8.2 Voice Call Control

During a call, the following WTA 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 created 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 an active voice connection. 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;<dtmf> [! <result>]
WMLScript:	sendDTMF(dtmf);
Function ID:	3
Parameters:	<dtmf> = String: Any valid sequence of standard DTMF characters
Output:	<result> = String: Integer value below zero indicates unsuccessful execution.
Examples:	URI: wtai://vc/sd; 555*1234 WMLScript: WTAVoiceCall.sendDTMF ("555*1234”);
Associated Events:	cc/cl, Call Cleared
Notes: -	

8.3 Network Text

The Network Text WTA function library handles sending and retrieval of text messages from the network text application in the device. 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:	-
Notes: -	

8.3.2 Read Text

Description	
Read 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. If id is empty (“”), the last unread message is returned.
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.
Examples:	WMLScript: WTANetText.getFieldValue($struct,”name”);
Associated Events:	-
Notes: If the field does not exist in <struct> then the result contains an empty string.	

8.4 Phonebook

The Phonebook WTA 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
- *number* Contains the entry's telephone number
- *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. "+46 555 418 466"" 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.	

8.5 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 (“”). In case the end of the log is reached, this field contains a negative number, the WTAI error code "Id not found" (-1, see Appendix B, Table 5). 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.
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 (""). In case the end of the log is reached, this field contains a negative number, the WTAI error code "Id not found" (-1, see Appendix B, Table 5). 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.
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 (“”). In case the end of the log is reached, this field contains a negative number, the WTAI error code "Id not found" (-1, see Appendix B, Table 2). 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.
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:	-
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
WMLScript:	endcontext;
Function ID:	1
Parameters:	-
Output:	-
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. WTA 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

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

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

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 WTA 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 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.