Wirless Telephony Application Interface

PDC-specific Addendum

Version 08-Sep-2001

Wireless Application Protocol
WAP-270-WTAIPDC-20010908-a

A list of errata and updates to this document is available from the WAP Forum™ Web site, http://www.wapforum.org/, in the form of SIN documents, which are subject to revision or removal without notice.

The WAP Forum™ assumes no responsibility for errors or omissions in this document. In no event shall the WAP Forum be liable for any special, indirect or consequential damages or any damages whatsoever arising out of or in connection with the use of this information.

WAP Forum™ members have agreed to use reasonable endeavors to disclose in a timely manner to the WAP Forum the existence of all intellectual property rights (IPR's) essential to the present document. The members do not have an obligation to conduct IPR searches. This information is publicly available to members and non-members of the WAP Forum and may be found on the "WAP IPR Declarations" list at http://www.wapforum.org/what/ipr.htm. Essential IPR is available for license on the basis set out in the schedule to the WAP Forum Application Form.

No representations or warranties (whether express or implied) are made by the WAP Forum™ or any WAP Forum member or its affiliates regarding any of the IPR's represented on this list, including but not limited to the accuracy, completeness, validity or relevance of the information or whether or not such rights are essential or non-essential.

This document is available online in PDF format at http://www.wapforum.org/.

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

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

Document History	
WAP-270-WTAIPDC-20010714-p	Proposed
WAP-270-WTAIPDC-20010908-a	Current

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"* [WAPARCH].

This document is an addendum to the *Wireless Telephony Application Interface* (WTAI). While WTAI defines an API that is valid for all supported types of mobile networks, this document outlines functions that are specific to PDC networks

2. References

2.1. Normative References

[CREQ] WAP-221, “Specification of WAP Conformance Requirements”, WAP Forum™, 25-Apr-2001 URL: http://www.wapforum.org/

[FORMAT] WAP-188, "WAP General Formats Document", WAP Forum™, 10-Jul-2001 URL: http://www.wapforum.org/

[PDC] RCR STD-27. Association of Radio Industries and Businesses (ARIB).

[RFC2119] “Key words for use in RFCs to Indicate Requirement Levels”. S. Bradner. March 1997 URL: http://www.ietf.org/rfc/rfc2119.txt

[WMLScript] WAP-193, "WMLScript Language Specification", WAP Forum™, 25-Oct-2000 URL: http://www.wapforum.org/

[WTA] WAP-266, "Wireless Telephony Application Specification", WAP Forum™, 08-Sep-2001 URL: http://www.wapforum.org/

[WTAI] WAP-268, "Wireless Telephony Application Interface Specification", WAP Forum™, 08-Sep-2001 URL: http://www.wapforum.org/

2.2. Informative References

[WAPARCH] WAP-210, “WAP Architecture”, WAP Forum™, 12-Jul-2001 URL: http://www.wapforum.org/

3. Terminology and Conventions

3.1. Conventions

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].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2. Definitions

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

3.3. Abbreviations

API	Application Programming Interface
PDC	Personal Digital Cellular
RFC	Request For Comments
WAP	Wireless Application Protocol
WTA	Wireless Telephony Applications
WTAI	Wireless Telephony Applications Interface

4. Introduction

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 described in [WTA].

The PDC addendum extends the support of [WTAI] to PDC technology devices by extending the network common Voice Call models and introuding a model for multiparty calls.

5. PDC Call Model

The WTA PDC call model is an extension of the network-common WTA voice call model described in [WTAI]. The PDC-specific extensions are described in this section.

5.1. PDC Call States

The WTA PDC call model is depicted in Figure 1 for an incoming call, in Figure 2 for an outgoing call, and in Figure 3 for a multiparty call. The call models are an extension of the WTA call models in [WTAI]. They represent the lifetime of one call and show all the call states and all the events that result in state transitions. A call may stay in a particular state for an indefinite amount of time.

PDC WTA implementations MUST generate WTA events according to these models. WTA implementations will generally rely on the underlying network signaling layer such that:

- network events correspond to zero or more WTA events
- the WTA imp lementation can support these call models without maintaining call state
- no WTA events need to be generated without an underlying network event.

Throughout this document, "PDC voice call" refers to an individual PDC voice call running on a WTA device that has implemented the WTAPDC library. It does not refer to a multiparty call. "PDC multiparty call" refers to a multiparty call running on a WTA device that has implemented the WTAPDC library. "PDC call" refers to either a "PDC voice call" or a "PDC multiparty call".

Note that:

- A PDC call enters the "call active" state and the WTAI "in call" state at the same time.
- A PDC voice call may transition between the "call active" and "call held" states at any time. These transitions are signaled by CallActive and CallHeld WTA PDC events.
- A PDC call may be released at any time. This transition is signaled by a CallCleared WTA event.

A PDC multiparty call is a special type of PDC call that controls a set of individual PDC voice calls. The state transitions for a PDC multiparty call are shown in Figure 3.

- A PDC multiparty call has its own, unique call handle. This call handle is conveyed to the WTA service when the PDC multiparty call is created. This is signaled by a CallConnected event for the multiparty call.
- A PDC multiparty call may be released at any time. This ends the PDC multiparty call and releases all the associated PDC voice calls. This transition is signaled by a CallCleared event on each of the associated PDC voice calls and the PDC multiparty call itself.

PDC voice calls that are part of a PDC multiparty call are always in the "call active" state. A specific PDC voice call that is part of a PDC multiparty call can be controlled using the following operations:

- A specific PDC voice call may be separated from a PDC multiparty call.

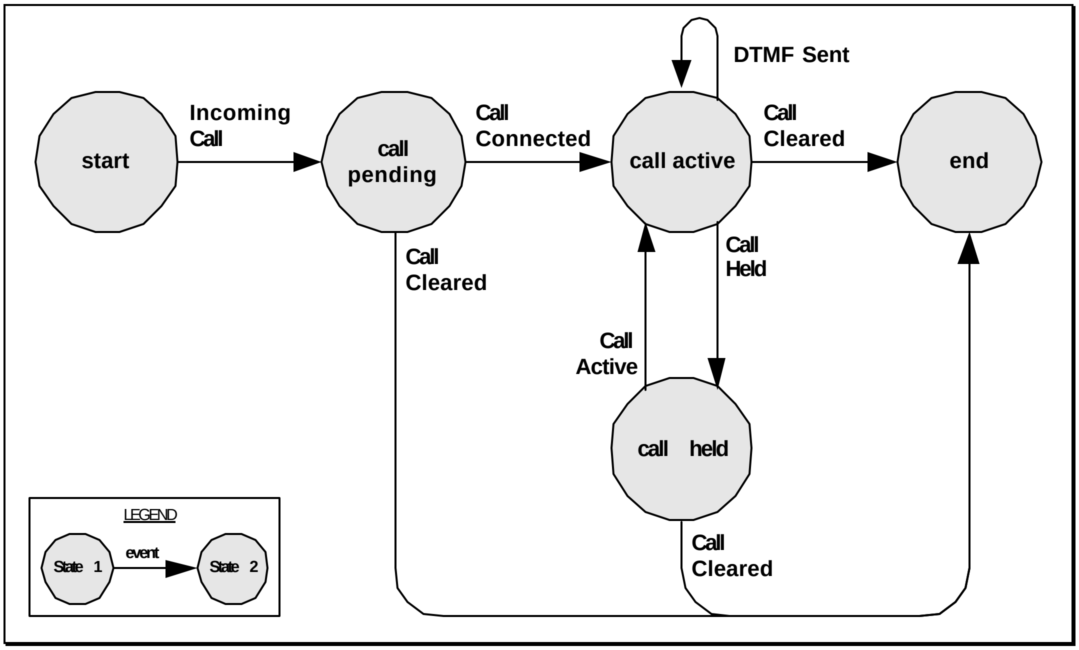

Figure 1 - PDC WTA Incoming Call Model

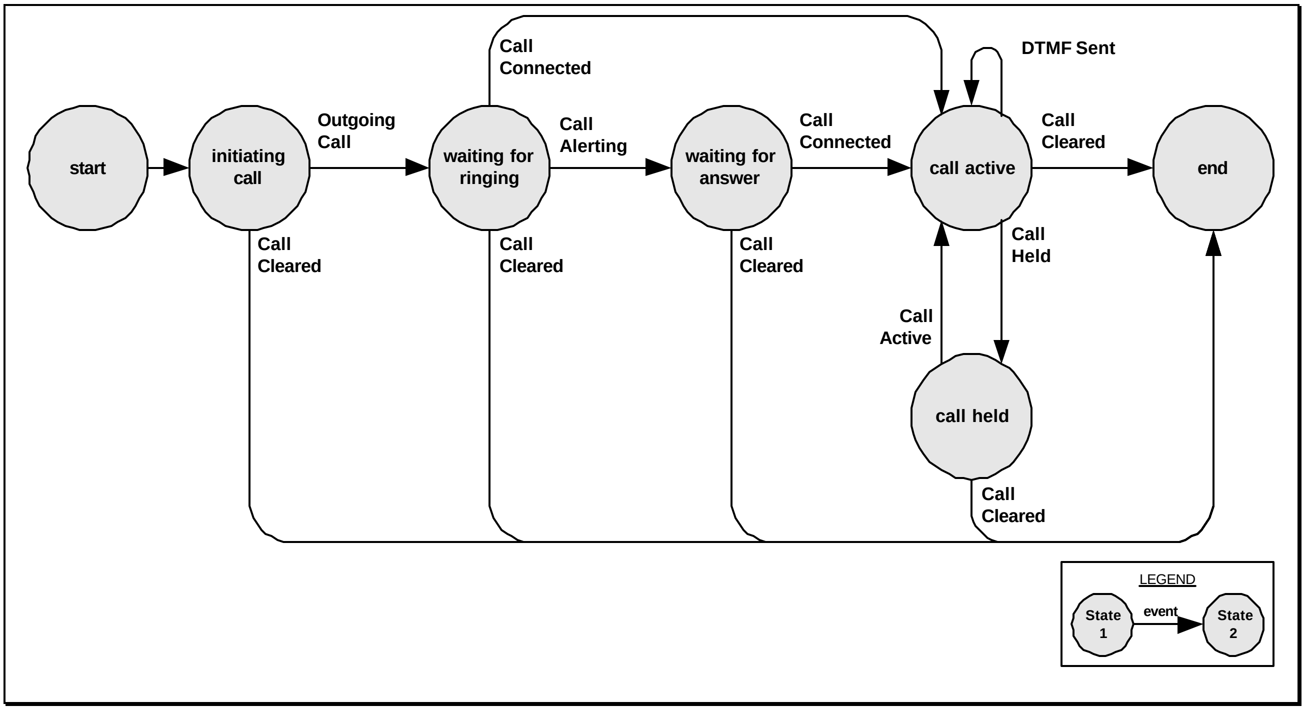

Figure 2 - PDC WTA Outgoing Call Model

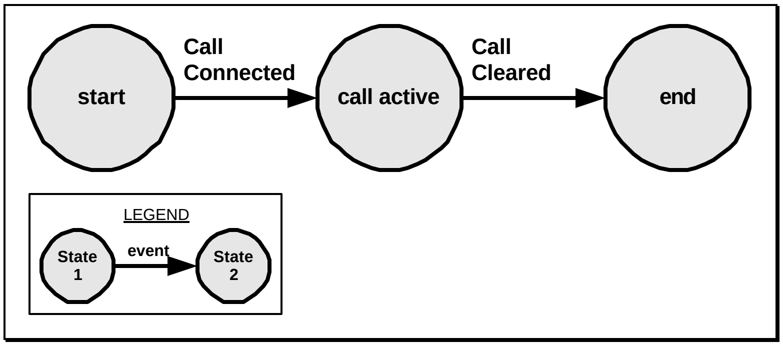

Figure 3 - PDC Multiparty Call Model

The WTA PDC call states are independent from the network-common WTA call states, however, there is a correlation between them:

If the PDC WTA voice call (incoming or outgoing) state is…	Then the network-common WTA call state MUST be…
call pending	call pending
initiating call	initiating call
waiting for ringing	waiting for ringing
waiting for answer	waiting for answer
call active	in call
call held	in call
End	End

If the PDC multiparty call state is…	Then the network-common WTA call state MUST be…
call active	in call
End	End

5.1.1. PDC Voice Call Information

A WTA user agent that implements the PDC library provides access to additional information about each PDC voice call. Each information field has a name and value. A field value may be retrieved using its field name.

The following PDC-specific fields MUST be available for each PDC voice call:

"PDCstatus" integer indicating the recent state of the call in the WTA PDC Incoming, Outgoing, or Multiparty Call Model diagram (see Figures 1, 2, and 3). Note that this field value may not be completely accurate or up-to-date since the state of the call may change at any time. It MUST be one of the following values (all numbers are shown in decimal):

1 = the PDC call is in the "call pending" state

2 = the PDC call is in the "initiating call" state

3 = the PDC call is in the "waiting for ringing" state

4 = the PDC call is in the "waiting for answer" state

5 = the PDC call is in the "call active" state

6 = the PDC call is in the "end" state

7 = the PDC call is in the "call held" state

"PDCmultiparty" integer indicating whether the PDC voice call is part of a multiparty call or not. If the call is not a participant of a PDC multiparty call, this value is zero. If the call is a participant of a PDC multiparty call, this value contains the handle of the multiparty call.

These fields exist in addition to the network-common WTA fields specified in [WTAI]. For example, a WTA service running on a PDC phone must be able to discover that a voice call's "status" is "in call" while at the same time it's "PDCstatus" is "call active".

6. Special Behavior of Network-Common WTAI

This section describes changes to the semantics or behavior of the network-common portion of WTA. A PDC WTA implementation MUST support the Network Common WTAI - Voice Call models specified in [WTAI].

6.1. WTA Events

These network-common events behave differently when used within a WTA device implementing the WTAPDC library. A PDC WTA implementation MUST support the changes to the Network Common WTAI – Voice Calls events as specified in this chapter.

6.1.1. wtaev-cc/cl

Description: In addition to the description specified in [WTAI], this event can be used to indicate that a PDC multiparty call has ended. In this case, the *`callHandle`* parameter contains the call handle for the PDC multiparty call as defined in [WTAI].

The *`result`* parameter contains a description of why the call was cleared. It must be one of the values defined for this event in [WTAI] or:

"100" normal termination of a PDC multiparty call, e.g., the near or far end released the voice call

"101" termination of PDC multiparty call - unspecified, no details available

6.1.2. wtaev-cc/co

Description: In addition to the description specified in [WTAI], this event can be used to indicate that a PDC multiparty call has been established. In this case, the *`callHandle`* parameter contains the call handle for the PDC multiparty call as defined in [WTAI], and the *`callerId`* parameter must contain an empty string.

6.2. WMLScript functions

These network-common functions behave differently when used within a WTA device implementing the WTAPDC library. A PDC WTA implementation MUST support the changes to the Network Common WTAI – Voice Calls functions as specified in this chapter.

6.2.1. WTAVoiceCall.release

Description: In addition to the behavior defined in [WTAI], this function can be used to release a PDC multiparty call. If the function is invoked with a call handle for a PDC multiparty call, the multiparty call must be released and all PDC voice calls that were part of the multiparty call must be released. This will be signaled by a CallCleared event on each of the associated PDC voice calls and a CallCleared event for the PDC multiparty call.

7. Network Specific WTAI - PDC

7.1. WTA Events

These events are related to PDC devices. All WTA event parameters are conveyed as strings. A PDC WTA implementation MUST support the Network Specific WTAI – PDC events specified in this chapter.

7.1.1. wtaev-pdc/ch

Event Name: CallHeld

Event ID: wtaev-pdc/ch

Parameters: *callHandle*

Description: Indicates a PDC voice call has been put on hold. The call may be remo ved from hold, i.e., made active again, using WTAPDC.retrieve or may be released using WTAVoiceCall.release.

The *callHandle* parameter contains the call handle for the PDC voice call as defined in [WTAI].

7.1.2. wtaev-pdc/ca

Event Name: CallActive

Event ID: wtaev-pdc/ca

Parameters: *callHandle*

Description: Indicates a PDC voice call has been made active, i.e., retrieved from hold.

The *callHandle* parameter contains the call handle for the PDC voice call as defined in [WTAI].

7.2. WMLScript Functions

The functions defined in this chapter follow the same function definition format as the one used in [WTAI]. Technical terms used in this chapter, e.g. events and error codes, are also explained in [WTAI]. A PDC WTA implementation MUST support the Network Specific WTAI – PDC WTAPDC.hold, WTAPDC.retrieve, WTAPDC.deflect, WTAPDC.multiparty and WTAPDC.separate functions specified in this chapter. A PDC WTA implementation MAY support the Network Specific WTAI – PDC WTAPDC.transfer function.

Name: WTAPDC

Library ID: 520

Description: This library contains functions that are unique to PDC implementations of WTA.

7.2.1. WTAPDC.hold

Function: hold (*callHandle*)

Function ID: 0

Description: Puts an active PDC voice call on hold. This function is non-blocking. Subsequent WTA events signal that the call has been put on hold.

The `callHandle` parameter identifies which call is to be put on hold. (See [WTAI] for a description of the call handle.)

This function returns an empty string if successful, or returns invalid if the function fails.

Permission Types: BLANKET, CONTEXT, SINGLE (see [WTA]).

Parameters: `callHandle` = handle

Return value: empty string or invalid

Associated Events: A CallHeld event occurs when the call is put on hold.

Exceptions: If the `callHandle` parameter does not refer to a PDC voice call that can be put on hold, e.g., the call is not in the "call active" state, this function returns invalid.

Example: var flag = WTAPDC.hold (handle);

7.2.2. WTAPDC.retrieve

Function: retrieve (`callHandle`)

Function ID: 1

Description: Makes a held PDC voice call active. This function is non-blocking. Subsequent WTA events signal the call has been made active.

The `callHandle` parameter identifies which call is to be retrieved from hold. (See [WTAI] for a description of the call handle.)

This function returns an empty string if successful, or returns invalid if the function fails.

Permission Types: BLANKET, CONTEXT, SINGLE (see [WTA]).

Parameters: `callHandle` = handle

Return value: empty string or invalid

Associated Events: A CallActive event occurs when the call is retrieved from hold.

Exceptions: If the `callHandle` parameter does not refer to an existing PDC voice call that can be retrieved, e.g., the call is not in the "call held" state, this function returns invalid.

Example: var flag = WTAPDC.retrieve (handle);

7.2.3. WTAPDC.transfer

Function: transfer (`callHandle`, `number`)

Function ID: 2

Description: Transfers a PDC voice call to another party. Given a PDC voice call, this function establishes a connection to a third party, connects the existing call to it, and disconnects the local device from both calls. This function is non-blocking. Subsequent WTA events signal the progress of the PDC calls involved.

The `callHandle` parameter identifies the call to be transferred. (See [WTAI] for a description of the call handle.) This call must be established before invoking this function; that is, the call must be in the "call active" state.

The `number` parameter specifies the destination to transfer the call and must be a phone-number as defined in [FORMAT].

This function returns an empty string if successful or returns invalid if the function fails.

Permission Types: BLANKET, CONTEXT, SINGLE (see [WTA]).

Parameters: `callHandle` = handle

`number` = string (phone-number)

Return value: empty string or invalid

Associated Events: A CallCleared event occurs when the original call is no longer available to the WTA user agent.

Exceptions: If the `callHandle` parameter does not refer to a PDC voice call that can be transferred, e.g., the call is in an unacceptable state, this function returns invalid.

If the `number` parameter is not a phone-number as defined in [FORMAT], this function returns invalid.

Example: var flag = WTAPDC.transfer (handle, "5551234");

7.2.4. WTAPDC.deflect

Function: deflect (`callHandle`, `destination`)

Function ID: 3

Description: Deflects an unanswered incoming PDC voice call to the voice mail system or to a pre-assigned number. The call is transferred without being answered first. This function is non-blocking. Subsequent WTA events signal that the call has ended.

The `callHandle` parameter identifies the call of interest. (See [WTAI] for a description of the call handle.)

The `destination` parameter specifies to where the call should be deflected.

This function returns an empty string if successful, or returns invalid if the function fails.

Permission Types: BLANKET, CONTEXT, SINGLE (see [WTA]).

Parameters: `callHandle` = handle

`destination` = integer (0=voice mail, 1=other pre-assigned number)

Return value: empty string or invalid

Associated Events: A CallCleared event for the incoming call occurs once the call has been deflected.

Exceptions: If the `callHandle` parameter does not refer to a PDC voice call that can be deflected, e.g., the call is not in the "call pending" state, this function returns invalid.

If the `destination` parameter is not acceptable, e.g. it is not one of the allowed values, this function returns invalid.

Example: var flag = WTAPDC.deflect (handle, 0);

7.2.5. WTAPDC.multiparty

Function: multiparty ()

Function ID: 4

Description: Establishes a PDC multiparty call. This function is non-blocking. Subsequent WTA events signal the progress of the PDC calls involved.

There must be one PDC voice call active and one PDC voice call on hold prior to invoking this function. Both calls will become part of the PDC multiparty call. The newly created PDC multiparty call is assigned a unique call handle.

This function returns the call handle of the PDC multiparty call if successful, or returns invalid if the function fails.

Permission Types: BLANKET, CONTEXT, SINGLE (see [WTA]).

Parameters: -

Return value: handle or invalid

Associated Events: A CallActive event occurs for the previously held call (since it automatically becomes active again).

A CallConnected event occurs for the newly-created PDC multiparty call.

Exception: If it is not possible to create the PDC multiparty call, e.g., there is not one active and one held voice call, this function returns invalid.

Example: var handle = WTAPDC.multiparty ();

7.2.6. WTAPDC.separate

Function: separate (*callHandle*)

Function ID: 5

Description: Separates a specific PDC voice call from a PDC multiparty call. This function is non-blocking. Subsequent WTA events signal the progress of the calls involved.

When the function is invoked, the specified PDC voice call is separated from the PDC multiparty call and a private communication is setup between the local PDC device and the specified party. At the same time, the other PDC voice call will be placed on hold and the PDC multiparty call is ended.

The *callHandle* parameter identifies the PDC voice call of interest. (See [WTAI] for a description of the call handle.)

This function returns an empty string if successful, or returns invalid if the function fails.

Permission Types: BLANKET, CONTEXT, SINGLE (see [WTA]).

Parameters: *callHandle* = handle

Return value: empty string or invalid

Associated Events: A CallHeld event occurs for the PDC voice call that gets put on hold.

A CallCleared event occurs for the PDC multiparty call.

Exceptions: If the *callHandle* parameter does not refer to a PDC voice call that can be retrieved from the PDC multiparty call, e.g., it is not a participant in a PDC multiparty call, this function returns invalid.

Example: var flag = WTAPDC.separate (handle);

Appendix A. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [CREQ].

A 1 Client features

A 1.1 PDC Call Model

Item	Function	Reference	Status	Requirement
WTAIPDC-CM-C-001	PDC Call Model	5	M	WTAI-VCM-C-001

A 1.2 WTA Events

Item	Function	Reference	Status	Requirement
WTAIPDC-E-C-001	Call Held (wtaev-pdc/ch)	7.1.1	M	
WTAIPDC-E-C-002	Call Active (wtaev-pdc/ca)	7.1.2	M	
WTAIPDC-E-C-003	Call Cleared (wtaev-cc/cl)	6.1.1	M	WTAI-CVE-C-002
WTAIPDC-E-C-004	Call Connected (wtaev-cc/co)	6.1.2	M	WTAI-CVE-C-003

A 1.3 WMLScript Functions

Item	Function	Reference	Status	Requirement
WTAIPDC-S-C001	WTAPDC.hold	7.2.1	M	
WTAIPDC-S-C-002	WTAPDC.retrieve	7.2.2	M	
WTAIPDC-S-C-003	WTAPDC.transfer	7.2.3	O	
WTAIPDC-S-C-004	WTAPDC.deflect	7.2.4	M	
WTAIPDC-S-C-005	WTAPDC.multiparty	7.2.5	M	
WTAIPDC-S-C-006	WTAPDC.separate	7.2.6	M	
WTAIPDC-S-C-007	WTAVoiceCall.release	6.2.1	M	WTAI-CVS-C-003

A 1.4 WMLScript Bytecode Interpreter Capabilities

Item	Function	Reference	Status	Requirement
WTAIPDC-INT-C-001	Supports PDC Network WTAI library identifier	7.2	M	WMLS:MCF
WTAIPDC-INT-C-002	Supports PDC Network WTAI function identifiers	7.2	M	WMLS:MCF

A 2 Server features

A 2.1 WMLScript Encoder Capabilities

Item	Function	Reference	Status	Requirement
WTAIPDC-ENC-S-001	Supports PDC Network WTAI library identifier	7.2	M	WMLS:MSF
WTAIPDC-ENC-S-002	Supports PDC Network WTAI function identifiers	7.2	M	WMLS :MSF

Appendix B. WMLScript Function Libraries (Informative)

Lib/Func ID	WMLScript call	Description
520.0	WTAPDC.hold	Put a call on hold
520.1	WTAPDC.retrieve	Make a held call active again
520.2	WTAPDC.transfer	Transfer an active call
520.3	WTAPDC.deflect	Deflect an unanswered call
520.4	WTAPDC.multiparty	Create a multiparty call
520.5	WTAPDC.separate	Retrieve a party from a multiparty call

Table 1 , WMLScript Functions

Appendix C. Change History (Informative)

Type of Change	Date	Section	Description
Class 0	08-Sep-2001		The initial version of this document.