Wirless Telephony Application Interface

GSM-specific Addendum

Version 08-Sep-2001

Wireless Application Protocol
WAP-255-WTAIGSM-20010908-a

.

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-255-WTAIGSM-20010710-p	Proposed
WAP-255-WTAIGSM-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 networks using GSM technology. In this specification, the following networks are supported: GSM, DCS1800 and PCS1900.

2. References

2.1. Normative References

[CREQ] WAP-221, “Specification of WAP Conformance Requirements”, WAP Forum™, 24-Apr-2001 URL: http://www.wapforum.org/

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

[GSM 02.30] GSM 02.30 "Digital cellular telecommunications system (Phase 2+); Man-Machine Interface (MMI) of the Mobile Station (MS)"

[GSM 02.83] GSM 02.83 "Digital cellular telecommunications system; Call Waiting (CW) and Call Holding (HOLD) Supplementary Services - Stage 1"

[GSM 02.84] GSM 02.84 "Digital cellular telecommunications system; Multi Party (MPTY) supplementary services; Stage 1"

[GSM 02.90] GSM 02.90 "Digital cellular telecommunications system; Unstructured Supplementary Services Data (USSD) – Stage 1"

[GSM 03.38] GSM 03.38 " Digital cellular telecommunications system (Phase 2+); Alphabets and language-specific information"

[GSM 04.07] GSM 04.07 "Digital cellular telecommunications system (Phase 2+); Mobile radio interface signalling layer 3; General aspects"

[GSM 04.08] GSM 04.08 "Digital cellular telecommunications system (Phase 2+); Mobile radio interface; Layer 3 specification"

[GSM 04.80] GSM 04.80 " Digital cellular telecommunications system (Phase 2+); Mobile radio interface layer 3 supplementary services specification; Formats and coding"

[GSM 04.90] GSM 04.90 " Digital cellular telecommunications system (Phase 2+); Unstructured Supplementary Service Data (USSD); Stage 3"

[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
DCS	Digital Communications System
GSM	Global System for Mobile Communication
PCS	Personal Communications System
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 GSM addendum extends the support of [WTAI] to GSM technology devices by extending the network common Voice Call models and introducing a model for multiparty calls. The addendum also provides access to GSM location information as well as access to GSM specific network services.

5. GSM

5.1. GSM Call Model

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

5.1.1. GSM Call States

The WTA GSM 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.

GSM 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 implementation can support these call models without maintaining call state
- no WTA events need to be generated without an underlying network event.

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

Note that:

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

A GSM multiparty call is a special type of GSM call that controls a set of individual GSM voice calls. Its behaviour is defined in [GSM 02.84]. The state transitions for a GSM multiparty call are shown in Figure 3. Note that the diagram applies to a GSM multiparty call where the GSM WTA device is the “served mobile subscriber” (as defined in [GSM 02.84]).

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

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

- A specific GSM voice call may be separated from a GSM multiparty call. The GSM multiparty call must initially be in the "call active" state.
- A specific GSM voice call may be disconnected from a GSM multiparty call. The GSM multiparty call could be in either the “call active” or “call held” state.

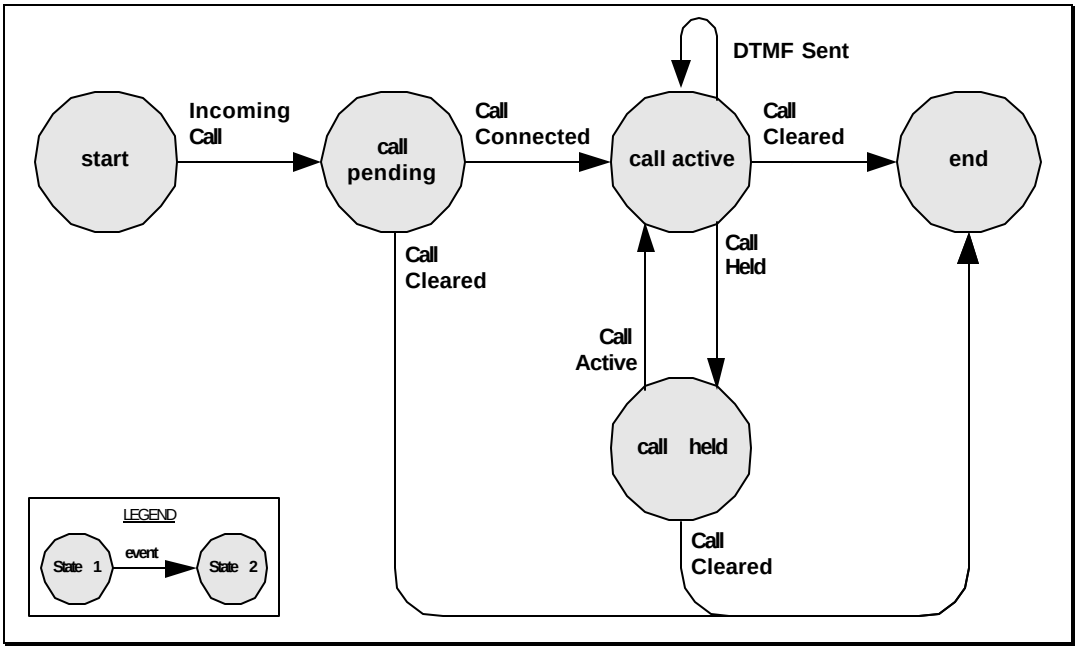

Figure 1 - GSM WTA Incoming Call Model

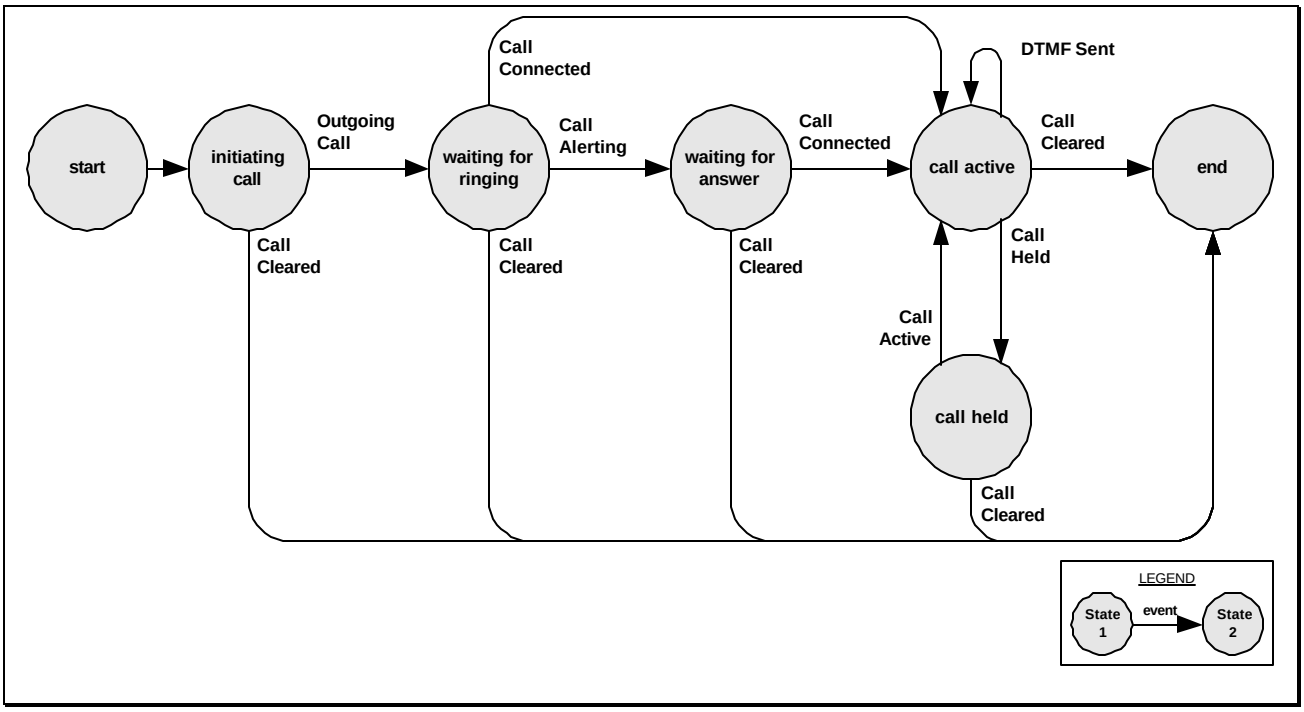

Figure 2 - GSM WTA Outgoing Call Model

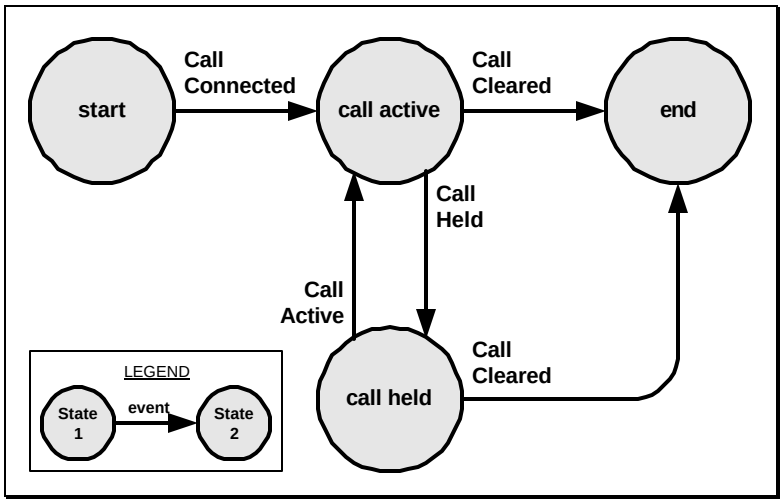

Figure 3 - GSM multiparty Call Model

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

If the GSM 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 GSM multiparty call state is…	Then the network-common WTA call state MUST be…
call active	In call
call held	In call
End	End

5.1.2. GSM Voice Call Information

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

The following GSM-specific fields MUST be available for each GSM voice or multiparty call:

"`GSMstatus`" `integer` indicating the recent state of the call in the WTA GSM 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 GSM call is in the "call pending" state

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

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

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

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

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

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

"`GSMmultiparty`" `integer` indicating whether the GSM call is part of a multiparty call or not. If the call is not a participant of a GSM multiparty call, this value is zero. If the call is a participant of a GSM multiparty call, `GSMmultiparty` 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 GSM phone must be able to discover that a voice call's "`status`" is "`in call`" while at the same time it's "`GSMstatus`" is "`call active`".

5.2. GSM Location Information

The WTA GSM user agent provides access to GSM location information. The following octets MUST be returned for the GSM location information:

octets 1 – 3 Mobile Country & Network Codes (MCC & MNC), coded as specified in [GSM 04.08].

octets 4 – 5 Location Area Code (LAC), coded as specified in [GSM 04.08].

octets 6 – 7 Cell Identity Value (Cell ID), coded as specified in [GSM 04.08].

octet 8 Timing Advance, coded as specified for the Timing Advance information element in [GSM 04.08] starting at octet 2 (the IEI is removed). The Timing Advance value is that of the active dedicated connection (call or SMS). If there is no active dedicated connection, the value is that of the last active dedicated connection.

NOTE: The ME should store the last value of the Timing Advance.

NOTE: Using the Status value, the application can be aware of potential misinterpretation of the Timing Advance value.

octet 9 Status, coded as a bit field:

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Phone status	NMR present	SPARE					

Phone status:

0 = the device is in idle mode (not connected)

1 = the device is not in idle mode

NMR (Network Measurement Results) present:

0 = the NMR field is not present

1 = the NMR field is present

The following octets are returned for the Network Measurement Results (as indicated by the NMR bit of the Status octet):

octets 10 - 25 Network Measurement Results, coded as for the Measurement Results information element in [GSM 04.08] starting at octet 2 (the IEI is removed).

6. Special Behaviour of NetworkCommon WTAI

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

6.1. WTA Events

These network-common events behave differently when used within a WTA device implementing the GSM library. A GSM 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 GSM multiparty call has ended. In this case, the `callHandle` parameter contains the call handle for the GSM 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 his event in [WTAI] or:

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

"101" termination of GSM 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 GSM multiparty call has been established. In this case, the `callHandle` parameter contains the call handle for the GSM 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 GSM library. A GSM 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 behaviour defined in [WTAI], this function can be used to release a GSM multiparty call. If the function is invoked with a call handle for a GSM multiparty call, the multiparty call must be released and all GSM voice calls that were part of the multiparty call must be released. This will be signalled by a CallCleared event on each of the associated GSM voice calls and a CallCleared event for the GSM multiparty call.

7. Network Specific WTAI - GSM

7.1. WTA Events

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

7.1.1. wtaev-gsm/ch

Event Name: CallHeld

Event ID: `wtaev-gsm/ch`

Parameters: *`callHandle`*

Description: Indicates a GSM voice or multiparty call has been put on hold. The call may be removed from hold, i.e., made active again, using WTAGSM.retrieve or may be released using WTAVoiceCall.release.

The *`callHandle`* parameter contains the call handle for the GSM voice or multiparty call as defined in [WTAI].

7.1.2. wtaev-gsm/ca

Event Name: CallActive

Event ID: `wtaev-gsm/ca`

Parameters: *`callHandle`*

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

The *`callHandle`* parameter contains the call handle for the GSM voice or multiparty call as defined in [WTAI].

7.1.3. wtaev-gsm/ru

Event Name: USSDReceived

Event ID: `wtaev-gsm/ru`

Parameters: *`message`*, *`codingScheme`*, *`type`*, *`transactionId`*

Description: Indicates a USSD message has been received.

The *message* parameter contains the USSD message. It may contain any of the USSD characters permitted by [GSM 02.90]. For type 3 messages, this parameter must contain an empty string.

The *`codingScheme`* parameter specifies the encoding of the *message* parameter. The *`codingScheme`* parameter must contain one of the values specified in [GSM 04.90] and [GSM 03.38]. For type 3 messages, this parameter must contain an empty string.

The *`type`* parameter indicates the type of USSD message, as specified in [GSM 04.80]. It must be one of the following values:

"0" = result to a ProcessUnstructuredSS-Request operation

"1" = UnstructuredSS-Request operation

"2" = UnstructuredSS-Notify operation

"3" = error to a ProcessUnstructuredSS-Request operation

The *transactionId* parameter contains the transaction ID of the message as specified in [GSM 04.07] §11.

7.1.4 wtaev-gsm/cw

Event Name: CallWaitingStatus

Event ID: `wtaev-gsm/cw`

Parameters: *type*

Description: Indicates a result message to a Call Waiting operation (7.2.9) has been received.

The *type* parameter indicates the status of Call Waiting. It must be one of the following values:

"1" = Call Waiting is activated

"2" = Call Waiting is deactivated

"3" = Call Waiting operation disallowed

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 GSM WTA implementation MUST support the Network Specific WTAI – GSM functions specified in this chapter.

Name: WTAGSM

Library ID: 518

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

7.2.1. WTAGSM.hold

Function: `hold`(*callHandle*)

Function ID: 0

Description: Puts an active GSM voice or multiparty 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: `string` (empty), or `invalid` (failure indication)

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

Exceptions: If the *callHandle* parameter does not refer to a GSM call that can be put on hold, eg, the call is not in the "call active" state, this function returns invalid.

Example:
```
var flag = WTAGSM.hold(handle);
```

7.2.2. WTAGSM.retrieve

Function: `retrieve`(*callHandle*)

Function ID: 1

Description: Makes a held GSM voice or multiparty 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: `string` (empty), or `invalid` (failure indication)

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

Exceptions: If the *callHandle* parameter does not refer to an existing GSM voice or multiparty call that can be retrieved, eg, the call is not in the "call held" state, this function returns `invalid`.

Example:
```
var flag = WTAGSM.retrieve(handle);
```

7.2.3. WTAGSM.transfer

Function: `transfer(`*callHandleB*`,`*callHandleC*`)`

Function ID: 2

Description: Transfers a GSM call to another party. Given two GSM calls, to parties B and C, this function connects the two calls and disconnects the local device from the call. This function is non-blocking. Subsequent WTA events signal the progress of the GSM calls involved.

The *callHandleB* parameter identifies the first call involved in the transfer. (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" or "call held" state.

The *callHandleC* parameter identifies the second call involved in the transfer. (See [WTAI] for a description of the call handle.) This call may already be established or may be in the process of being established when this function is invoked; that is, the call may be in any state other than "end".

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

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

Parameters: *callHandleB* = `handle`

callHandleC = `handle`

Return value: `string` (empty), or `invalid` (failure indication)

Associated Events: A CallCleared event occurs for each of the GSM calls B and C when they are no longer available to the WTA user agent.

Exceptions: If the *callHandleB* parameter does not refer to a GSM call that can be transferred as party B, e.g., the call is in an unacceptable state, this function returns `invalid`.

If the *callHandleC* parameter does not refer to a GSM call that can be transferred as party C, e.g., the call is in an unacceptable state, this function returns `invalid`.

Example:
```
var flag = WTAGSM.transfer(handleB, handleC);
```

7.2.4. WTAGSM.deflect

Function: `deflect(`*callHandle*`,`*number*`)`

Function ID: 3

Description: Deflects an unanswered incoming GSM voice call to another 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 *number* parameter specifies the destination for 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: `string` (empty), or `invalid` (failure indication)

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

Exceptions: If the *callHandle* parameter does not refer to a GSM voice call that can be deflected, e.g., the call is not in the "call pending" 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 = WTAGSM.deflect(handle, "5551234");
```

7.2.5. WTAGSM.multiparty

Function: `multiparty()`

Function ID: 4

Description: Establishes a GSM multiparty call or adds an additional GSM voice call to an established GSM multiparty call. This function is non-blocking. Subsequent WTA events signal the progress of the GSM calls involved.

If a GSM voice call is on hold and a second GSM voice call is active, then this function creates an active GSM multiparty call that contains both of the GSM voice calls. The newly created GSM multiparty call is assigned a unique call handle.

If a GSM multiparty call is on hold and a GSM voice call is active, then this function adds the active GSM voice call to the existing multiparty and makes the resultant multiparty call active. The resultant GSM multiparty call keeps the previously assigned call handle.

If a GSM multiparty call is active and a GSM voice call is on hold, then this function adds the held GSM voice call to the existing multiparty and keeps the resultant multiparty call active. The resultant GSM multiparty call keeps the previously assigned call handle.

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

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

Parameters: -

Return value: `handle` (call handle of GSM multiparty call), or `invalid` (failure indication)

Associated Events: If the function is successfully invoked, a `CallActive` event occurs for the previously held call.

If a multiparty call is created as a result of this function, a `CallConnected` event also occurs for the newly created GSM multiparty call.

Exceptions: If it is not possible to create or add to the multiparty call, e.g., there is not one active and one held call, this function returns `invalid`.

Example:
```
var handle = WTAGSM.multiparty();
```

7.2.6. WTAGSM.separate

Function: `separate(`*`callHandle`*`)`

Function ID: 5

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

The GSM multiparty call must be active and there must not be a call on hold.

When the function is invoked, the specified GSM voice call is separated from the GSM multiparty call and a private communication is setup between the local GSM device and the specified party.

Depending on the number of parties in the GSM multiparty call, the multiparty call may be placed on hold or may be ended (as defined in [GSM 02.84]).

The *`callHandle`* parameter identifies the GSM 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: `string` (empty), or `invalid` (failure indication)

Associated Events: If the GSM multiparty call is put on hold as a result of this function call, a `CallHeld` event occurs for the GSM multiparty call.

If the GSM multiparty call is cleared as a result of this function call, a `CallCleared` event occurs for the GSM multiparty call.

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

Example:
```
var flag = WTAGSM.separate(handle);
```

7.2.7. WTAGSM.sendUSSD

Function: `sendUSSD(`*`message, codingScheme, type, transactionId`*`)`

Function ID: 6

Description: Sends a USSD message. This function is blocking. If this function is invoked successfully, the USSD message is guaranteed to be conveyed to the network.

The *`message`* parameter contains the USSD message to send. It may contain any of the USSD characters permitted by [GSM 02.90].

The *`codingScheme`* parameter specifies the encoding of the *`message`* parameter. The *`codingScheme`* parameter must contain one of the values specified in [GSM 04.90] and [GSM 03.38].

The *`type`* parameter indicates the type of USSD message, as described in section 7.1.3.

NOTE: For a type 2 message, the mobile must send a RELEASE COMPLETE message for the transaction ID associated with this USSD message after sending the FACILITY message.

In the case where the sent USSD message is in response to a network initiated USSD message, i.e. a type 1 or 2 message, the *transactionId* parameter contains the transaction ID of the corresponding network initiated USSD message as specified in [GSM 04.07] §11. If the sent USSD message is not in response to a network initiated USSD message, i.e. a type 0 message, then this parameter shall take the value 0.

This function returns the transaction ID of the USSD message as specified in [GSM 04.07] §11 if successful, or returns `invalid` if the function fails. Note that the returned value is equal to the *transactionId* parameter only for type 1 and type 2 messages.

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

Parameters: *message* = `string` (USSD message body)

codingScheme = `integer`

type = `integer` (message type:

0 = ProcessUnstructuredSS-Request operation

1 = result of a UnstructuredSS-Request operation

2 = result of a UnstructuredSS-Notify operation)

transactionId = `integer` (message transaction ID)

Return value: `integer` (transaction ID), or `invalid` (failure indication) or `error-code` (must be one of the following decimal values:

-100 = USSD dialogue in progress

-101 = illegal characters or too many characters

-106 = network is not available

-1 = unspecified error)

Associated Events: -

Exceptions: If the *codingScheme* parameter contains an unacceptable value, this function returns `invalid`.

If the *type* parameter contains an unacceptable value, this function returns `invalid`.

If the *type* parameter indicates the message is type 1 or type 2 and the *transactionId* parameter contains an unacceptable value, this function returns `invalid`.

If the device cannot deliver of the USSD message this function returns `invalid`.

Example:
```
var tid2 = WTAGSM.sendUSSD(resultMsg, coding, 1, tid);
```

7.2.8. WTAGSM.netinfo

Function: `netinfo(`*type*`)`

Function ID: 7

Description: Provides the current network information of the GSM terminal.

The *type* parameter specifies how much of the network measurement results to return: no Network Measurement Result values, Network Measurement Result values for the six "best" surrounding cells, or Network Measurement Result values for all possible surrounding cells.

This function returns the GSM location information in hexadecimal format if successful, or returns invalid if the function fails. (See section 5.2 for a description of the GSM location information.)

Each octet of information is represented by two hexadecimal characters. There are no space or other delimiting characters between the pairs of hexadecimal digits representing each octet.

Permission Types: SINGLE (see [WTA]).

Parameters: *type* = `integer` (amount of network measurement results to return:

0 = return no network measurement results

1 = return network measurement results for the six "best" surrounding cells

2 = return network measurement results for all possible surrounding cells)

Return value: `string` (GSM location information), or `invalid` (failure indication)

Associated Events: -

Exceptions: If the *type* parameter is unacceptable, i.e., it is not one of the allowed values, this function returns `invalid`.

If the network information is not available, e.g., the device is not in service, this function returns the empty `string`.

Example:
```
var info = WTAGSM.netinfo(1);
```

7.2.9. WTAGSM.callWaiting

Function: callWaiting(*type*)

Function ID: 8

Description: Controls the GSM Supplementary Service Call Waiting for telephony as specified in [GSM 02.83]. This function is non-blocking.

The type parameter defines whether the Call Waiting service shall be activated, deactivated or interrogated.

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

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

Parameters: *type* = `integer` (type:

0 = interrogation of Call Waiting service

1 = activation of Call Waiting service

2 = deactivation of Call Waiting service)

Return value: `string` (empty), or `invalid` (failure indication) or `error-code` (must be one of the following decimal values:

-106 = network is not available

-1 = unspecified error)

Associated Events: If the function is successfully invoked, a `CallWaitingStatus` event occurs as a result of the network answer.

Exceptions: If the *type* parameter contains an unacceptable value, this function returns `invalid`.

Example: var flag = WTAGSM.callWaiting(1);

7.2.10. WTAGSM.sendBusy

Function: sendBusy(*callHandle*)

Function ID: 9

Description: Releases a pending voice call. On being alerted by an incoming call the WTAGSM.sendBusy function shall set User Determined User Busy (UDUB) for that call (according to [GSM 02.30]). This presents BUSY to the calling party or invokes call forwarding on busy, if active and operative. This function is non-blocking. Subsequent WTA events signal that the call has been cleared.

The *callHandle* parameter identifies which pending call is to be released(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 CallCleared event occurs when the call terminates.

Exceptions: If the *callHandle* parameter does not refer to a voice call that can be released, this function returns `invalid`.

Example: var flag = WTAGSM. sendBusy(handle);

Appendix A. Static Conformance Requirements (Normative)

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

A 1 Client features

A 1.1 GSM Call Model

Item	Function	Reference	Status	Requirement
WTAIGSM-CM-C-001	GSM Call Model	5.1	M	WTAI-VCM-C-001

A 1.2 GSM Location Information

Item	Function	Reference	Status	Requirement
WTAIGSM-LI-C-001	Accessible GSM location information.	5.2	M	

A 1.3 WTA Events

Item	Function	Reference	Status	Requirement
WTAIGSM-E-C-001	USSD Received (wtaev-gsm/ru)	7.1.3	M	
WTAIGSM-E-C-002	Call Held (wtaev-gsm/ch)	7.1.1	M	
WTAIGSM-E-C-003	Call Active (wtaev-gsm/ca)	7.1.2	M	
WTAIGSM-E-C-004	Call Cleared (wtaev-cc/cl)	6.1.1	M	WTAI-CVE-C-002
WTAIGSM-E-C-005	Call Connected (wtaev-cc/co)	6.1.2	M	WTAI-CVE-C-003
WTAIGSM-E-C-006	Call Waiting Status (wtaev-gsm/cw)	7.1.4	M	

A 1.4 WMLScript Functions

Item	Function	Reference	Status	Requirement
WTAIGSM-S-C-001	WTAGSM.hold	7.2.1	M	
WTAIGSM-S-C-002	WTAGSM.retrieve	7.2.2	M	
WTAIGSM-S-C-003	WTAGSM.transfer	7.2.3	M	
WTAIGSM-S-C-004	WTAGSM.multiparty	7.2.5	M	
WTAIGSM-S-C-005	WTAGSM.separate	7.2.6	M	
WTAIGSM-S-C-006	WTAGSM.netinfo	7.2.8	M	
WTAIGSM-S-C-007	WTAGSM.sendUSSD	7.2.7	M	
WTAIGSM-S-C-008	WTAGSM.deflect	7.2.4	M	
WTAIGSM-S-C-009	WTAVoiceCall.release	6.2.1	M	WTAI-CVS-C-003
WTAIGSM-S-C-010	WTAGSM.callWaiting	7.2.9	M	
WTAIGSM-S-C-011	WTAGSM.sendBusy	7.2.10	M	

A 1.5 WMLScript Bytecode Interpreter Capabilities

Item	Function	Reference	Status	Requirement
WTAIGSM-INT-C-001	Supports GSM Network WTAI library identifier	7.2	M	WMLS:MCF
WTAIGSM-INT-C-002	Supports GSM Network	7.2	M	WMLS:MCF

Item	Function	Reference	Status	Requirement
	WTAI function identifiers			

A 2 Server features

A 2.1 WMLScript Encoder Capabilities

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

Appendix B. WMLScript Function Libraries (Informative)

In the table below, the WMLScript Function Libraries Calls valid for GSM networks are summarised. The arguments have been left out in order to increase readability. The values in the column named "Lib/Func ID" denote the *Library* and *Function ID*s.

Lib/Func ID	*WMLScript call*	*Description*
518.0	WTAGSM.hold	Put a call on hold
518.1	WTAGSM.retrieve	Make a held call active again
518.2	WTAGSM.transfer	Transfer an active call
518.3	WTAGSM.deflect	Deflect an unanswered call
518.4	WTAGSM.multiparty	Join/create a multiparty call
518.5	WTAGSM.separate	Retrieve a party from a multiparty call
518.6	WTAGSM.sendUSSD	Send a USSD message
518.7	WTAGSM.netinfo	Get network information
518.8	WTAGSM.callWaiting	Control of Call Waiting Service
518.9	WTAGSM.sendBusy	Send Busy on incoming 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.