Specification Information Note

WAP-169_104-WTA-20010508-a

Version 08-May-2001

for

Wireless Application Protocol
WAP-169-WTA-20000707-a

Wireless Telephony Application Specification
Version 07-Jul-2000

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.

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/.

Contents

1. Scope

This document provides changes and corrections to the following document files:

- WAP-169-WTA-20000707-a
- WAP-169_102-WTA-20010206-a

It includes changes from the following change requests:

- CREC-WTA-MOTOROLA/NOKIA/ERICSSON-29-Nov-2000, Event handling
- CREC-WTA-ERICSSON-01-Sep-2000-5, user agent behaviour
- CREC-WTA-ERICSSON-15-Feb-2001-1, removing storage space requirements
- CR-WTA-PHONEDO-02-MAR-2001(Event Handling)

2. Notation

In the subsections describing the changes new text is underlined. Removed text has ~~strikethrough~~ marks. The presented text is copied from the specification. Text that is not presented is not affected at all. The change descriptions may also include editor's notes similar to the one below. The notes are not part of the actual changes and must not be included in the changed text.

Editor's note: Framed notes like these only clarify where and how the changes shall be applied.

3. Event Handling

3.1 Change Classification

Class 3 – Clerical Corrections

3.2 Change Summary

The current WTA spec is ambiguous with regards to handling of events as evidenced by questions posed to the WTA mailing list that indicate there are differing interpretations of the process that a WTA user agent must follow regarding WTA events. Unfortunately, a full understanding of the WTA user agent is unclear from reading of the WTA spec alone and such confusion will almost certainly lead to incompatible implementations resulting from devices that exhibit different behaviours.

This change proposes that the section on Event Handling be modified to make it more precise and complete with the intention of helping ensure interoperability. This change does not propose any changes to the accepted or understood behaviour of the WTA user agent, hence it is proposed as an Editorial change.

3.3 Change Description

Editor's note: Changes in chapter 4 includes added, removed or changed definitions to be consistent with other proposed changes.

Section 10 Event Handling: The description of WTA user agent behaviour with respect to event handling is modified to make the text more pricise, to clearly separate requirements from examples, and to make the description consistent with the WML spec (use of terminology, etc).

Annex 2 – Static Conformance Requirements: The subsection describing SCR items related to event handling is updated.

Annex 3 – Using Events: This annex describes serveral behaviour that a content developer must be aware of.

Editor's note: On page 7.

4.1 Definitions

~~**Clear** - used in conjunction with WTA context means that all variables stored in the WTA user agent are removed.~~

~~**Implicit Variables**~~**Event Parameters** - ~~are~~ variables that ~~are created by the user agent~~ convey data specific to a WTA event instance and can be accessed using WTA-WML~~as a result of a WTA event reception~~.

~~**Temporary**~~ **Local Binding** - a WTA event binding that is specified using WTA-WML content.

WTA Event - ~~a notification, in the form of an abstract WTA event that conveys a change of state in the mobile network~~the representation of a network event, e.g. an ~~Incoming~~ incoming call, as conveyed to the WTA user agent.

Network Event – An event or activity related to the mobile network, for example an i~~I~~ncoming c~~C~~all, which occurs outside of any WTA context~~, but~~ and which ~~is~~ may be conveyed to the WTA user-agent in the form of a WTA event and subsequently processed within a WTA context.

Editor's note: On page 32.

10. Event Handling

10.1 Description

A mobile device detects and responds to network events. Examples of network events are an incoming voice call and receipt of a network text message. How the network event occurs and is generated or detected is outside the scope of WTA.

Within the WTA framework, a network event may be conveyed to the WTA user agent as a WTA event; *all* network events are not conveyed since some network events do not have an equivalent WTA event. and some network events which do have equivalent WTA events may be associated with processing which is out of the scope of the WTA user agent (see section 5.1.1 The WTA and WAE User Agents).

The WTA user agent subsequently processes each WTA event according to the current WTA context. Typically, an author will bind a task (e.g., `go`) to a WTA event type [WML]. When the event occurs, the task bound to the event type is invoked. Using this mechanism, the author can control the responses to WTA events and thus implement the WTA service as desired.

10.2 Event Bindings

A WTA user agent may respond to a WTA event using two different methods: local bindings and global bindings.

A local binding associates a WTA event type with a task that applies to the current (or active) card. A local binding is similar in form and function to a WML intrinsic event binding. It is specified using the WTA-WML `onevent` element where the `type` attribute specifies the WTA event type (see [WML] for more information on event bindings and see [WTAI] for a list of WTA event types).

The following example shows a card, `c1`, that has a local binding between an Incoming Call event (i.e., `wtaev-cc/ic`) and a `go` task:

```
<wta-wml>
      <card id="c1">
            <!-- setup a local binding for an incoming call -->
            <onevent type="wtaev-cc/ic"> <go href="#c2"/> </onevent>
            Waiting for a call.
      </card>

      <card id="c2">
            <!-- an incoming call was detected, so display a message -->
            You have a call.
      </card>
</wta-wml>
```

Local bindings can be defined in card and template elements. Card-level bindings can shadow template-level bindings. Scope and shadowing semantics of local bindings (i.e., the `onevent` element) are defined in [WML]. Local bindings may not be invoked under certain conditions (see section 10~~9~~.6 below).

In the absence of a local binding, the user agent may check (see section 10~~9~~.6) the set of global bindings. A global binding associates a WTA event with a URL that is fetched and processed. A global binding is specified using a WTA `channel` element with the URL specified in the `href` attribute of the first `resource` element (see section 8 for more information on WTA channels and the repository). For example, a global binding to an Incoming Call event would use:

```
      <channel maxspace="1024" channelid="Incoming Call Handler"
EventIdeventid="wtaev-cc/ic">
            <title>Incoming Call Selection Service</title>
            <resource href="someDeck.wml#someCard">
      </channel>
```

Local bindings hide global bindings while the local bindings are in scope.

10.3 Event Parameter Reference

WTA events contain data specific to each event instance. During WTA event handling, the WTA context is updated with these event parameters so that the content author may access them. The WTA event parameters are accessed left to

right using an indexing notation. The following syntax is used to substitute an event parameter into a card or deck (see [WML] for definitions of `conv` and `digit`):

```
eventvar =      ( "$" eventvarname ) |
              ( "$(" eventvarname ( conv )? ")" )
eventvarname =  ( digit )+
```

Thus, when a WTA context is updated with WTA event parameters, the variable named "0" contains the first event parameter and is referenced using the notation "$0", the variable named "1" contains the second and is referenced using "$1", etc. Any reference to a non-existent WTA event parameter is resolved with the empty string according to rules defined in [WML].

Event parameters can be used as any other WML variables and are subject to the same constraints (with the exception to their syntax). Event parameter references are legal anywhere variable references are legal in [WML]. Event parameters are read-only so they cannot be the target variable for an element that could attempt to change their value. For example, the following are not allowed; setvar with name = “1”; select with name or iname = “1”; or input with name= “1” ~~and are not allowed to be target variables for the `setvar` element (e.g. name="1")~~. However, such state may be overwritten by the next WTA event (see section 7.3). ~~However, unlike “traditional” variables, event parameters may change when a new event arrives.~~ Since a subsequent WTA event will overwrite the event parameters, authors should copy the event parameters into ~~traditional~~ WML variables to prevent loss of the values.

~~**Editor's note**: The index notation used for referencing event parameters, as described in this section, is not specified in [WML]. There is work in progress in the WAP WTA group and WAG WAE Drafting Committee to make sure that the WTA and WML specifications will not conflict.~~

10.4 Stable State

A WTA user agent is in a stable state when it has no context or when all of the following conditions are met:

- It is not processing a WTA event (as described in section 10.6)
- It is not “processing content” (e.g., parsing WTA-WML, rendering WTA-WML, executing WMLScript, executing a WMLScript library)
- It is not establishing bindings for the current context
- It is not navigating or transitioning between cards or between decks

~~, except if it is waiting for a WML deck to be retrieved from the network.~~

—In essence, the WTA user agent is in a stable state if there is no risk of a race condition occurring during processing of a WTA event. ~~Note that when there is no context (for example, after the WTAMisc.endcontext() function is invoked), the WTA user agent is inherently stable~~.

10.5 Event-based Invocation and Access Control

According to [WML], the `access` element of a WML deck (and hence also a WTA-WML deck) specifies access control information for the entire deck. It contains `domain` and `path` attributes that specify which other resources (e.g., WTA-WML decks) may access the deck. That is, access control for a target deck is imposed in terms of the resource from which the navigation is being made. The referring resource is identified by its URI.

Whenever a WTA user agent navigates to a resource that was the result of a local binding, the user agent must enforce the `access` element of the target deck using the URI of the referring card. That is, the referring resource is the card that contains the local binding.

Whenever a WTA user agent navigates to a resource that was the result of a global binding, the user agent must establish a new context and enforce the `access` element of the target deck using the URI of the channel specifying that global binding. That is, the referring resource is the channel document that contains the global binding. If the channel does not have an associated URI, then the user agent must fail the access request

10.6 Event Handling Process

A single WTA user agent instance must not allow more than one WTA context to exist at a time; it is illegal for a WTA user agent to have multiple concurrent WTA contexts. ~~This specification does not, however, preclude the existence of~~

~~multiple WTA user agents, although the behavior of or interaction between the multiple WTA user agents running concurrently within the same device is not specified.~~

The step-by-step process below describes the reference model for handling of WTA events by the WTA user agent. All WTA user agents must implement this process, or one that is indistinguishable from it. The WTA user agent must not begin processing of any WTA event unless it is in a stable state as defined in section 10~~9~~.4. If ~~the WTA user agent receives a WTA event~~ a network event which has a corresponding WTA event occurs while the WTA user agent is in an unstable state, ~~it~~the implementation should delay processing of the WTA event until ~~it~~the WTA user agent is in a stable state. Any delayed WTA events ~~and~~ must eventually be processed ~~the delayed WTA events~~ in the same order as they were received~~. The WTA user agent must not interrupt or suspend the executing context~~. ~~Instead of having the WTA user agent delay the WTA event, the implementation could choose to delay it and deliver it to the WTA user agent once the user agent enters the stable state again. This behaviour is however implementation dependent. Also in this case the order of the events must be preserved, meaning that all events received and delayed by the implementation must eventually be conveyed as WTA events to the WTA user agent in the order they were received by the implementation.~~

If the WTA user agent can not ~~queue handle the~~ start processing a WTA event due to, e.g. resource limitations, ~~e.g. if the event queue is full,~~ the WTA user agent ~~must~~MUST~~T~~MAY terminate the current context. ~~and flush the WTA event queue. In this case, the default MMI should handle all pending WTA events, i.e. those in the WTA event queue plus the one that caused the exception condition.~~ How it is decided that the WTA user agent is not able to start processing a WTA event, and how delayed events are taken care of when the current context is terminated, is implementation dependent. It is also up to the implementation when to resume delivery of WTA events to the WTA user agent.

This is the step-by-step process:

1. If there is no WTA context, go to step ~~5~~4.

~~2.If the WTA context is untrusted as defined in section 6, go to step 5. Note: an untrusted WTA context can not be protected and may be terminated unexpectedly.~~

~~3.~~2. If there is a local binding for the WTA event …
   a. Remove all WTA event parameters from the current WTA context.
   b. Update the current WTA context with the WTA event parameters, if any, as defined in section 10~~9~~.3.
   c. Invoke the task associated with the local binding using the current WTA context (see [WML] for more detail on processing tasks).
   d. If the invocation succeeds, go to step ~~8~~6.
   e. If the invocation fails~~, restore the current WTA context to the state it was in immediately prior to step 0~~…
      i) Terminate the current WTA context. ~~Remove the WTA event parameters from the current WTA context.~~
      ii) ~~Restore the previous WTA event parameters into the current WTA context.~~
      iii) ~~Perform the Task Execution Failure process in [WML].~~
      iv) Go to step ~~1~~6.

~~4.~~3. If the current WTA context is protected, it must remain unaffected…
   a. go to step ~~7~~6.

~~5.~~4. If there is a global binding for the WTA event …

a. Terminate the current WTA context, if any, as defined in section 7.

b. Create a new WTA context as defined in section 7. The new context becomes the current WTA context.

c. Update the current WTA context with the WTA event parameters, if any, as defined in section 10~~9~~.3.

d. Using the current WTA context, process the content indicated by the URI specified in the `href` attribute of the first `resource` element in the `channel` associated with the event. (See [WML] for more detail on processing content.)

e. If the invocation succeeds, go to step ~~8~~6.

f. If the invocation fails…

   i) Terminate the current WTA context.

   ii) Go to step ~~7~~6.

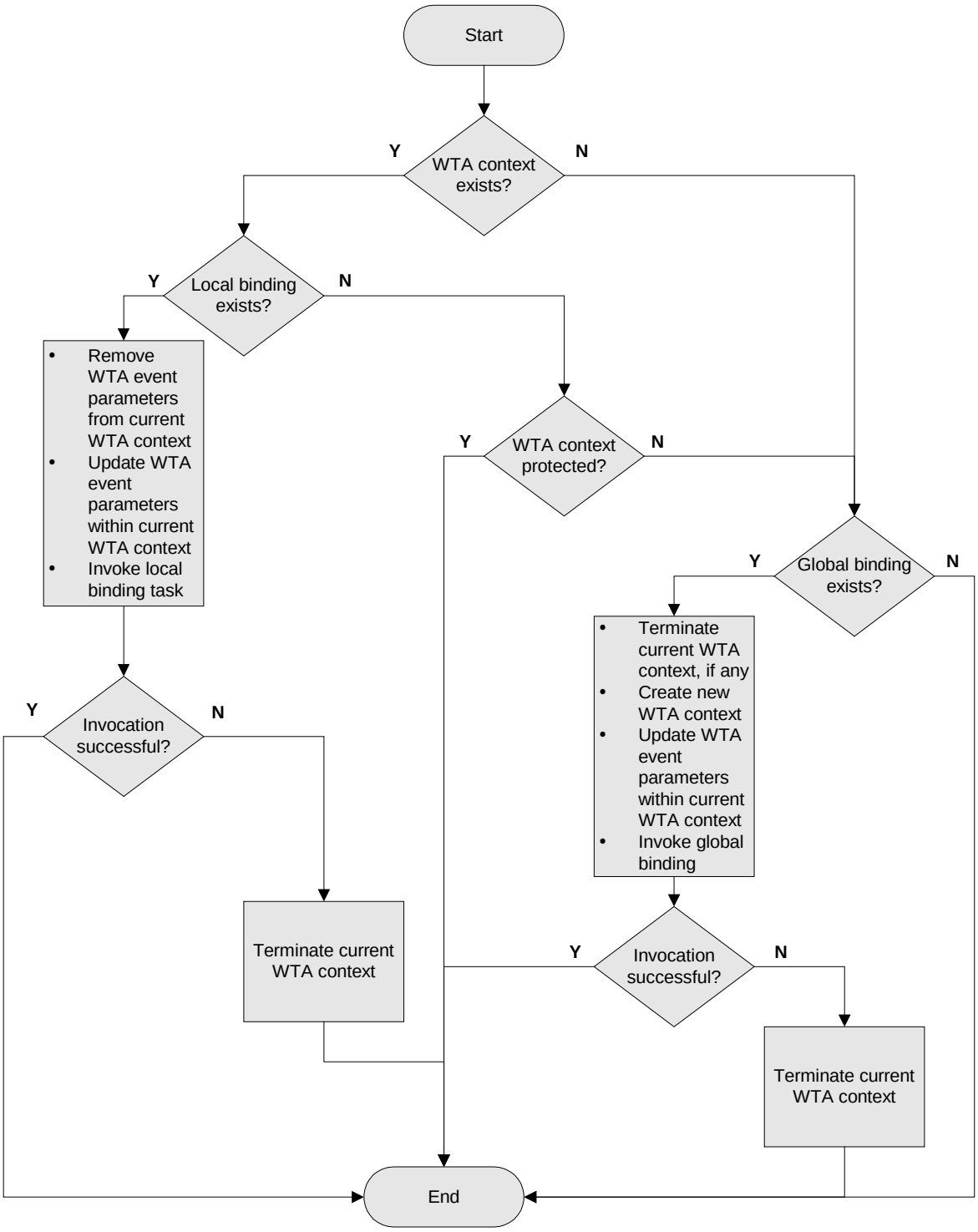
~~6.~~5. The WTA context, if any, may be terminated as defined in section 7.

~~7.Defer the network event to the default MMI.~~

~~8.~~6. End of processing of the WTA event.

~~WTA does not define how the default MMI processes network events. For example, the default MMI may terminate the existing WTA context~~.

This process is illustrated in the following flowchart:



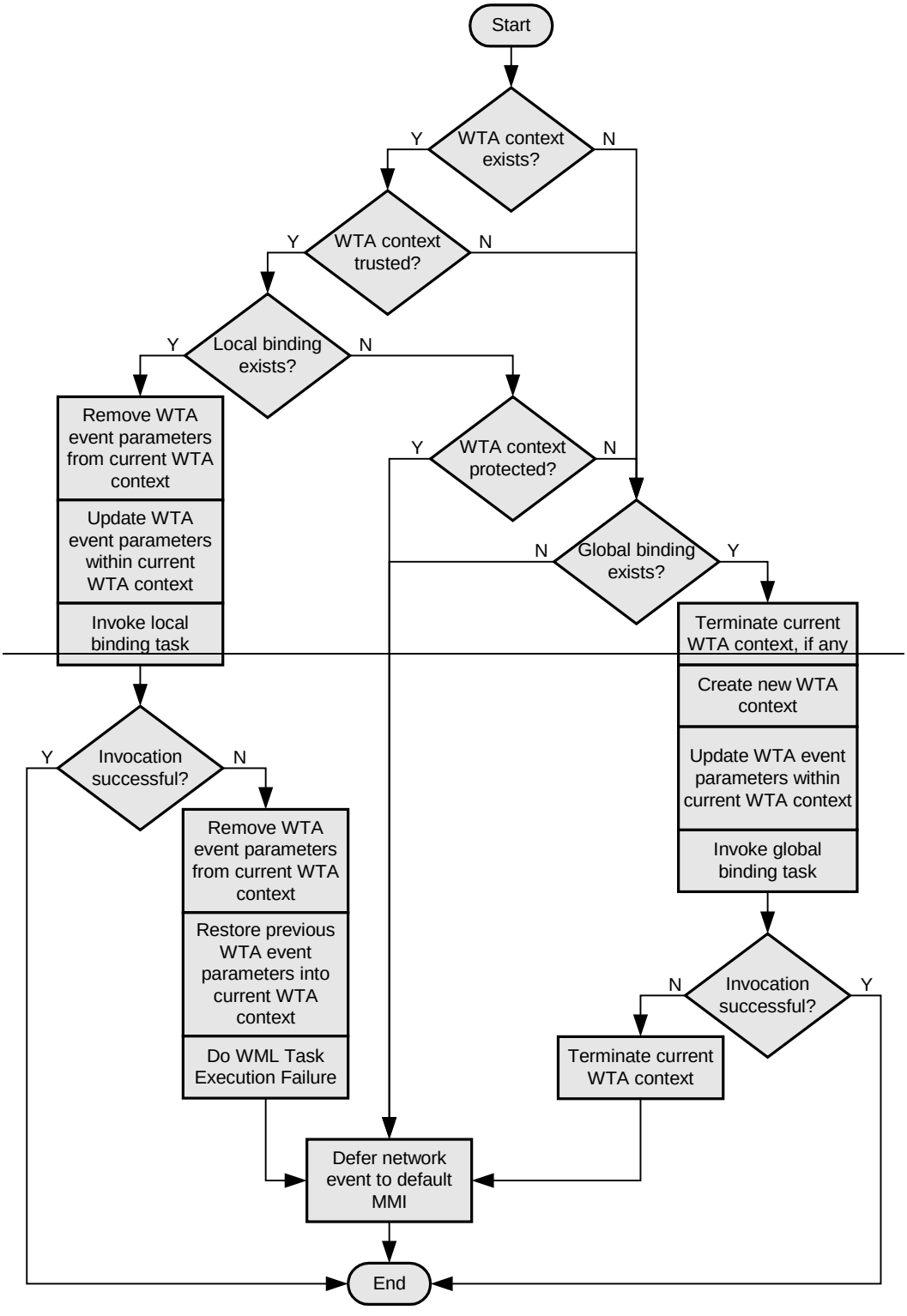


Figure 1 – WTA Event Handling Flowchart

Annex 2 – Static Conformance Requirements

A 2.1.6 Event Handling

Item	Functionality	Reference	Status	Requirement
WTA_EV_C001	Network events are transformed to WTA events.	10.1	M	
WTA-EV-C-001	The WTA user agent responds to WTA events using local and global bindings.	10.2	M	
WTA_EV_C007	Support for the use of the WTA-WML onevent element with WTA events to implement temporary event bindings.	10.2, 10.3	M	
WTA-EV-C-002	A local binding is specified using the WTA-WML `onevent` element.	9.2	M	
WTA-EV-C-003	Scope and shadowing semantics of local bindings are as defined for WML.	10.2	M	WML:MCF (WML-08)
WTA_EV_C008	Support the use of channels to represent global event bindings.	10.2, 10.4	M	
WTA-EV-C-004	A global binding is specified using a WTA `channel` element.	10.2	M	
WTA_EV_C009	Support for event parameters and event parameter referencing using index notation.	10.5	M	
WTA-EV-C-005	A WTA event parameter is referenced using an indexing notation.	10.3	M	
WTA-EV-C-006	A reference to a non-existent WTA event parameter is resolved as defined in WML.	10.3	M	WML:MCF
WTA-EV-C-007	WTA event parameter references are legal anywhere variable references are legal in WML.	10.3	M	WML:MCF
WTA-EV-C-008	A WTA event parameter is read-only and cannot be the target of a setvar element.	10.3	M	
WTA-EV-C-009	When navigating as a result of a local binding, the WTA user agent enforces the WML `access` element of the target deck using the URI of the referring card.	10.5	M	WML:MCF
WTA-EV-C-010	When navigating as a result of a global binding, the WTA user agent enforces the WML `access` element of the target deck using the URI of the channel specifying that global binding.	10.5	M	WML:MCF
WTA-EV-C-011	It is illegal for a single WTA user agent to have multiple concurrent WTA contexts.	10.6	M	

Item	Functionality	Reference	Status	Requirement
WTA-EV-C-012	The WTA user agent only processes WTA events when in a stable state.	10.6	M	
WTA-EV-C-013	The WTA user agent processes WTA events in the same order as they were received.	10.6	M	
WTA-EV-C-014	The WTA user agent terminates the current context if it cannot handle WTA events due to e.g. resource limitations.	10.6	M	
~~WTA_EV_C002~~	~~Order of precedence when processing a WTA event.~~	~~10.2~~	~~M~~	
~~WTA_EV_C003~~	~~Support for interruption of a WTA context.~~	~~10.2.1~~	~~M~~	
~~WTA_EV_C004~~	~~Interrupted context is terminated if there is a global binding to the detected event.~~	~~10.2.1~~	~~M~~	
~~WTA_EV_C005~~	~~Interrupted context is terminated if no global binding exists and fallback to standard MMI is used.~~	~~10.2.1~~	~~MMI:O~~	
~~WTA_EV_C006~~	~~Interrupted context is resumed.~~	~~10.2.1~~	~~O~~	
~~WTA_EV_C010~~	~~Support for fallback handling meaning the ability to forward unbound WTA events to the standard MMI.~~	~~10.2, 10.6.1~~	~~MMI:M~~	
~~WTA_EV_C011~~	~~Support for fallback handling meaning the ability to forward WTA events to the standard MMI if content loading fails.~~	~~10.6.1~~	~~MMI:M~~	
WTA-EV-C-015	The WTA user agent implements the event handling process.	10.6	M	

Annex 3~~2~~ - Using Events

This annex is informative only. It describes implementation aspects that a WTA content developer should consider when writing WTA services that use WTA events. The ~~T~~two ~~of the~~ WTA models ~~in~~ presented in this Annex are specific, distinct, and extreme approaches to implementing WTA services; the content developer is advised to mix the two approaches as appropriate.

A 3~~2~~.1 Sharing Data

Authors should consider the limitations of sharing data when constructing applications. Consider an application that has no local bindings but rather uses global bindings for all events as depicted in Figure 2. In that diagram, a box represents a WML card and the arrows show navigation between cards (and possibly decks). The WTA application can thus be thought of as a sequence of card navigation resulting from user interaction, inherent events (e.g., timeouts) and WTA events.

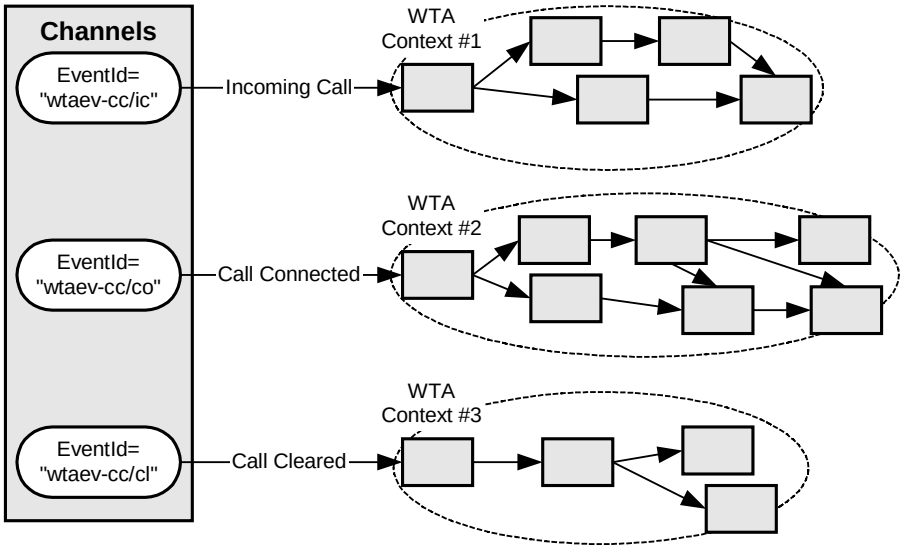


Figure 2 – Application Model Using Only Global Bindings

As depicted in the figure, each global binding may result in a number of cards being rendered as the application executes. However, when a WTA event occurs, a new global binding is invoked and this results in a new context according to the event handling process (see Section 109.6).

Since all context data is lost in the new context, data (e.g., call ids, caller info, and user inputs) can not be shared between the different phases of the single application, i.e., between Context #1, #2, and #3 in the figure.

A 32.2 Event Handler Lifetime

Authors should consider an application's lifetime when constructing applications. Consider an application that has only one global binding and uses local bindings to navigate between cards as depicted in Figure 3. The WTA application can still be thought of as a sequence of card navigation resulting from user interaction, inherent events (e.g., timeouts) and WTA events.

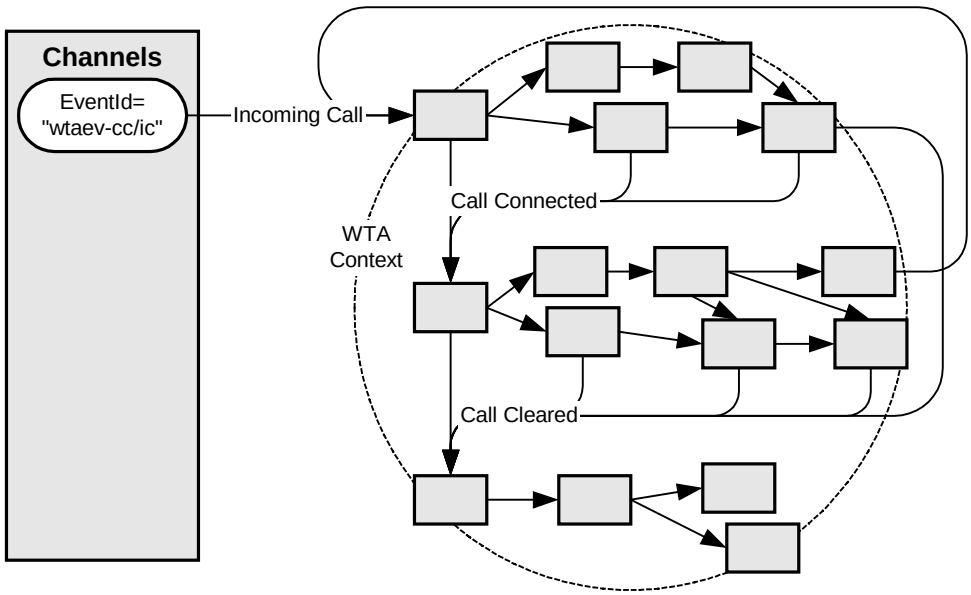


Figure 3 – WTA Application Model Using Only One Global Binding

Since only one context ever exists (the one created when the global binding to the Incoming Call event was processed), this application model allows data to be shared among the various cards and decks of the application. However, its expected lifetime would force the author to have to deal with other events such as incoming messages, etc.

As such, it is most likely that each card will need to be able to handle a wide array of WTA events. Applications built using this model could be difficult and time-consuming to maintain. ~~Race conditions may exist.~~

~~A 2.3 Maintaining sync between WTA Call Model and underlying implementation~~

~~A device running a WTA application can be viewed as having three layers and each of these layers can be considered to be a state machine:~~

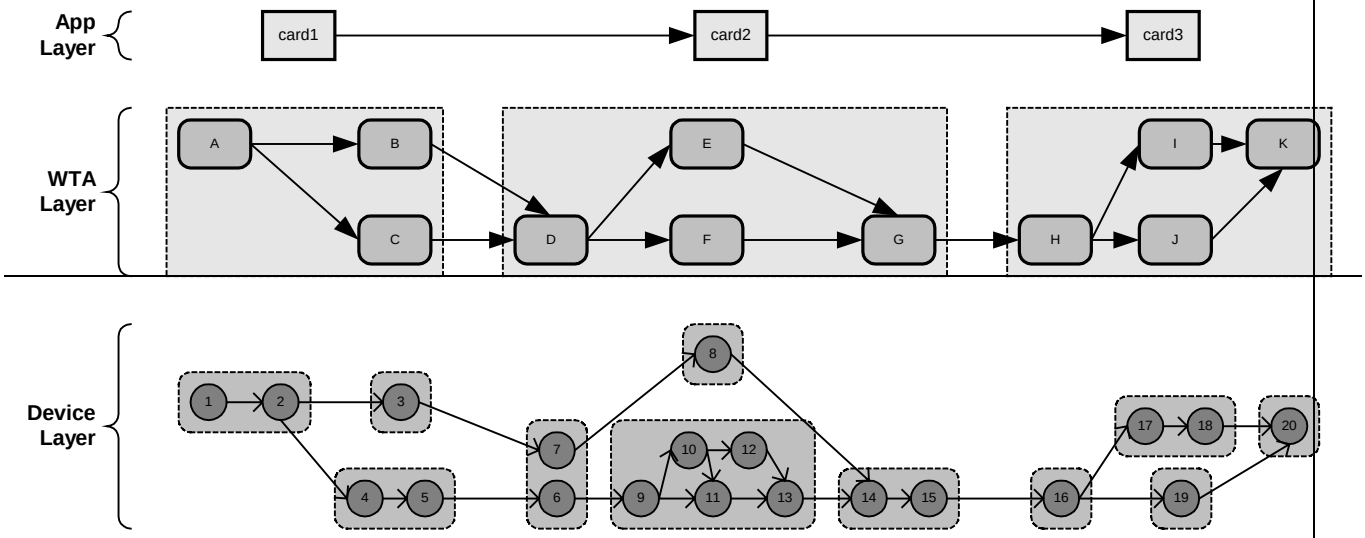
~~1.The **application layer** is the user's view into the WTA application. It is programmed by the author and consists of WML and WMLScript content. Each WML card can be considered a state within an overall state machine that represents the entire application. Navigations between cards can be considered state transitions.~~

~~2.The **WTA layer** is the implementation of the WTA states and events as specified in [WTAI].~~

~~3.The **device layer** is a representation of the telephony operation as implemented by each device or manufacturer. A device may not use an actual state machine per se; however, every device can be conceptualized with an underlying state machine.~~

~~When the device layer transitions between states, the WTA layer may transition between states as well. This may result in a WTA event that may in turn cause the application layer to transition between states. However, the WTA event is only processed if the WTA user agent is in a stable state. Thus, if the WTA user agent is processing one WTA event and another WTA event occurs as a result of a device layer transition, the second WTA event is not immediately processed. This delay of processing means that the underlying device layer will be in a different state than the WTA and application layers above. In most cases, the upper layers will eventually "catch up" to the device layer, however, the temporary loss of synchronization may cause some WTAI functions to fail (i.e., return an error code).~~

~~For example, consider an application depicted in the three layer representation of Figure 4. This application consists of three WML cards, eleven WTA states, and twenty device states.~~



~~Figure 4 – Sample WTA Application Layers~~

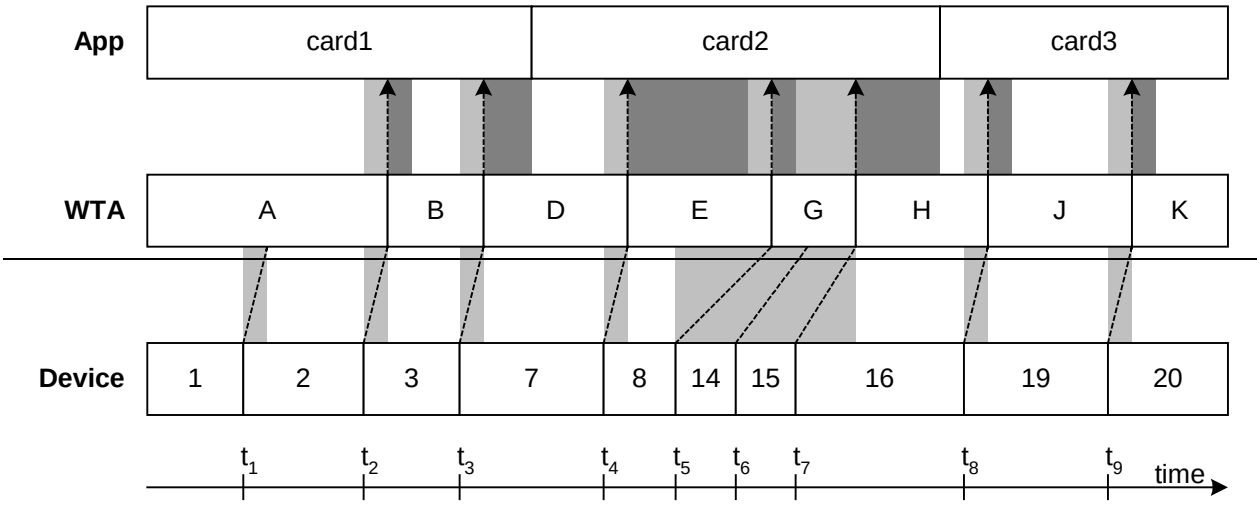
~~Given this state diagram, one possible sequence of state transitions for the three layers is shown in Figure 5. The grayed areas indicate the time during which the WTA and application layers are out of sync with the device layer:~~

~~– At time t_1, the device transitions from state 1 to state 2 but there is a slight lag before the WTA layer completes its internal processing of that underlying transition. Since the WTA layer stays in state A, no WTA event occurs. The~~

~~light gray area in the figure indicates the interval during which the application and WTA layers are out of sync with the device layer.~~

- ~~At time t_2, the device transitions from state 2 to state 3 and again there is a slight lag before the WTA layer completes its internal processing. When the WTA layer eventually transitions from state A to state B, it generates a WTA event but that WTA event does not affect the application layer (perhaps it had no local or global binding for that WTA event). The dark gray area in the figure indicates the interval during which the WTA user agent is unstable and can not process events.~~
- ~~At time t_3, the device transition causes a WTA state change and the WTA user agent is unstable for a longer period of time (perhaps because it is executing a WML Script function).~~
- ~~At time t_4, the device transition causes a WTA event that keeps the WTA user agent in an unstable state for a long time. As shown, the WTA user agent remains unstable during device transitions at time t_5 and t_6, forcing the associated WTA events to be delayed even further. These delayed WTA events are queued during the unstable period, then are processed in rapid succession (with no idle time between them).~~
- ~~Eventually, the WTA and application layers "catch up" with the device layer so that by time t_8, device transitions are processed in the WTA and application layers with minimal delay.~~

~~This simple example shows how it is possible that the device layer may transition several states and become out of sync with the WTA and application layers. Authors should inspect the WTAI return codes and recover from such cases.~~



~~Figure 5 – Example State Transitions~~

4. Bug fix to user-agent behaviour when user denies permisson to access WTAI URI

4.1 Change Classification

Class 2 – Bug Fixes

4.2 Change Summary

Section "6.3 User Permissions" specifies ambiguous behaviour of the WTA user-agent when the user does not grant permission to invoke the URI version of a WTAI function. Currently the specification says that in such a case an empty string should be returned. However, the WTAI specification of 07-Jul-2000 states that the empty string is used to indicate successful invocation (see section "8.3 URI functions" of that specification). Therefor, the right user-agent behaviour is instead to return an invocation error, which is represented by the string "-200", when the user does not grant permission to function invocation via a WTAI URI.

4.3 Change Description

Editor's note: This change only proposes changes to the section 6.3 User permissions, page 14.

6.3 User permissions

User permission shall be given for all WTAI (public and non-public) function calls performed by executables. An executable is any ~~entity which~~entity that calls WTAI functions.

If a WTAI function is invoked and the user does not grant permission, the invocation must return immediately without any side effects from the invocation and without any change within the device as a result of the attempted execution, as if the WTAI function had not been invoked. In this case, the WMLScript version of the WTAI function must return invalid and the URI version of the WTAI function must return an ~~empty string~~invocation error [WTAI].

5. Removing arbitrary requirement on storage capabilities

5.1 Change Classification

Class 2 – Bug Fixes

5.2 Change Summary

Event parameters are passed to the WTA user-agent along with WTA events. In section 7.3 of the WTA specification there is a statement saying, “A user-agent MUST have enough space to hold at least 10 event parameters and 250 symbols (e.g., characters).”

The statement forces a kind of requirement that is unnecessary and difficult to define for a WTA user-agent. Among the currently defined WTA events there is no event that passes more than four event parameters and the actual length of parameters varies between events. Future defined WTA events may pass more or less than four parameters and those parameters may require more or less space. For forward compatibility, the WTA specification should not say anything about space requirements for WTA event parameters. It should be up to each implementation to provide the space capabilities needed for successfully executing services that use any of the WTA events defined in WTAI, including Network Specific WTAI where applicable.

This change proposes removing the statement.

5.3 Change Description

Editor's note: This change proposes changes to the section 7.3 Event Parameters, and to Annex 2 Static Conformance Requiremenst, section A 2.1.3 State Model, table Event Parameter Management.

Editor's note: On page 17.

7.3 Event Parameters

Event parameters are transported with WTA events as defined in [WTAI]; e.g. caller id is transported with the incoming call event. How to reference an event parameter is defined in section 10.5.

The following principles MUST be applied:

- The WTA context retains the actual parameters of the last bound event received. A bound event is one that has a task associated with it in the given WTA context or has an associated channel present in the repository.
- Upon receiving a WTA event, the user-agent must set the event parameter state according to the following steps:
    1. If the WTA event has a task (including `NOOP`) temporarily bound in the current WTA context, the user-agent must clear all event parameters and then assign them new values based on the WTA event prior to invoking the bound task.
    2. Otherwise, if the WTA event is globally bound and the user-agent intends to react to the WTA event, the user-agent must clear all event parameter variables and then assign them new values based on the WTA event prior to initiating a new WTA context to deal with the WTA event.
    3. Otherwise, the user-agent must not alter the event parameter state.

~~A user-agent MUST have enough space to hold at least 10 event parameters and 250 symbols (e.g., characters).~~

Annex 2 - Static Conformance Requirements

A 2.1.3 State Model

Event Parameter Management

Item	Functionality	Reference	Status	Requirement
WTA-EPM-C-001	The WTA context retains the actual parameters of the last bound event received.	7.3	M	
WTA-EPM-C-002	The WTA user-agent sets event parameter state.	7.3	M	
~~WTA_EPM_C003~~	~~Support for storage of at least 10 event parameters and 250 symbols (e.g. characters).~~	~~7.3~~	~~M~~	

6. Event handling clarifications

6.1 Change Classification

Class 2 – Bug Fixes

6.2 Change Summary

There is uncertainty as for the problem of how and when the mobile phone should indicate to the user (e.g. by ringing) that an incoming call has arrived.

On one side, the Incoming Call Selection sample service in the WAP spec requires that the phone indicate that a call has arrived, so that the user can further be notified of menu options to select. On the other hand, in another sample service, the Voice Mail service, it is clearly stated that the "client answers the call automatically", implies that no ring should occur.

In the old WTA Temporary Event Binding (10.3) it specified that "An existing WTA service can be programmed to handle a WTA event within the existing WTA context. **Any other event handling mechanism (standard MMI or repository) must be overridden by temporary event bindings**". In the approved Event Handling CR (CREC-WTA-M-N-E-29-NOV-2000.doc), which was submitted and accepted as an editorial change, there is no such note. Thus, in order to match the new Event Handling to the old spec and to provide the ability to support the sample services given in the spec the following should be stated:

1. A service that is bound to an event must replace the standard MMI handling for that event.

2. Any MMI activity (e.g. ring) should be explicitly invoked by the service.

6.3 Change Description

Editor's note: This change proposes changes to section 10.6 Event Handling Process as it is described in the CR CREC-WTA-MOTOROLA/NOKIA/ERICSSON-29-Nov-2000, also included in this SCD.

10.6 Event Handling Process

A single WTA user agent instance must not allow more than one WTA context to exist at a time; it is illegal for a WTA user agent to have multiple concurrent WTA contexts.

A service that is executed upon detection of a WTA event, MUST replace the default MMI handling for that event. Any MMI functionality that is required by the service must be explicitly invoked by the service content.

The step-by-step process below describes the reference model for handling of WTA events by the WTA user agent. All WTA user agents must implement this process, or one that is indistinguishable from it. The WTA user agent must not begin processing of any WTA event unless it is in a stable state as defined in section 10.4. If a network event which has a corresponding WTA event occurs while the WTA user agent is in an unstable state, the implementation should delay processing of the WTA event until the WTA user agent is in a stable state. Any delayed WTA events must eventually be processed in the same order as they were received.