Contents

1. SCOPE **5**

2. REFERENCES **6**

2.1. NORMATIVE REFERENCES **6**
2.2. INFORMATIVE REFERENCES **7**

3. TERMINOLOGY AND CONVENTIONS **8**

3.1. CONVENTIONS **8**
3.2. DEFINITIONS **8**
3.3. ABBREVIATIONS **10**

4. INTRODUCTION **11**

4.1. ARCHITECTURAL OVERVIEW **11**
4.1.1. The WTA and WAE User-Agents 11
4.1.2. WTA Server 12
4.2. WTA SERVICES **12**
4.2.1. Initiation of WTA Services 13
4.3. ACCESS TO REPOSITORY **13**
4.3.1. How to Access the Repository 13

5. WTA SECURITY REQUIREMENTS **15**

5.1. SECURITY DELEGATION **15**
5.2. ACCESS CONTROL **15**
5.3. USER PERMISSIONS **16**
5.4. WTA SECURITY MODEL **17**
5.5. AVAILABLE SECURITY FRAMEWORK **17**

6. STATE MODEL **18**

6.1. SESSION MANAGEMENT **18**
6.1.1. Start-up of a WTA Session 18
6.2. USER-AGENT CONTEXT **18**
6.2.1. WTA Context Lifecycle 19
6.2.2. The Newcontext Attribute 19
6.2.3. The Endcontext function 19
6.3. EVENT PARAMETERS **19**
6.4. SERVICE INDICATION **20**
6.5. CALL STATE MANAGEMENT **20**

7. WTA-WML CONTENT **21**

7.1. WTA-WML TO WML2 TRANSFORMATION **21**
7.2. WTA-WML CONTENT FORMAT **21**
7.2.1. Document Identifiers 21
7.2.2. WTA-WML Element Semantics 21
7.2.3. The WTA-WML DTD 22
7.2.4. WBXML Tokens 23
7.2.5. Gateway Considerations 24

8. REPOSITORY **25**

8.1. CHANNEL LOADING **26**
8.2. CHANNEL UNLOADING **26**
8.3. REPOSITORY GC **26**
8.4. PROGRAMMING THE REPOSITORY **27**
8.4.1. The Channel Content Format 27
8.4.2. Example Channel 32
8.4.3. Channel Installation 34
8.5. REPOSITORY ACCESS POLICY **35**

9. EVENT HANDLING **36**

9.1. DESCRIPTION 36
9.2. EVENT BINDINGS 36
9.3. EVENT PARAMETER REFERENCE 37
9.4. STABLE STATE 37
9.5. EVENT-BASED INVOCATION AND ACCESS CONTROL 37
9.6. EVENT HANDLING PROCESS 38
APPENDIX A. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) 40
A 1.1 CLIENT FEATURES 40
A 1.2 SERVER FEATURES 48
APPENDIX B. EXAMPLES OF WTA SERVICES (INFORMATIVE) 50
B 1.1 INTRODUCTION 50
B 1.2 INCOMING CALL SELECTION 51
B 1.3 VOICE MAIL 52
APPENDIX C. USING EVENTS (INFORMATIVE) 54
C 1.1 SHARING DATA 54
C 1.2 EVENT HANDLER LIFETIME 54
APPENDIX D. CHANGE HISTORY (INFORMATIVE) 56

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 providing new 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 specification defines the Wireless Telephony Application (WTA) framework and the WTA user-agent. The WTA framework supports Wireless Telephony Applications that interface with the in-device telephony related functions and the network telephony infrastructure.

The WTA framework extends the WAE framework by adding:

- An interface from WML and WMLScript to a specific set of local, telephony related, functions in the client. This interface is called the "Wireless Telephony Application Interface" [WTAI].
- Network event handling. This means that events originating from the mobile network could be detected by the WTA user-agent and actions in response to the events could be defined.
- A repository, which is a storage container, used by the WTA user-agent, that persistently stores content that executes WTA services. The purpose of the repository is to fulfil the real-time requirements that are placed on the execution of WTA services.
- A model for WTA user-agent state and WTA context management.
- A MANDATORY security model.

2. References

2.1. Normative References

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

[CACHE] WAP-120, “WAP Caching Model Specification”, WAP Forum™, 13-April-2001 URL: http://www.wapforum.org/

[RFC2046] "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", N. Freed, et al., November 1996 URL: http://www.ietf.org/rfc/rfc2046.txt

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax." T. Berners-Lee, R. Fielding, L. Masinter, et al., August 1998 URL: http://www.ietf.org/rfc/rfc2396.txt

[RFC2616] "Hypertext Transfer Protocol - HTTP/1.1". R. Fielding,et al. June 1999 URL: http://www.ietf.org/rfc/rfc2616.txt

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

[SI] WAP-167, "WAP Service Indication Specification", WAP Forum™, 31-Jul-2001 URL: http://www.wapforum.org/

[WAE] WAP-236, “Wireless Application Environment Specification” , WAP Forum™ URL: http://www.wapforum.org/

[WBXML] WAP-192, “Binary XML Content Format Specification”, WAP Forum™, 25-Jul-2001 URL: http://www.wapforum.org/

[WDP] WAP-259, “Wireless Datagram Protocol Specification”, WAP Forum™, 14-Jun-2001 URL: http://www.wapforum.org/

[WML1] WAP-155, “Wireless Markup Language Specification”, WAP Forum™, 04-Nov-1999 URL: http://www.wapforum.org/

[WML2] WAP-238, “Wireless Markup Language Version 2.0”, WAP Forum™ URL: http://www.wapforum.org/

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

[WSP] WAP-230, “Wireless Session Protocol Specification”, WAP Forum™, 05-Jul-2001 URL: http://www.wapforum.org/

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

[WTAIGSM] WAP-255, “Wireless Telephony Application Interface Specification, GSM specific addendum”, WAP Forum™, 08-Sep-2001 URL: http://www.wapforum.org/

[WTAIIS136] WAP-269, “Wireless Telephony Application Interface Specification, ANSI 136 specific addendum”, WAP Forum™, 08-Sep-2001 URL: http://www.wapforum.org/

[WTAIIS95] WAP-228, “Wireless Telephony Application Interface Specification, IS95 specific addendum”, WAP Forum™, 08-Sep-2001 URL: http://www.wapforum.org/

[WTAIPDC] WAP-270, “Wireless Telephony Application Interface Specification, PDC specific addendum”, WAP Forum™, 08-Sep-2001 URL: http://www.wapforum.org/

[WTLS] WAP-261, “Wireless Transport Layer Security Specification”, WAP Forum™, 06-Apr-2001 URL: http://www.wapforum.org/

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

2.2. Informative References

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

[RFC2376] "XML Media Types", E Whitehead, et al., July 1998 URL: http://www.ietf.org/rfc/rfc2376.txt

[WMLTR] WAP-244-WMLTR, “WML Transformations”, WAP Forum™ 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

Author - an author is a person or program that writes or generates WML, WMLScript or other content.

Card - a single WML unit of navigation and user interface. May contain information to present on the screen, instructions for gathering user input, etc.

Channel –a named collection of resources with a well-known entry point.

Client – is a device (or service) that initiates a request for connection to a server.

Content - subject matter (data) stored or generated at an origin server. Content is typically displayed or interpreted by a user-agent in response to a user request.

Content Encoding - when used as a verb, content encoding indicates the act of converting a data object from one format to another. Typically the resulting format requires less physical space than the original, is easier to process or store, and/or is encrypted. When used as a noun, content encoding specifies a particular format or encoding standard or process.

Content Format – actual representation of content.

Context – an execution space where variables, state and content are handled within a well-defined boundary.

Deck – a collection of WML cards. A WML deck is also an XML document.

Default MMI – Non-WAP-specific behaviour of a device under specific circumstances. For example, user interface functionality of a mobile terminal in response to a network event if no WTA service is loaded.

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

Event Parameters - variables that convey data specific to a WTA event instance and can be accessed using WML2 [WML2].

Global Binding – an association between a WTA event and a dedicated resource; e.g. a channel in the repository.

Local Binding - a WTA event binding that is specified using WML2 [WML2] content.

Network Event – An event or activity related to the mobile network, for example an incoming call, which occurs outside of any WTA context and which may be conveyed to the WTA user-agent in the form of a WTA event and subsequently processed within a WTA context.

Origin Server – is the server on which a given resource resides or is to be created. Often referred to as a web server or an HTTP server.

Pull - when the client requests content from a server, i.e. a client initiated content delivery.

Push - when a server sends content to a client without the client requesting it, i.e. a server initiated content delivery.

Repository - a persistent storage containing resources collected into channels.

Repository GC – Repository Garbage Collection is another term for reclaiming memory.

Resource – is network data object or service that can be identified by a URL. Resources may be available in multiple representations (e.g. multiple languages, data formats, size, and resolutions) or vary in other ways.

Server - a device (or service) that passively waits for connection requests from one or more clients. A server may accept or reject a connection request from a client. A server may initiate a connection to a client as part of a service (using push technology).

Service Indication – A content type that is used to cause a user-agent to indicate to the user that an external, asynchronous event has occurred in an application. It lets a WTA server dynamically deploy new services to a WTA device.

Stable State – A condition in which the WTA user-agent context is active, but the user-agent is effectively idle – i.e. no loading, navigation or WMLScript activity is taking place.

Terminal - a device typically used by a user to request and receive information. Also called a mobile terminal or mobile station.

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

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

Web Server - a network host that acts as an HTTP server.

WML - the Wireless Mark-up Language is a hypertext mark-up language used to represent information for delivery to a narrow-band device, e.g. a mobile phone.

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

WTA Client - a client supporting WTA services.

WTA Content – content retrived from a WTA server within a WTA session.

WTA Context - specifically the context of the WTA user-agent.

WTA Event - the representation of a network event, e.g. an incoming call, as conveyed to the WTA user agent.

WTA Framework – a framework, based on the WAE framework, which contains functions in order to support WTA services.

WTA Server - an origin server that provides WTA services.

WTA Service - a service that is executed within a WTA session. It is allowed for a WTA service to use the functions that are included in the WTA framework.

WTA Session - a WSP session created by the WTA user-agent over one of the WDP ports dedicated for WTA. Both connection-mode and connectionless WSP session services are used for WTA sessions (over separate WDP ports) [WSP].

WTA User-agent - an extension of the WAE user-agent, that uses the WTA framework.

WTA-WML - the WTA Wireless Mark-up Language has the same capabilities as WML1 [WML1] but extends WML1 with WTA specific features for handling event parameters.

XML - the Extensible Mark-up Language is a World Wide Web Consortium (W3C) recommended standard for Internet mark-up languages, of which WML is such a language. XML is a restricted subset of SGML.

3.3. Abbreviations

CGI	Common Gateway Interface
DTD	Document Type Definition
GC	Garbage Collection
HTTP	Hypertext Transfer Protocol
IN	Intelligent Network
MSISDN	Mobile Station International Subscriber Device Number
RFC	Request For Comments
URI	Uniform Resource Identifier
URL	Uniform Resource Locator [RFC2396]
WAE	Wireless Application Environment
WAP	Wireless Application Protocol [WAPARCH]
WDP	Wireless Datagram Protocol [WDP]
WSP	Wireless Session Protocol [WSP]
WTA	Wireless Telephony Applications
WTAI	Wireless Telephony Applications Interface
W3C	World Wide Web Consortium
XML	Extensible Mark-up Language [XML]

4. Introduction

4.1. Architectural Overview

WTA is an application framework for telephony services. The WTA user-agent is an extension to the standard WML user-agent with the addition of capabilities for interfacing with mobile network services available to a mobile telephony device, e.g. setting up and receiving phone calls. The figure below (Figure 1) describes one *possible* configuration of the WTA framework. However, this specification solely defines the components contained in the client.

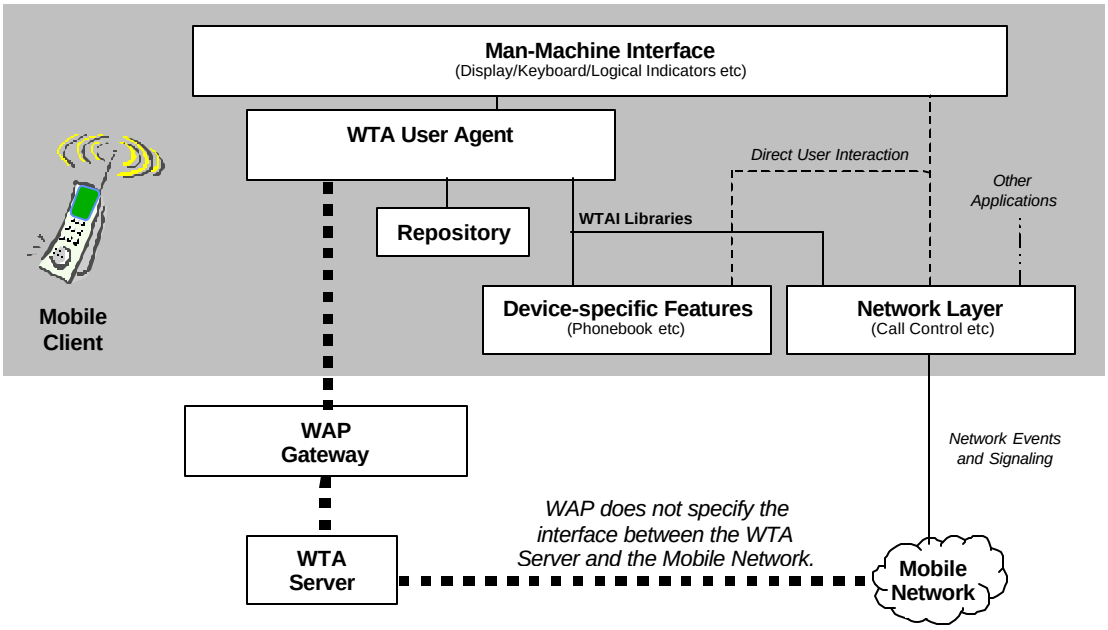


Figure 1, Overview of the WTA architecture

It is recognised that the ability to support simple telephony functions from within a WAE user-agent is also very important. With this in mind a special WTA Library, the WTAI Public Library has been defined. This library contains functions which can be called from any WAE application (Figure 2) and provides access to simple telephony functionality as an aide to the user experience. For instance it allows WML authors to include "click to phone" functionality within their content, to save users from typing a number using the default Man Machine Interface.

As mentioned above, the WTA framework relies on a dedicated WTA user-agent in the client, briefly described in section 4.1.1. The WTA server is not specified by WAP, but a brief overview is given in section 4.1.2.

4.1.1. The WTA and WAE User-Agents

Figure 1 above illustrates how the WTA user-agent, the repository (persistent storage) and WTAI (telephony application interface) interact with each other and other entities in a WTA-capable mobile client device. The WTA user-agent is able to retrieve content from the repository and WTAI ensures that the WTA user-agent can interact with mobile network functions (e.g. setting up calls) and device specific features (e.g. manipulating the phonebook). The WTA user-agent receives "network events" that can be bound to content, thus enabling dynamic telephony applications.

Network events that will be available to the WTA user-agent are those that are the result of actions taken by services running in the WTA user-agent itself. Telephony events initiated from outside the device are also passed to the WTA user-agent, as are network text message events that are not explicitly directed towards another user-agent (e.g. events intended for a SIM). This means, for instance, that network events caused by the WML user-agent will not affect the WTA user-agent.

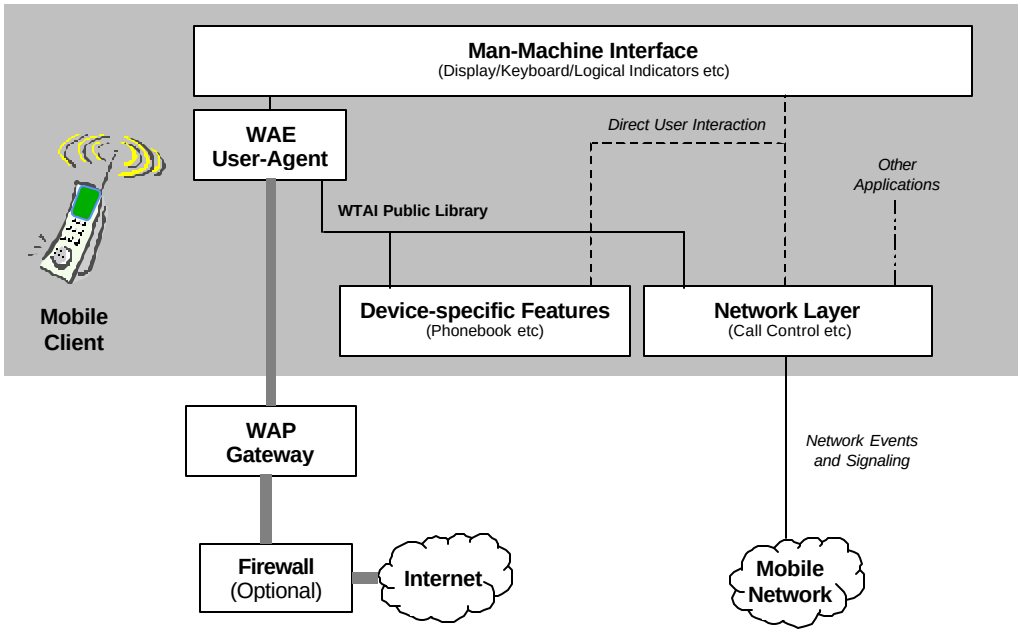


Figure 2, The WAE User Agent and WTA Public Library

Figure 2 above illustrates how the WAE user-agent and WTAI (Telephony Application Interface) Public Library interact with each other and other entities in a WTA-capable mobile client. The WAE user-agent only retrieves its content via the WAP gateway and only has access to the WTAI Public Library functions. These functions expose simple functionality such as the ability to place a call, but do not allow fully featured telephony control. Only a WTA user-agent is able to fully control the telephony features of the device. In particular, the WAE user-agent is not able to receive and react to telephony and network text events.

Note that Figure 1 and Figure 2 show logical separations of the two user-agents. They could co-exist on the same device and are likely to be implemented with common code elements.

4.1.2. WTA Server

The WTA server can be thought of as a web server delivering content requested by a client. Like an Internet web browser, a WTA user-agent uses URLs to reference content on the WTA server.

A URL can also be used to reference an application on a web server (e.g. a CGI script[1]) that is executed when it is referenced. Such applications can be programmed to perform a wide range of tasks, for example generate dynamic content and interact with external entities.

A WTA server may also make use of this concept. By referencing applications on a WTA server it is possible to create services that use URLs to interact with the mobile network (e.g. an IN-node) and other entities (e.g. a voice mail system). Thus, the concept of referencing applications on a WTA server provides a simple but yet powerful model for how to seamlessly integrate services in e.g. the mobile network with services executing locally in the WAP client.

4.2. WTA Services

WTA services are what the end-user ultimately experiences from using the WTA framework. A WTA service appears to the client in the form of various content formats, e.g. WML2 [WML2], WMLScript etc. The WTA user-agent

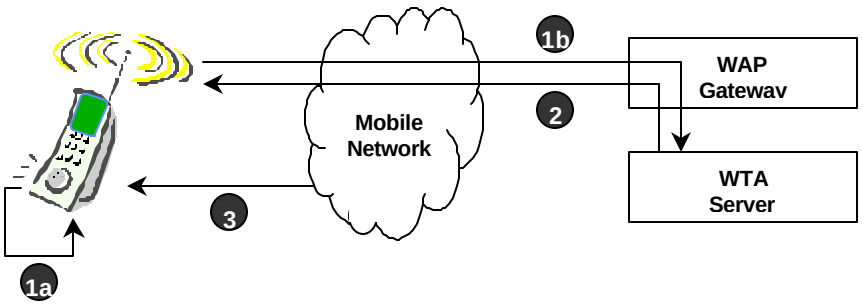
[1] Essentially the CGI script is an executable entity that produces content as output.

executes content that is persistently stored in the client's repository or content retrieved from a WTA server. The framework also allows the WTA user-agent to act on events from the mobile network (e.g. an incoming call).

4.2.1. Initiation of WTA Services

The WTA user-agent essentially executes content within the boundary of a well-known context. The term service is used to define the extent of a context and its associated content. Initiation of a new context is a definite start of a service. Termination of a context marks the definite end of a service.

The figure below shows the possible ways of initiating a WTA service in the WTA user-agent.



Explanations of numbered items:

1a Access to a URL (via the repository)

1b Access to a URL (via the WTA server)

2 Service Indication (Push)

3 Network event (transformed to WTA event in client)

Figure 3, Initiation of WTA services

4.3. Access to Repository

The repository is a persistent storage module within the mobile terminal that may be used to eliminate the need for network access when loading and executing frequently used WTA services. The repository also addresses the issue of how a WTA service developer ensures that time-critical WTA events (see section 8) are handled in a timely manner.

The repository addresses two specific issues:

- How does the WTA service developer pre-program the device with content?
- How does the WTA service developer improve the response time for a WTA service?

4.3.1. How to Access the Repository

The repository can be accessed by a service using one of the following methods:

- A WTA event (see section 9) may be associated with a channel. When a WTA event is detected, the user-agent will invoke a URL as specified by the associated channel.
- The end-user may access services stored in the repository through an implementation dependent representation (e.g. a menu containing the labels of the channels, see section 8) of the allowed services (channels explicitly specified as "user accessible" by the channel definition) in the repository.

- A URL may be given to the user-agent (provided in content or delivered by a Service Indication [SI]). The content for this URL may be retrieved from the repository.

Only WTA applications (that is, content loaded or otherwise received from the WTA server) may access the repository.

5. WTA security requirements

A WTA service can invoke WTAI-functions that enable access to local functions in the mobile client. Since such functions make it possible to e.g. set up calls and access the users local phonebook, it must be ensured that only authorised WTA services are allowed to execute.

5.1. Security delegation

It is within every trusted mobile telephony service provider's interest to provide an acceptable level of security in the network. The trusted mobile telephony service provider can choose

- to run all WTA services itself (allow no other providers),
- or to delegate the administration of its WTA services to a third party.

5.2. Access Control

The Wireless Datagram Protocol [WDP] provides a means for the implementation to separate a WTA service from a common WAE service by using predefined port numbers. The figure below illustrates the mandatory configuration.

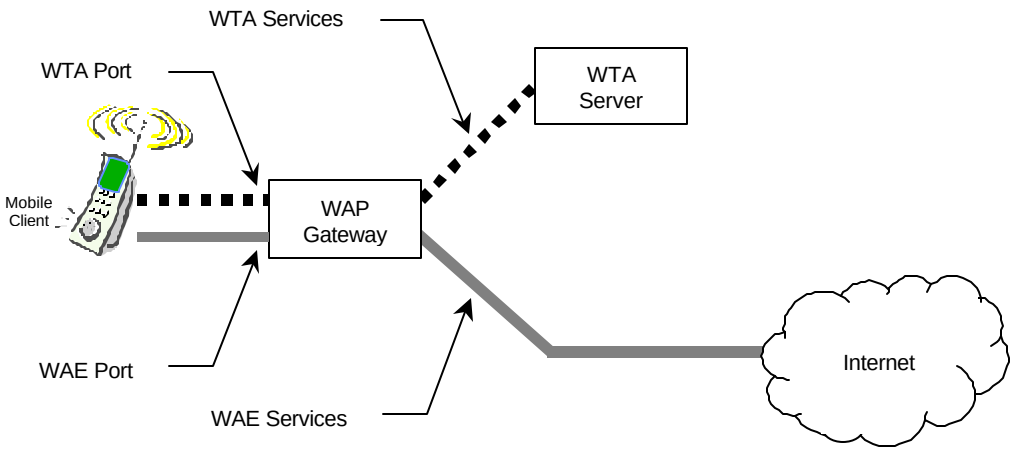


Figure 4, WDP Port Numbers and Access Control

A WTA session, established by the WTA user-agent, MUST utilise one of the dedicated, secure WTA ports on the gateway. The WTA user-agent MUST NOT retrieve WTA content outside the WTA session. WTA content received outside the WTA session, and Service Indications addressing the WTA user-agent but delivered outside a WTA session, SHALL be discarded.

5.3. User permissions

User permission SHALL be given for all WTAI (public and non-public) function calls performed by executables. An executable is any entity 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 invocation error [WTAI].

The user gives permission for a specific WTAI function call by an executable. Blanket permisson and single permission MUST be supported, context permission MAY be supported. (see Table 1).

Each WTAI function definition will specify which permission types it will support. In general, only single action permission will be supported by public WTAI functions.

	User Permissions		
Permission Type	**Description**	**Invocation**	**Revocation**
blanket permission	The user gives blanket permission to the executable for the specified WTAI function, and the executable subsequently uses the user's original permission for the identified subsequent WTAI functions whenever the executable is running.	Typically such permission would be given at executable configuration or run time.	The blanket permission may be revoked by the user at any time. The user permission no longer applies once the executable has been removed.
context permission	The user gives permission to the executable for the specified WTAI function during a specific run time context of an executable, and the executable subsequently uses the user's permission for the identified subsequent WTAI functions whilst the executable context is still running.	Typically such permission would be given at executable run time.	The context permission may be revoked by the user at any time. The user permission no longer applies once the executable run time context has terminated.
single action permission	The user gives a single permission to the executable for the specified WTAI function; if the executable subsequently wishes to repeat the WTAI function it must again request the user's permission for the identified subsequent WTAI function.	Typically such permission would be given at executable run time.	The user permission no longer applies once the WTAI function has terminated.

Table 1, User Permissions

5.4. WTA Security Model

In the WTA security model, any entity may become a WTA Service Provider by being approved for access to a trusted gateway. Access control of the trusted gateway by the WTA servers should be enforced using existing secure solutions.

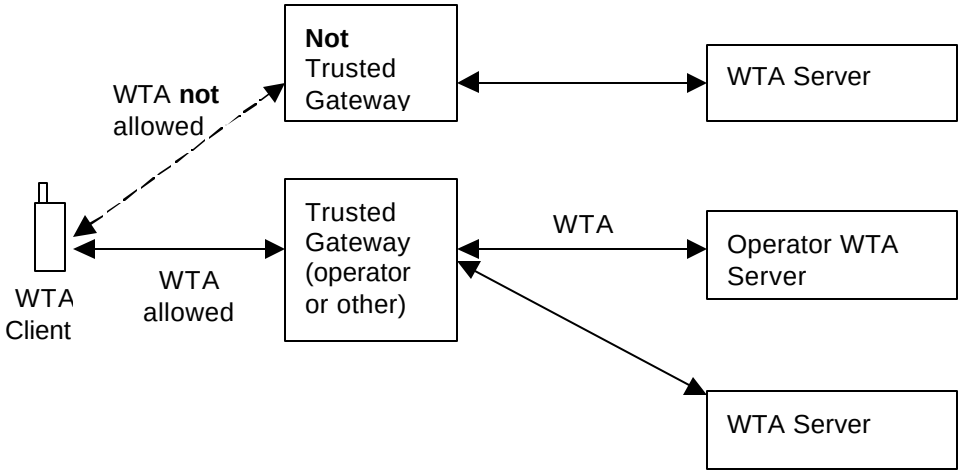


Figure 5, WTA Security Model

In order to provide security to WTA, a WAP gateway may control the access between the WTA user-agent and the WTA server. The WAP gateway should verify that the providers of WTA pull/push content are authorised.

5.5. Available security framework

A security mechanism is mandatory for WTA. Until a signed content model is available, the security between the client and the WTLS connection end point MUST be ensured by using WTLS class 2 with mandatory use of server certificate authentication. The use of a WTLS session is mandatory for WTA services (non-public). For a description of WTLS, see [WTLS].

Using WTLS, any WTA service can be delivered by the trusted telephony service provider (or an entity delegated by the trusted telephony service provider) through a secure pipe from the gateway to the client. WAP gateway to WTA server security is left up to the implementation. For connections over the Internet, SSL/TLS may be used.

The client MUST only allow connections for WTA services to specified (trusted) gateways.

6. State Model

The WTA user-agent is a telephony-capable extension of the standard WAE user-agent. The WTA user-agent follows the rules and methods supported by the WAE user-agent and hence it can be initiated in the same way as a WAE user-agent. Consequently, WTA utilises one or more WSP sessions [WSP] over a dedicated WTA port [WDP] and the same state model as in [WML2]. WTA can use WSP connectionless session services as well as connection-mode session services [WSP].

WTA extends the WAE model further by adding support for managing WTA user-agent specific elements such as:

- Network events - a framework for the handling of network events, e.g. managing event parameters (see section 6.3).
- Call state management – the coupling between a call and the context of a WTA service (see section 6.5).
- Content repository – access to a client-side storage container that houses content, thus minimising the need for network access (see section 8.5).

WTA also uses Service Indications [SI] to facilitate server-initiated services.

6.1. Session Management

WTA uses a WSP session [WSP], which is called the WTA session. The term WTA session is used in this document to denote the use of WSP connection-mode or connectionless session services over a secured dedicated WDP port. This section describes the required procedures for WTA session management between a WTA client and a WAP gateway.

A WTA session is used by the WTA client to facilitate interaction with a WAP gateway that communicates with a WTA server. A WTA user-agent can have one or many WTA sessions simultaneously, e.g. one WTA session can be used for service execution and another session can be used to receive pushed service indications [SI]. When using connection-mode session services, the lifetime of a WTA session consists of the following phases:

- WTA session start-up.
- Content exchange over the WTA session between a WTA server and the WTA client.
- Termination of a WTA session [WSP]

When the WSP connectionless session services are used there is no explicit start-up or termination of a session. The parts involved, i.e. the WTA user-agent and the WAP gateway, assume a common context [WSP].

6.1.1. Start-up of a WTA Session

A new WTA session, using connection-mode session services, is established by the WTA user-agent by requesting the WSP layer to establish a new session. A common context is established between the WAP gateway and the WTA user-agent. The characteristics of the user-agent are conveyed to the WAP gateway as described in [WAE].

For the start-up of the WTA session the following information MUST be provided by the implementation:

- Required bearer (based on preference or availability)
- The WAP gateway address (bearer specific, for example MSISDN)
- The well known WTA port number on the gateway [WDP]

This information is also needed when the WSP connectionless session services are used.

6.2. User-agent Context

Like [WML2], WTA stores its context in a single scope called the *user-agent context*. The context is used to manage the user-agent's various states.

6.2.1. WTA Context Lifecycle

A new WTA context is initialised whenever the WTA user-agent is started. After which, the WTA context is reinitialised whenever:

- Any of the conditions defined in [WML2] occur,
- The content author indicates that the WTA context should be reinitialised or terminated,
- A WTA event occurs and the current WTA context does not bind a task to the event and the WTA user-agent is configured to react to the event,
- An error in the content is discovered.

If an error is encountered, the user-agent MUST terminate the current WTA context and indicate to the user that a content error occurred. The user-agent MUST NOT handle any additional events until the user acknowledges the error or any other service is initiated by a global binding. Any events that occur while the user-agent is waiting for user acknowledgement may be handled by the device's default MMI mechanisms.

6.2.2. The Newcontext Attribute

The `newcontext` attribute of the `wml:card` and the body element re-initialises elements of the current user-agent context. When the WTA user-agent processes a `newcontext` attribute, it must perform all the tasks defined in [WML2], in addition to this it MUST:

- Terminate the call if appropriate to the call mode (see section 6.5),
- Process ongoing calls according to the call mode (see section 6.5),
- Preserve the event parameters (see section6.3), and
- Preserve the contents of the repository (see section 6.3).

6.2.3. The Endcontext function

The `endcontext` function [WTAI] provides the author the means to indicate that the current WTA context is done and is no longer useful (see section 6.2.1). Upon processing an `endcontext` function, the user-agent MUST:

- Remove all variables defined in the current WTA user-agent context,
- Clear the navigational history state,
- Remove all content associated with the current WTA context from the active memory,
- Terminate the call if appropriate to the call mode,
- Process ongoing calls according to the call mode (see section 6.5),
- Clear the event parameters, and
- Preserve the contents of the repository.

The user-agent may choose to terminate provided that it has means to restart again upon a WTA event and can preserve the repository.

6.3. 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 9.3.

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.

6.4. Service Indication

Service Indications [SI] are used to enable the WTA server to notify the end-user about new content to be retrieved from the WTA server. A Service Indication may relate to messaging applications (such as voice mail or e-mail), but also to events in the mobile network.

6.5. Call State Management

WTA includes support for managing the call state associated with a call established in the context of a WTA service using the [WTAI] interface. The author of the WTA service can indicate the coupling between the established call and the current WTA context by setting the mode for the call state management to one of two modes:

`DROP` – indicates that the call state is tightly coupled with the WTA context in which the call was established. In the event of the WTA context terminates before the call is dropped, the user-agent MUST terminate the call.

`KEEP` – indicates no coupling between the call state and the WTA context in which the call was established. In the event of the WTA context terminates before the call is dropped, the user-agent MUST NOT terminate the call.

The call state of a given WTA context is only coupled to the particular call in progress that was established in that specific WTA context. Implementations MUST ensure that a call state mode is always bound to the intended call. However a given WTA context can get information about the other calls available to the WTA user-agent in the host device by using the List Call [WTAI] function. The call state mode in a particular WTA context MUST NOT impact any call established in any other context or by some other means or applications.

7. WTA-WML Content

The WTA-WML content format is deprecated, instead WML2 defined in [WML2] should be used to write WTA services.

WTA-WML markup language is a superset of WML1.

7.1. WTA-WML to WML2 transformation

This section is informative.

The mapping rule for transforming WTA-WML into WML2 is defined using XSLT [WMLTR] but the use of the XSLT directly to transform the content is not required.

7.2. WTA-WML Content Format

This section defines the content format used to represent wta-wml content.

WTA-WML is an application of [XML] version 1.0.

7.2.1. Document Identifiers

7.2.1.1. SGML Public Identifier

```
-//WAPFORUM//DTD WTA-WML 1.2//EN
```

7.2.1.2. WTA-WML Media Type

Textual form: `text/x-wap-wta-wml`

Tokenised form: `application/x-wap-wta-wmlc`

Required parameters: none

Optional parameters: `charset` and `version`

The `version` parameter indicates the WTA-WML version. The value has the format: <major>.<minor> - where major and minor are integers. For example, `version=1.2` indicates version 1.2. Should the version parameter not be available, the default version 1.2 applies.

The `charset` parameter is the same as specified in [RFC2046].

Encoding of the `text/x-wap-wta-wml` media type follows the same principles as specified for the XML media types in [RFC2376]. Encoding of the `application/x-wap-wta-wmlc media` type follows the same principles as specified for WBXML in [WBXML], section 5.2.

The media type may be used by WSP/HTTP for content negotiation between the WTA client and server. See [RFC2616].

7.2.2. WTA-WML Element Semantics

The WTA-WML DTD contains the WML1 DTD as a subdocument. This allows forward-compatibility with WML1 while also allowing the variant syntax required for WTA event parameters.

```
<!ENTITY % wml-dtd PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
   "http://www.wapforum.org/DTD/wml12.dtd">

%wml-dtd;
```

The syntax and semantics of all WML1 elements (see [WML1]) are identical for WTA-WML with the exceptions specified in the following sections.

7.2.2.1. The WTA-WML Element

```
<!ELEMENT wta-wml ( head?, template?, card+ )>
<!ATTLIST wta-wml
   xml:lang                NMTOKEN      #IMPLIED
   %coreattrs;
>
```

The `wta-wml` element is identical in purpose and usage to the `wml` element in [WML1]. It is merely a container for content that adheres to the semantics required for WTA services.

7.2.2.2. The SETVAR Element

The `setvar` element is syntactically and semantically identical to the `setvar` element in [WML1] except that the `value` attribute may contain WTA event parameter references. The BNF for event parameters is found in section 9.3.

7.2.2.3. The ONEVENT Element

The `onevent` element is syntactically and semantically identical to the `onevent` element in [WML1] except that the `type` attribute may contain a WTA event identifier as specified in [WTAI].

7.2.3. The WTA-WML DTD

```
<!--
Wireless Telephony Applications Wireless Markup Language (WTA-WML) Document Type
Definition.
WTA-WML is an XML language. Typical usage:
<?xml version="1.0"?>
<!DOCTYPE wta-wml PUBLIC "-//WAPFORUM//DTD WTA-WML 1.2//EN"
   "http://www.wapforum.org/DTD/wta-wml12.dtd">
<wta-wml>
...
</wta-wml>
-->

<!ENTITY % wml-dtd PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
   "http://www.wapforum.org/DTD/wml12.dtd">

%wml-dtd;

<!ELEMENT wta-wml ( head?, template?, card+ )>
<!ATTLIST wta-wml
   xml:lang                NMTOKEN      #IMPLIED
   %coreattrs;
>
```

7.2.4. WBXML Tokens

7.2.4.1. Global Extension Tokens

None defined.

7.2.4.2. Tag Tokens

WTA-WML uses all of the tag tokens in code page zero (0) of [WML1] plus the following tag token codes in code page one (1). All numbers are in hexadecimal.

Tag Name	***Token***
wta-wml	3F

7.2.4.3. Attribute Start Tokens

WTA-WML uses all of the attribute start tokens in code page zero (0) of [WML1] plus the following attribute start token codes in code page one (1). All numbers are in hexadecimal.

Attribute Name	***Attribute Value Prefix***	***Token***
Href	wtai://	05
Href	wtai://wp/mc;	06
Href	wtai://wp/sd;	07
Href	wtai://wp/ap;	08
Href	wtai://ms/ec	09
Type	wtaev-	10
Type	wtaev-cc/	11
Type	wtaev-cc/ic	12
Type	wtaev-cc/cl	13
Type	wtaev-cc/co	14
Type	wtaev-cc/oc	15
Type	wtaev-cc/cc	16
Type	wtaev-cc/dtmf	17
Type	wtaev-nt/	20
Type	wtaev-nt/it	21
Type	wtaev-nt/st	22
Type	wtaev-pb/	30
Type	wtaev-lg/	38
Type	wtaev-ms/	50
Type	wtaev-ms/ns	51
Type	wtaev-gsm/	58
Type	wtaev-gsm/ru	59

Type	wtaev-gsm/ch	5A
Type	wtaev-gsm/ca	5B
Type	wtaev-pdc/	60
Type	wtaev-ansi136/	68
Type	wtaev-ansi136/ia	69
Type	wtaev-ansi136/if	6A
Type	wtaev-cdma/	70

7.2.4.4. Attribute Value Tokens

WTA-WML uses all of the attribute value tokens in code page zero (0) of [WML1]. No additional tokens are defined.

7.2.5. Gateway Considerations

This section is informative.

The WAP Gateway must ensure that WTA-WML content has valid syntax, including event parameter variable references. The WTA-WML DTD and associated syntax rules specified herein must be enforced.

8. Repository

The repository is used to store WTA content persistently. This provides a mechanism that ensures timely handling of content related to WTA services (e.g. WTA services initiated by WTA events, see section 9) and has the following characteristics:

- The repository contains a set of *channels* and *resources*.
- Resources are data that have been downloaded with WSP (e.g., a WML2 document), and are stored along with their meta-data (e.g. content type and the HTTP 1.1 *entity-tag* [RFC2616]) and location (URL).
- A channel is a resource that contains a set of links to resources. Channels have an identity and freshness.
- Channels in the repository have a *freshness lifetime* (the HTTP 1.1 *expiry-date* header [RFC2616]), beyond which time they are considered *stale*. Stale channels are subject to automatic removal by the user-agent (see section 8.3). Resources are subject to automatic removal from the repository if a channel does not reference them (see section 8.3).
- If the repository contains a channel that is not stale, it is guaranteed that the repository contains all resources named in that channel. The loading and unloading of a channel is an atomic operation, in that no user-agent operation will recognise the presence of the channel until all of the content in the channel has been successfully stored in the repository.
- A label may be associated with a channel to give a textual description of the service indicated by the channel.

Resources in the repository may be referenced by more than one channel. A resource is present in the repository if one or more channels reference it.

The figure below shows an example of how channels may share resources stored in the repository.

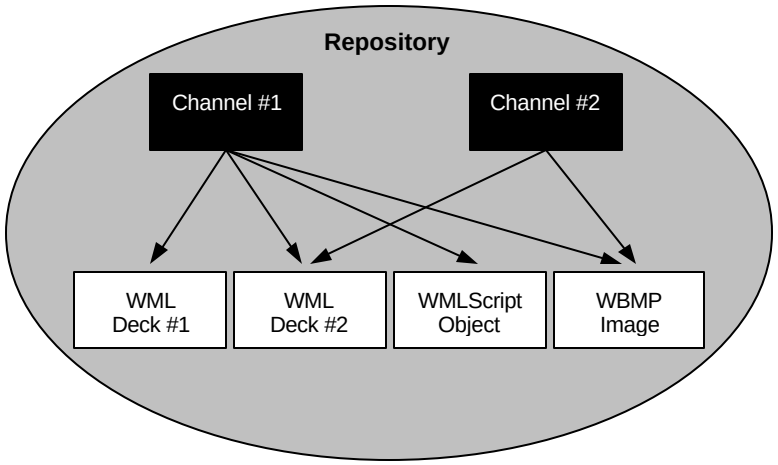


Figure 6, Repository

8.1. Channel Loading

Channels can be pushed into the repository or loaded into the repository whenever the user-agent retrieves them (see section 8.4.3). Channels are uniquely identified by the *channelid* attribute value. If the *channelid* of a new channel has a value which is identical to that of a channel which has been previously loaded, the new channel information takes precedence over (overwrites) the previous information. In the case where both the old and new channel elements contain global event bindings, the old binding will be replaced by the new binding, whether or not the *eventid* attribute value is identical.

If the *eventid* attribute of the new channel element has a value which is identical to that of a channel which has been previously loaded, the new channel replaces the previous channel, whether or not the *channelid* attribute values of the new and old channels are identical. This implies that the old global event binding is replaced (see Global Event Bindings).

Installation of a channel involves the following steps:

1. Always load a resource indicated by a channel *except* when the resource already exists in the repository *and* both the indicated and the stored resource are equally fresh.

    1. If the loading of all resources succeeds and a "success" URL is specified for the channel, a message is sent to the server (using the “success” URL). Request for the “success” URL tells the server that the channel is about to be activated. The response to the “success” URL request may contain a message to the end-user with information about the new service.

    2. Upon a successful reception of a response to the “success” URL or if no "success" URL exists for the channel, the channel is activated (becomes visible to the WTA user-agent). The channel is stored in the repository. All *new* resources are stored in the repository. All resources are updated with *new* attributes as indicated by the channel.

2. Else, if this loading fails for any reason (including if the "success" URL could not be accessed), the channel (along with any new resources) is discarded. If a previous channel has the same identity (stale or not), it (and its indicated resources) MUST be left unchanged.

    1. If the channel has the “failure” attribute, the “failure” URL must be requested. Request for the “failure” URL tells the server that the channel installation failed. If the response message fails to be delivered then the user-agent SHOULD proceed as if no “failure” URL exists in the channel.

    2. If no “failure” URL exists in the channel, the user-agent MUST present an appropriate error message to the end-user, e.g. “channel installation failed”.

Also see examples in section 8.4.3.2.

8.2. Channel Unloading

Loading an empty channel performs channel unloading. To unload a channel with a certain identity, an empty channel with the same identity should be loaded. The empty channel should thereafter be removed from the repository.

8.3. Repository GC

It is occasionally necessary to reclaim space in the repository. This should be done with the following process:

1. Remove all empty channels.

2. Remove all channels that are stale.

3. Remove all resources that are not referenced by a channel.

Editor’s note: The recommended minimum size of the repository will be subject to investigation after publication of this specification. The WTA group will work with the WAP-IOT group in trying to define a suitable size.

8.4. Programming the Repository

The repository is programmed by the delivery of a Channel document to the user-agent.

8.4.1. The Channel Content Format

This section defines the content format used to represent the channel content format.

Channel is an application of [XML] version 1.0.

8.4.1.1. Document Identifiers

8.4.1.1.1. SGML Public Identifier

```
-//WAPFORUM//DTD CHANNEL 1.2//EN
```

8.4.1.1.2. Channel Media Type

Textual form: `text/vnd.wap.channel`

Tokenised form: `application/vnd.wap.channelc`

Required parameters: none

Optional parameters: `charset` and `version`

The `version` parameter indicates the Channel version. The value has the format: <major>.<minor> - where major and minor are integers. For example, `version=1.2` indicates version 1.2. Should the version parameter not be available, the default version 1.2 applies.

The `charset` parameter is the same as specified in [RFC2046].

Encoding of the `text/vnd.wap.channel` media type follows the same principles as specified for the XML media types in [RFC2376]. Encoding of the `application/vnd.wap.channelc media` type follows the same principles as specified for WBXML in [WBXML], section 5.2.

The media type may be used by WSP/HTTP for content negotiation between the WTA client and server. See [RFC2616].

8.4.1.2. The Channel Element

```
<!-- Channel Events -->
<!ENTITY % ChannelEvent.attribs
   "success               %URI;                    #IMPLIED
    failure               %URI;                    #IMPLIED"
>
<!ENTITY % Boolean "(true|false)">

<!ELEMENT channel (title , abstract? , resource* ) >
<!ATTLIST channel
   maxspace               %Number;     #REQUIRED
   base                   %URI;                    #IMPLIED
   eventid                CDATA                    #IMPLIED
   channelid              CDATA                    #REQUIRED
   useraccessible         %Boolean;    "false"
   %ChannelEvent.attribs;
```

>

The channel element specifies a set of resources to be stored in the repository.

Attributes

maxspace=*%Number;*

The maxspace attribute specifies the maximum memory, in bytes, that is used by this channel. The server must guarantee that the sum of all the bytes in all the resources of the channel will not exceed this value. The client MAY opt to not install the channel should the maxspace be excessive or otherwise unacceptable (e.g. too large for its available memory).

base=*%URI;*

The base attribute specifies the base URL for the channel contents. If this attribute is specified, the location of resources in the channel may be specified using relative URLs. If this attribute is not specified, then the resources must all be identified using non-relative URLs.

success=*%URI;*

The success attribute (if present) specifies the URL that MUST be requested upon completion of the channel resource installation. The request for the URL tells the server that the channel is now ready to be activated, see section 8.4.3. The success attribute is optional. The value is a URI. If the success value is a URL, it may be relative to the channel's base attribute value (the base URL). If the channel does not have a base attribute, the success value is treated as an absolute URL.

failure=*%URI;*

The failure attribute (if present) specifies the URL that MUST be requested when a failure has been detected during channel installation. See section 8.4.3 on how to use the failure URL with the channel installation. The failure attribute is optional. The value is a URI. If the failure value is a URL, it may be relative to the channel's base attribute value (the base URL). If the channel does not have a base attribute, the failure value is treated as an absolute URL.

channelid=CDATA

The channelid attribute specifies the identity of the channel. This identity is used during channel loading and unloading (see sections 8.1 and 8.2). The channelid may alternatively identify a WTA service which is accessible by, for example, a menu selection.

eventid=CDATA

The eventid attribute (if present) specifies the WTA event that launches the service specified by the channel (see section 8). WTA event naming is specified in [WTAI].

useraccessible=(true|false);

The useraccessible attribute indicates whether the user can access this service manually, i.e. whether the user can invoke the channel through an implementation dependent representation (e.g. a menu containing the label of the channel).

8.4.1.3. The Title Element

```
<!ELEMENT title (#PCDATA)>
```

The title element specifies a short, human-readable title for the channel. It is suggested that this title be restricted to 12 characters or less for the purposes of displaying on limited function devices.

8.4.1.4. The Abstract Element

```
<!ELEMENT abstract (#PCDATA)>
```

The `abstract` element specifies a human-readable description of the channel. While the `abstract` element is optional, authors are encouraged to provide information that will help the end-user understand the purpose and value of this channel.

8.4.1.5. The Resource Element

```
<!ELEMENT resource EMPTY>
<!ATTLIST resource
   href                    %URI;         #REQUIRED
   lastmod                 %Number;      #IMPLIED
   etag                    NMTOKEN       #IMPLIED
   md5                     NMTOKEN       #IMPLIED
>
```

The `resource` element specifies a resource that is contained in a channel. The location of the resource is specified. The URL specified in the first resource element in a channel identifies the content to be invoked when the channel is referenced by some means.

Attributes

`href=`*`%URI;`*

The `href` attribute specifies the location of the resource. The value is a URI. If the `href` value is a URL, it may be relative to the channel's `base` attribute value (the base URL). If the channel does not have a `base` attribute, the `href` value is treated as an absolute URL.

`lastmod=`*`%number;`*

The `lastmod` attribute specifies the Last-Modified time of the current resource in the origin server. The user-agent may perform an "If-Modified-Since" calculation locally to determine if the resource in the repository is still fresh. If the user-agent maintains the Last-Modified time of a resource in the repository, it can compare that time to the `lastmod` attribute to determine whether the resource should be refreshed or not. The calculations for determining whether the content is still fresh are specified in [RFC2616]. If the user-agent does not maintain a resource's Last-Modified time or does not support the `lastmod` attribute, the user-agent MUST revalidate the resource with the origin server during channel installation if it has expired according to cache expiration algorithms [RFC2616]. The format of the `lastmod` attribute is an unsigned integer number representing the number of seconds from January 1, 1970, 00:00 UTC.

If the `lastmod` attribute is not specified, "If-Modified-Since" requests MUST be sent directly to the origin server.

`etag=`*`nmtoken`*

The `etag` attribute specifies the entity tag of the current resource in the origin server. The user-agent may perform an "If-Match", or "If-None-Match" calculation locally to determine if the resource in the repository is still fresh. If the user-agent maintains the entity tag of a resource in the repository, it can compare that entity tag with the `etag` attribute to determine whether the resource should be refreshed or not.

The format of the ETAG value is specified in [RFC2616] as the format of the "Etag" header value. The calculations for determining whether the content is still fresh are specified in [RFC2616] and [CACHE].

If the user-agent does not maintain a resource's entity tag or does not support the `etag` attribute, the user-agent MUST revalidate the resource with the origin server during channel installation if it has expired according to cache expiration algorithms in [RFC2616] and [CACHE].

If the `etag` attribute is not specified, "If-Match" and "If-None-Match" requests MUST be sent directly to the origin server.

`md5=`*`nmtoken`*

The `md5` attribute specifies the MD5 digest of the current resource in the origin server. The user-agent may perform a comparison between the MD5 digest of the resource in the repository with the MD5 attribute to determine if the content is still fresh.

The format of the MD5 value is specified in [RFC2616] as the format of the “Content-MD5” header value.

If the `md5` attribute is not specified or the repository does not maintain md5 for the resource stored in the repository, the user-agent MUST use HTTP/1.1 style cache validation algorithms to determine if the content is still fresh.

8.4.1.6. DTD

```
<!--
      This DTD is identified by the PUBLIC identifier:
            "-//WAPFORUM//DTD CHANNEL 1.2//EN"
-->

<!ENTITY % Boolean "( true | false )">

<!-- a Uniform Resource Identifier -->
<!ENTITY % URI "CDATA" >

<!-- one or more digits (NUMBER) -->
<!ENTITY % Number "CDATA" >

<!-- Channel Events -->
<!ENTITY % ChannelEvent.attribs
  "success               %URI;                    #IMPLIED
   failure               %URI;                    #IMPLIED"
>

<!ELEMENT channel (title , abstract? , resource* ) >
<!ATTLIST channel
  maxspace               %Number;     #REQUIRED
  base                   %URI;                    #IMPLIED
  eventid                CDATA                    #IMPLIED
  channelid              CDATA                    #REQUIRED
  useraccessible %Boolean;            "false"
  %ChannelEvent.attribs;
>

<!ELEMENT title (#PCDATA)>
<!ELEMENT abstract (#PCDATA)>

<!ELEMENT resource EMPTY>
<!ATTLIST resource
  href       %URI;              #REQUIRED
  lastmod                %Number;     #IMPLIED
  etag                   NMTOKEN      #IMPLIED
  md5                                 NMTOKEN      #IMPLIED
>
```

8.4.1.7. WBXML Tokens

8.4.1.7.1. Global Extension Tokens

None defined.

8.4.1.7.2. Tag Tokens

The following token codes represent tags in code page zero (0). All numbers are in hexadecimal.

Tag Name	***Token***
channel	5
title	6
abstract	7
resource	8

8.4.1.7.3. Attribute Start Tokens

The following token codes represent the start of an attribute in code page zero (0). All numbers are in hexadecimal.

Attribute Name	***Attribute Value Prefix***	***Token***
maxspace		5
base		6
href		7
href	http://	8
href	https://	9
lastmod		A
etag		B
md5		C
success		D
success	http://	E
success	https://	F
failure		10
failure	http://	11
failure	https://	12
eventid		13
eventid	wtaev-	14
channelid		15
useraccessible		16

8.4.1.7.4. Attribute Value Tokens

None defined.

8.4.2. Example Channel

This is an example of a simple WTA service specified using the WTA Channel.

```
<?xml version="1.0" ?>
```

```
<!DOCTYPE channel PUBLIC "-//WAPFORUM//DTD CHANNEL 1.2//EN"
   "channel.dtd">
<channel
   maxspace="2048"
   base="http://wap.operator.com/"
   eventid="wtaev-cc/ic"
   channelid="IncomingCallChannel"
   success="success.wml"
   failure="failure.wml"
>
   <title>Call Selection</title>

   <abstract>
     Incoming Call Selection service!
   </abstract>

   <resource href="welcome.wml" />
   <resource href="first.wml" />
   <resource href="script.wmls" />
</channel>
```

8.4.3. Channel Installation

A client supporting the channel content format MUST be able to install a channel at any time. The channel MUST be processed as soon as the client is idle (no service executing, i.e. a clean context and no content loaded) or be handled in the background. Channels can be loaded into the repository using any standard content transfer mechanism that is suitable to use with the specific type of network and bearer. Channel download methods may include:

- Returning the channel as part of a response to a standard URL request (GET or POST method).
- Pushing the channel to the device either directly or using a Service Indication [SI].

8.4.3.1. Completion of Channel Install

When all resources have been loaded and the channel is ready to be activated then the "success" URL is requested from the server. The response tells the client that the server has been informed about that the service is updated in the client. Then the client activates the channel (it becomes visible to the user-agent).

If any errors occur during the channel install process the installation MUST be terminated. The content indicated by the "failure" URL should then be loaded. If the "failure" URL is not present in the channel, or the request for it fails, the client MUST notify the end-user with an appropriate error-message.

8.4.3.2. Examples of Channel Installation

This section gives two examples of how installation of a channel may look like. The installation process has been illustrated (for practical reasons) from the point where the channel already has been introduced to the client. The first example shows a successful channel installation.

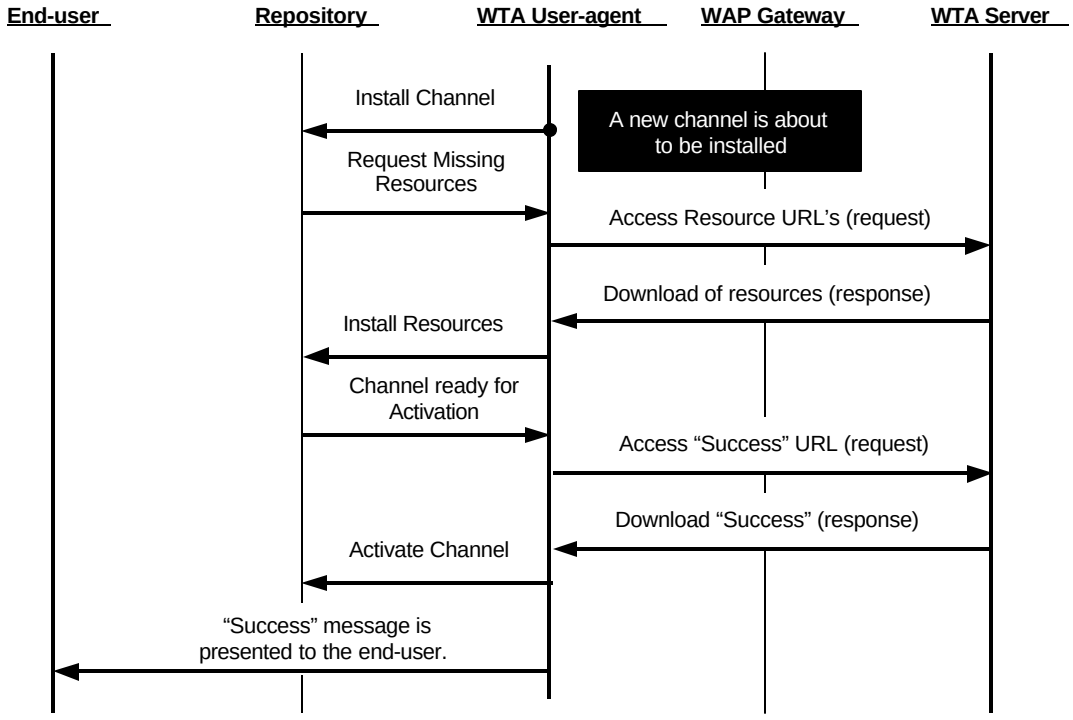


Figure 7, Successful Channel Installation

The following example shows some possible situations when a channel can not be installed due to an error condition.

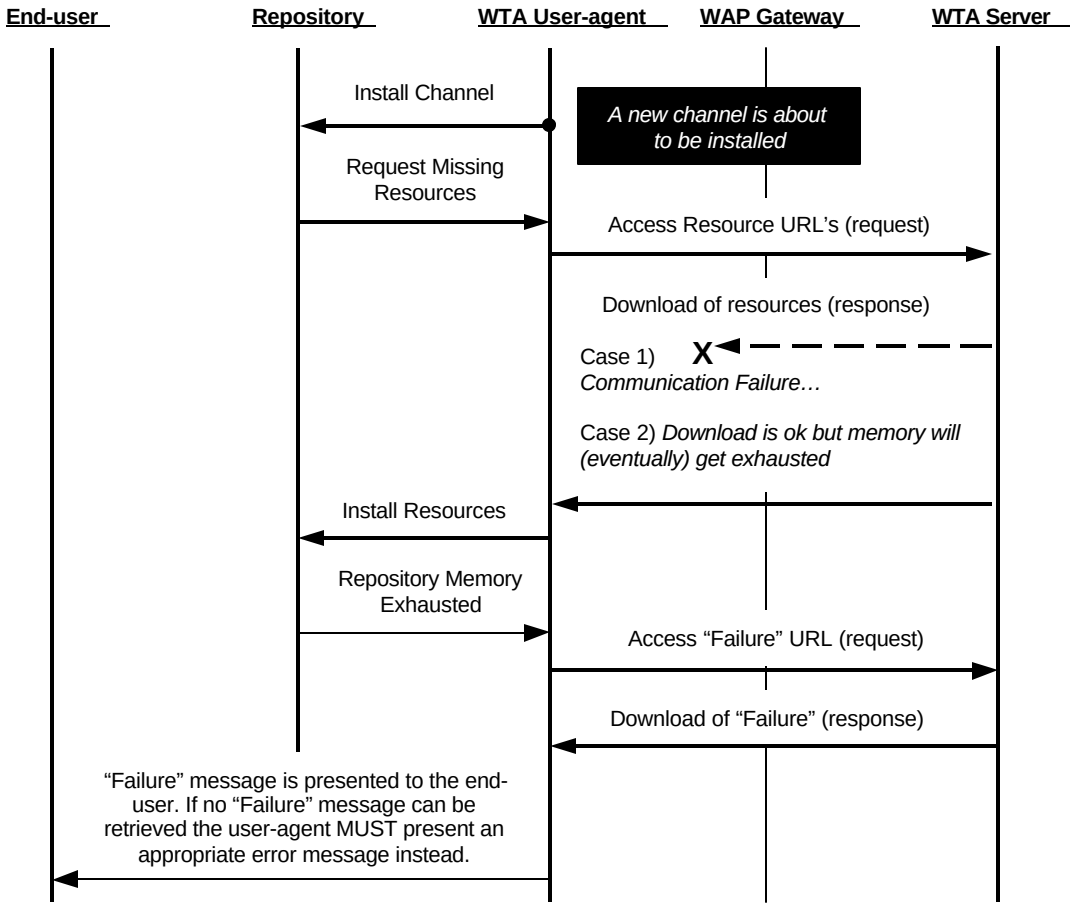


Figure 8, Failure during Channel Installation

8.5. Repository Access Policy

The repository is programmed (by the WTA service developer) to contain resources that in other cases might need to be retrieved from the network. Resources in the repository (not including channels) are stored together with its originating URL (the HTTP content location header [RFC2616]). This implies that an implementation MUST check the repository for a specific resource prior to requesting it from the network.

Implementation Note: If the WTA service developer wants to make sure that content on the WTA server is accessed, i.e. not content in the repository, then the URL for this service must be different from the service's URL in the repository.

9. Event Handling

9.1. 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 4.1.1).

The WTA user agent processes each WTA event according to the current WTA context. Typically, an author will bind a task (e.g., `wml:go`) to a WTA event type [WML2]. 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.

9.2. 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 or body. It is specified using the WML2 [WML2] `wml:onevent` element where the `type` attribute specifies the WTA event type (see [WML2] for more information on event bindings and see [WTAI] for a list of WTA event types).

The following example written in the WML2 [WML2] language shows a card, `c1`, that has a local binding between an Incoming Call event (i.e., `wtaev-cc/ic`) and a `wml:go` task:

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

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

Local bindings can be defined in `wml:card`, `body` and template elements. Card- and body- level bindings can shadow template-level bindings. Scope and shadowing semantics of local bindings (i.e., the `wml:onevent` element) are defined in [WML2]. Local bindings may not be invoked under certain conditions (see section 9.6 below).

In the absence of a local binding, the user agent may check (see section 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" eventid="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.

9.3. 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 body, card or deck (see [WML2] 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 [WML2].

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 [WML2]. 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”. However, such state may be overwritten by the next WTA event (see section 6.3). Since a subsequent WTA event will overwrite the event parameters, authors should copy the event parameters into WML variables to prevent loss of the values.

9.4. 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 9.6)
- It is not “processing content” (e.g., parsing WML2, rendering [WML2] , 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

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.

9.5. Event-based Invocation and Access Control

According to [WML2], the `wml:access` element of a WML document specifies access control information for the entire document. It contains domain and path attributes that specify which other resources (e.g., WML2 wml:card or `body` elements) may access the document. That is, access control for a target document 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 `wml:access` element of the target deck using the URI of the referring card. That is, the referring resource is the `wml:card` or `body` 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 `wml:access` element of the target document 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

9.6. 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 9.4. 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.

If the WTA user agent can not start processing a WTA event due to, e.g. resource limitations, the WTA user agent MAY terminate the current context. 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 4.
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 9.3.
   c. Invoke the task associated with the local binding using the current WTA context (see [WML2] for more detail on processing tasks).
   d. If the invocation succeeds, go to step 6.
   e. If the invocation fails…
      i) Terminate the current WTA context.
      ii) Go to step 6.
3. If the current WTA context is protected, it must remain unaffected…
   a. go to step 6.
4. If there is a global binding for the WTA event …
   a. Terminate the current WTA context, if any, as defined in section 6.
   b. Create a new WTA context as defined in section 6. 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 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 [WML2] for more detail on processing content.)
   e. If the invocation succeeds, go to step 6.
   f. If the invocation fails…
      i) Terminate the current WTA context.
      ii) Go to step 6.
5. The WTA context, if any, may be terminated as defined in section 6.
6. End of processing of the WTA event.

This process is illustrated in the following flowchart:

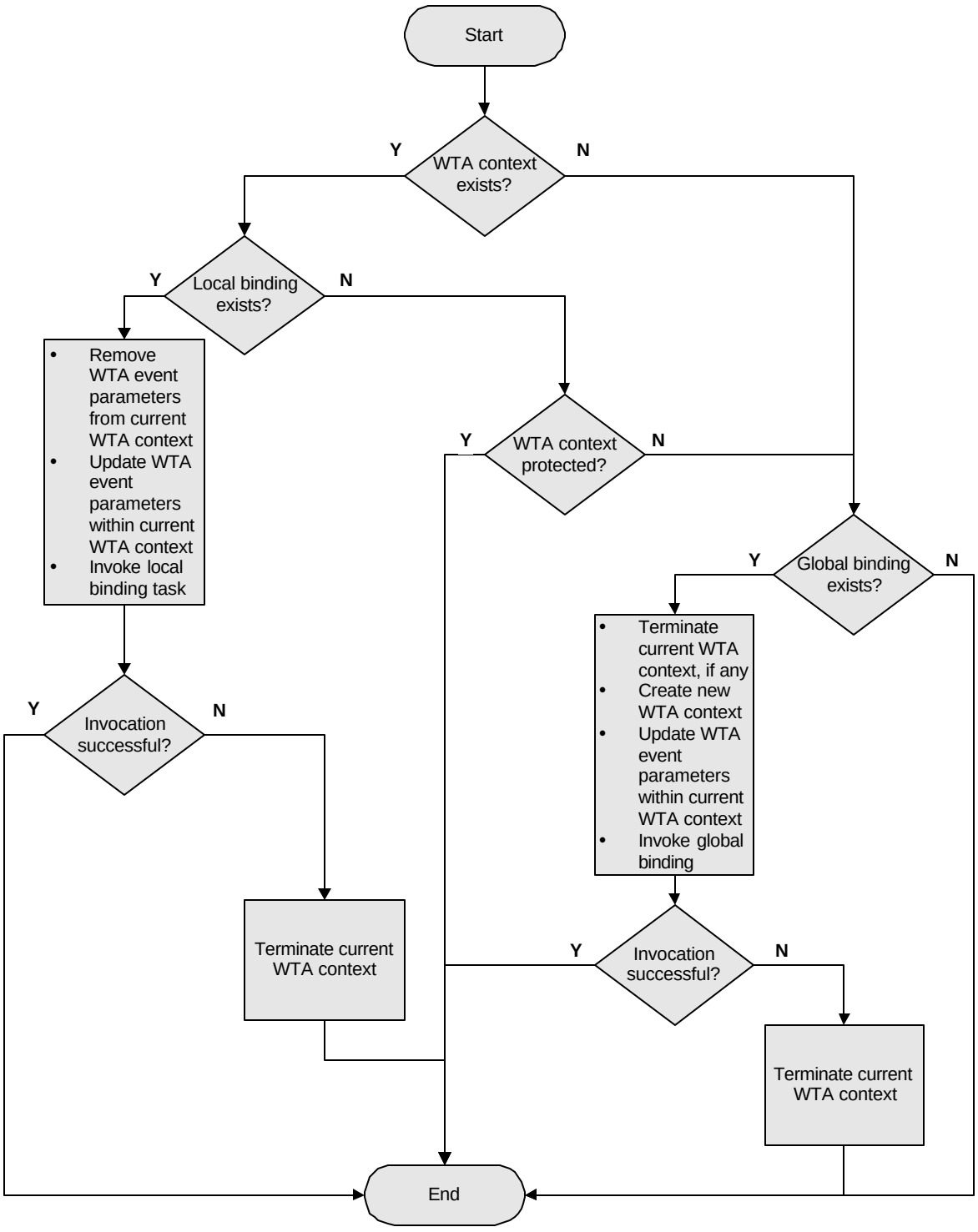


Figure 9 – WTA Event Handling Flowchart

Appendix A. Static Conformance Requirements (Normative)

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

A 1.1 Client features

A 1.1.1 Major Capabilities

Device Network Compliance

Item	Function	Reference	Status	Requirement
WTA-NETT-C-001	Network technology	-	M	WTA-GSM-C-001 OR WTA-ANSI136-C-001 OR WTA-PDC-C-001 OR WTA-IS95-C-001 OR WTA-ONETT-C-001
WTA-GSM-C-001	GSM technology	-	O	WTA-IF-C-003 AND WDP-GSM-C-000
WTA-ANSI136-C-001	ANSI 136 technology	-	O	WTA-IF-C-004 AND WDP-ANSI-C-000
WTA-PDC-C-001	PDC technology	-	O	WTA-IF-C-005 AND WDP-PDC-C-000
WTA-IS95-C-001	IS-95 technology	-	O	WTA-IF-C-006 AND WDP-CDMA-C-000
WTA-ONETT-C-001	Other network technology	-	O	WDP-CT-C-002 OR WDP-FLEX-C-001 OR WDP-CT-C-006 OR WDP-CT-C-008 OR WDP-TETRA-C-000 OR WDP-DECT-C-000 OR WDP-FLEX-C-002 OR WDP-MOBITEX-C-000

Device MMI

Item	Function	Reference	Status	Requirement
WTA-MMI-C-001	Default MMI functionality	-	O	

WTA Interface

Item	Function	Reference	Status	Requirement
WTA-IF-C-001	Public WTA Library	[WTAI]	M	WTAI-P-C-001 AND WTAI-INTP-C-001
WTA-IF-C-002	Network Common WTA Libraries	[WTAI]	M	WTAI-CV-C-001 AND WTAI-CT-C-001 AND WTAI-CP-C-001 AND WTAI-CL-C-001 AND WTAI-CM-C-001 AND WTAI-INTN-C-002
WTA-IF-C-003	Network Specific WTA Library for GSM	[WTAIGSM]	O	WTAIGSM:MCF
WTA-IF-C-004	Network Specific WTA Library for ANSI 136	[WTAIIS136]	O	WTAIIS136:MCF
WTA-IF-C-005	Network Specific WTA Library for PDC	[WTAIPDC]	O	WTAIPDC:MCF

Item	Function	Reference	Status	Requirement
WTA-IF-C-006	Network Specific WTA Library for IS-95	[WTAIIS95]	O	WTAIIS95:MCF

WTA User Agent

Item	Function	Reference	Status	Requirement
WTA-UA-C-001	WTA user-agent	4, 5, 6, 8 and 9	M	WAE:MCF

A 1.1.2 Security Requirements

Item	Function	Reference	Status	Requirement
WTA-SEC-C-001	Only authorised WTA services are allowed to execute.	5	M	WTLS:MCF AND WTLS-C-100
WTA-SEC-C-002	A WTA session utilises a dedicated, secure port on the WAP Gateway.	5.2	M	WTLS:MCF WTLS-C-100
WTA-SEC-C-003	The WTA user-agent only retrieves WTA content within a WTA session.	5.2	M	WTA-UA-C-001
WTA-SEC-C-004	WTA content received outside a WTA session is discarded.	5.2	M	
WTA-SEC-C-005	Support for blanket user permission for WTAI function calls.	5.3	M	
WTA-SEC-C-006	Support for context user permission for WTAI function calls.	5.3	O	
WTA-SEC-C-007	Support for single action user permission for WTAI function calls.	5.3	M	
WTA-SEC-C-008	The WTA user-agent allows WTA content only from trusted WAP Gateways.	5.4	M	WTA-UA-C-001 AND WTLS:MCF AND WTLS-C-100
WTA-SEC-C-009	Support for WTLS Class 2	[WTLS] and 5.5	M	WTLS:MCF AND WTLS-C-100

A 1.1.3 State Model

WSP Session ServicesItem	Function	Reference	Status	Requirement
WTA-SS-C-001	Support for WSP session services over WDP.	6.1	M	WTA-SM-C-001 OR WTA-SM-C-002

Session Management

Item	Function	Reference	Status	Requirement
WTA-SM-C-001	Support for WSP connectionless session services over a dedicated WDP port.	6.1	O	WSP-CL-C-001
WTA-SM-C-002	Support for WSP connection-mode session services over a dedicated WDP port.	6.1	O	WSP-CO-C-001
WTA-SM-C-003	Information to start up a WTA session is provided by the implementation.	6.1.1	M	
WTA-SM-C-004	The WTA user agent supports multiple simultaneous WTA sessions.	6.1	O	

Context Management

Item	Function	Reference	Status	Requirement
WTA-CM-C-001	WTA stores its context in a single scope.	6.2	M	
WTA-CM-C-002	Context is re-initialised when any of the conditions defined in WML occur.	6.2.1, [WML2]	M	WML-STRUCTMOD-C-001 AND WML-STRUCTMOD-C-002
WTA-CM-C-003	Context is re-initialised and terminated when indicated by the content author.	6.2.1	M	
WTA-CM-C-004	Context is re-initialised as a result of service activation upon a match between a WTA event and a global binding.	6.2.1	M	
WTA-CM-C-005	The WTA user-agent terminates the current context and indicates to the user in case an error is discovered in the content.	6.2.1	M	
WTA-CM-C-006	The WTA user-agent does not handle	6.2.1	M	

Item	Function	Reference	Status	Requirement
	additional events until the user acknowledges the error or other service is initiated by a global binding.			
WTA-CM-C-007	Support for redirecting new events to default MMI when the WTA user-agent is waiting for user acknowledgement due to content error.	6.2.1	O	
WTA-CM-C-008	The WTA user-agent handles calls, event parameters and repository when a newcontext attribute is processed.	6.2.2, [WML2]	M	WML-STRUCTMOD-C-001 AND WML-STRUCTMOD-C-002
WTA-CM-C-009	The WTA user-agent handles variables, navigational history state, content, calls, event parameters and repository when an endcontext function is processed.	6.2.3 [WTAI]	M	WTAI-CMS-C-002
WTA-CM-C-010	Support for WTA user-agent termination after call of an endcontext function.	6.2.3	O	
WTA-CM-C-011	Support for context protection.	[WTAI]	M	WTAI-CMS-C-003 AND WTAI-CMS-C-004

Event Parameter Management

Item	Function	Reference	Status	Requirement
WTA-EPM-C-001	The WTA context retains the actual parameters of the last bound event received.	6.3	M	
WTA-EPM-C-002	The WTA user-agent sets event parameter state.	6.3	M	

Call State Management

Item	Function	Reference	Status	Requirement
WTA-CSM-C-001	Support for management of call modes.	6.5	M	

A1.1.4 Content formats

Item	Function	Reference	Status	Requirement
WTA-FMT-C-001	Support for WML2 Content Format.	7.1, [WML2]	M	WML:MCF

A 1.1.5 Repository

Item	Function	Reference	Status	Requirement
WTA-REP-C-001	Support for channel loading.	8.1	M	WTA-REP-C-011 OR WTA-REP-C-012
WTA-REP-C-002	Support for channel unloading.	8.2	M	WTA-REP-C-011 OR WTA-REP-C-012
WTA-REP-C-003	Support for garbage collection.	8.3	M	
WTA-REP-C-004	Garbage collection removes all empty channels.	8.3	M	
WTA-REP-C-005	Garbage collection removes all stale channels.	8.3	M	
WTA-REP-C-006	Garbage collection removes all non-referenced resources.	8.3	M	
WTA-REP-C-007	Support for the Channel Content Format.	8.4.1	M	
WTA-REP-C-008	The channelid attribute of the channel is used to identify services.	8.4.1.2	M	
WTA-REP-C-009	The eventid attribute of the channel is used to establish a global binding for a WTA event.	8.4.1.2	M	
WTA-REP-C-010	Support for channel installation at any time.	8.4.3	M	
WTA-REP-C-011	Support for channel download using response to standard URL request.	8.4.3	O	
WTA-REP-C-012	Support for channel download using Service Indication.	8.4.3, [SI]	O	SI:MCF
WTA-REP-C-013	Support for accessing resources in the repository using URLs.	8.5	M	
WTA-REP-C-014	The repository is always checked for requested content before loading it from the network.	8.5	M	

A 1.1.6 Channel Loading

Item	Function	Reference	Status	Requirement
WTA-CHN-C-001	A channel is identified by the channelid attribute during loading and unloading.	8.1, 8.2, 8.4.1.2	M	
WTA-CHN-C-002	A resource indicated by a channel is loaded except when already in the repository and both the indicated and the stored resource are equally fresh.	8.1, 8.4.1.5	M	
WTA-CHN-C-003	The success URL, if present, is requested upon successful loading of all resources.	8.1, 8.4.1.2	M	
WTA-CHN-C-004	A channel is activated upon successful reception of response to the success URL.	8.1, 8.4.3.1	M	
WTA-CHN-C-005	The installation process is terminated if any errors occur.	8.4.3.1	M	
WTA-CHN-C-006	The failure URL, if present, is requested if loading fails.	8.1, 8.4.1.2, 8.4.3.1	M	
WTA-CHN-C-007	An error message is presented to the end-user in case of failure and no failure URL is available, or the request for the failure URL fails.	8.1, 8.4.3.1	M	
WTA-CHN-C-008	Channels and resources for which loading failed are discarded.	8.1	M	
WTA-CHN-C-009	The older channel, and its resources, is preserved in case loading of a newer version fails.	8.1	M	

A 1.1.7 Event Handling

Item	Function	Reference	Status	Requirement
WTA-EV-C-001	The WTA user agent responds to WTA events using local and global bindings.	9.2	M	
WTA-EV-C-002	A local binding is specified using the WML2 `wml:onevent` element.	9.2	M	WML:MCF
WTA-EV-C-003	Scope and shadowing semantics of local bindings are as defined for WML.	9.2	M	WML:MCF
WTA-EV-C-004	A global binding is specified using a WTA `channel` element.	9.2	M	
WTA-EV-C-005	A WTA event parameter is referenced using an indexing notation.	9.3	M	
WTA-EV-C-006	A reference to a non-existent WTA event parameter is resolved as defined in WML.	9.3	M	WML:MCF
WTA-EV-C-007	WTA event parameter references are legal anywhere variable references are legal in WML.	9.3	M	WML:MCF
WTA-EV-C-008	A WTA event parameter is read-only.	9.3	M	
WTA-EV-C-009	When navigating as a result of a local binding, the WTA user agent enforces the WML `wml:access` element of the target deck using the URI of the referring card or body.	9.5	M	WML:MCF
WTA-EV-C-010	When navigating as a result of a global binding, the WTA user agent enforces the WML `wml:access` element of the target deck using the URI of the channel specifying that global binding.	9.5	M	WML:MCF
WTA-EV-C-011	It is illegal for a single WTA user agent to have multiple concurrent WTA contexts.	9.6	M	

Item	Function	Reference	Status	Requirement
WTA-EV-C-012	The WTA user agent only processes WTA events when in a stable state.	9.6	M	
WTA-EV-C-013	The WTA user agent processes WTA events in the same order as they were received.	9.6	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.	9.6	M	
WTA-EV-C-015	The WTA user agent implements the event handling process.	9.6	M	

A 1.2 Server features

A1.2.1 Client Network Type Support

Item	Function	Reference	Status	Requirement
WTA-NETT-S-001	Client Network Type Support	-	M	WTA-GSM-S-001 OR WTA-ANSI136-S-001 OR WTA-PDC-S-001 OR WTA-IS95-S-001 OR WTA-ONETT-S-001
WTA-GSM-S-001	Supports GSM client	-	O	WTA-IF-S-002 AND WDP-GSM-S-000
WTA-ANSI136-S-001	Supports ANSI 136 client	-	O	WTA-IF-S-003 AND WDP-ANSI-S-000
WTA-PDC-S-001	Supports PDC client	-	O	WTA-IF-S-004 AND WDP-PDC-S-000
WTA-IS95-S-001	Supports IS-95 client	-	O	WTA-IF-S-005 AND WDP-CDMA-S-000
WTA-ONETT-S-001	Other Client Network Type Support	-	O	WDP-CT-S-002 OR WDP-FLEX-S-000 OR WDP-CT-S-006 OR WDP-CT-S-008 OR WDP-TETRA-S-000 OR WDP-DECT-S-000 OR WDP-FLEX-S-002 OR WDP-MOBITEX-S-000

A1.2.2 WAE features

Item	Function	Reference	Status	Requirement
WTA-WAE-S-001	WAE framework features	4, 5 and 6	M	WAE:MSF

A 1.2.3 Content Formats

Item	Function	Reference	Status	Requirement
WTA-FMT-S-001	Support for WBXML encoding of the WTA-WML Content Format.	7.2.4	O	WBXML:MSF
WTA-FMT-S-002	Support for WBXML encoding of the Channel Content Format.	8.4.1.7	M	WBXML:MSF

A 1.2.4 Security Features

Item	Function	Reference	Status	Requirement
WTA-SEC-S-001	Support for WTLS Class 2	5.5, [WTLS]	M	WTLS:MSF
WTA-SEC-S-002	A WTA session utilises a dedicated, secure port on the WAP Gateway.	5.2	M	WTLS:MSF AND (WDP-RP-S-001 OR WDP-RP-S-002)

A1.2.5 WTAI server features

Item	Function	Reference	Status	Requirement
WTA-IF-S-001	Supports server features of Public and Network Common WTA Libraries	[WTAI]	M	WTAI:MSF
WTA-IF-S-002	Support for server features of WTAI GSM Library	[WTAIGSM]	O	WTAIGSM:MSF
WTA-IF-S-003	Support for server features of WTAI ANSI 136 Library	[WTAIIS136]	O	WTAIIS136:MSF
WTA-IF-S-004	Support for server features of WTAI PDC Library	[WTAIPDC]	O	WTAIPDC:MSF
WTA-IF-S-005	Support for server features of WTAIIS95 Library	[WTAIIS95]	O	WTAIIS95:MSF

Appendix B. Examples of WTA Services (Informative)

B 1.1 Introduction

WTA services are created using WML2 [WML2] and WMLScript. From a WMLScript, telephony functions can be accessed through the Wireless Telephony Applications Interface (WTAI). WTAI also provides access to telephony functions from WML2 by using URIs [RFC2396]. URIs form a unifying naming model to identify features independently of the internal structure of the device and the mobile network. The WTA services reside on the WTA server. The client addresses WTA services by using URLs [RFC2396].

Examples of WTA services include:

- *Extended set of user options for handling incoming calls (incoming call selection):*

  The service is started when an incoming call is detected in the client. A menu with user options is presented to the user. Examples of options could be:

  - Accept call
  - Redirect to voice mail
  - Redirect to another subscriber
  - Send special message to caller

- *Voice mail:*

  The user is notified that she has new voice mails, and retrieves a list of them from the server. The list is presented on the client's display. When a certain voice mail has been selected, the server sets up a call to the client and the user listens to the selected voice mail.

- *Call subscriber from message list or log:*

  When a list of voice, fax or e-mails or any kind of call log is displayed the user has the option of calling the originator of a selected entry in the list or log.

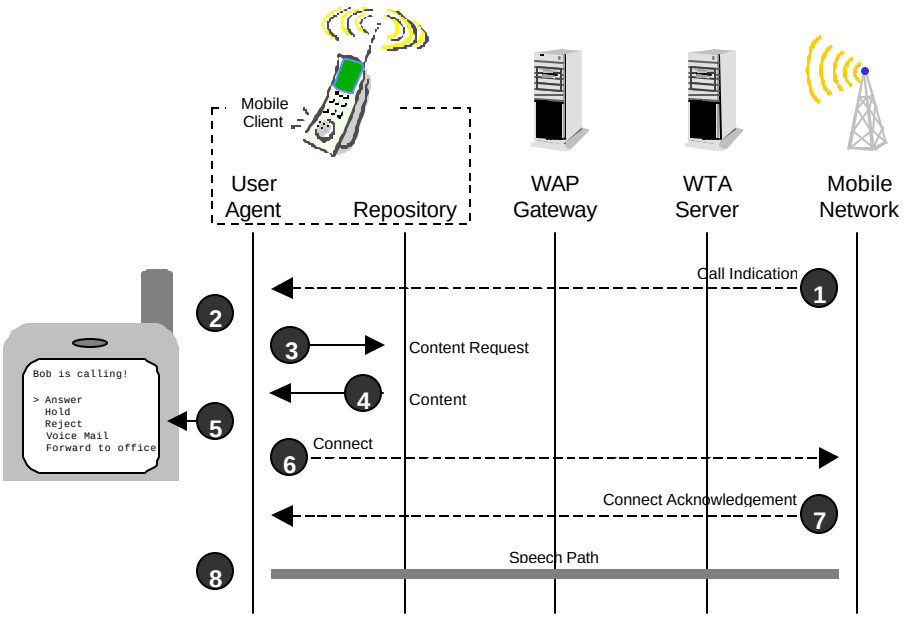
The following two sections show examples of how the WTA framework can be used for building telephony related services. Note that these examples only illustrate possible implementations, and are not the only way the architecture can be implemented. The WAP gateway and the WTA server could for example be implemented as one unit. The services could of course also be realised in other ways than those in the examples.

In the first example an "Incoming Call Selection" service is illustrated, and the second one shows how a voice mail service could be established.

B 1.2 Incoming Call Selection

The ”Incoming Call Selection” service is started when an incoming call is detected in the client, and a menu with various call-handling options is presented to the user.

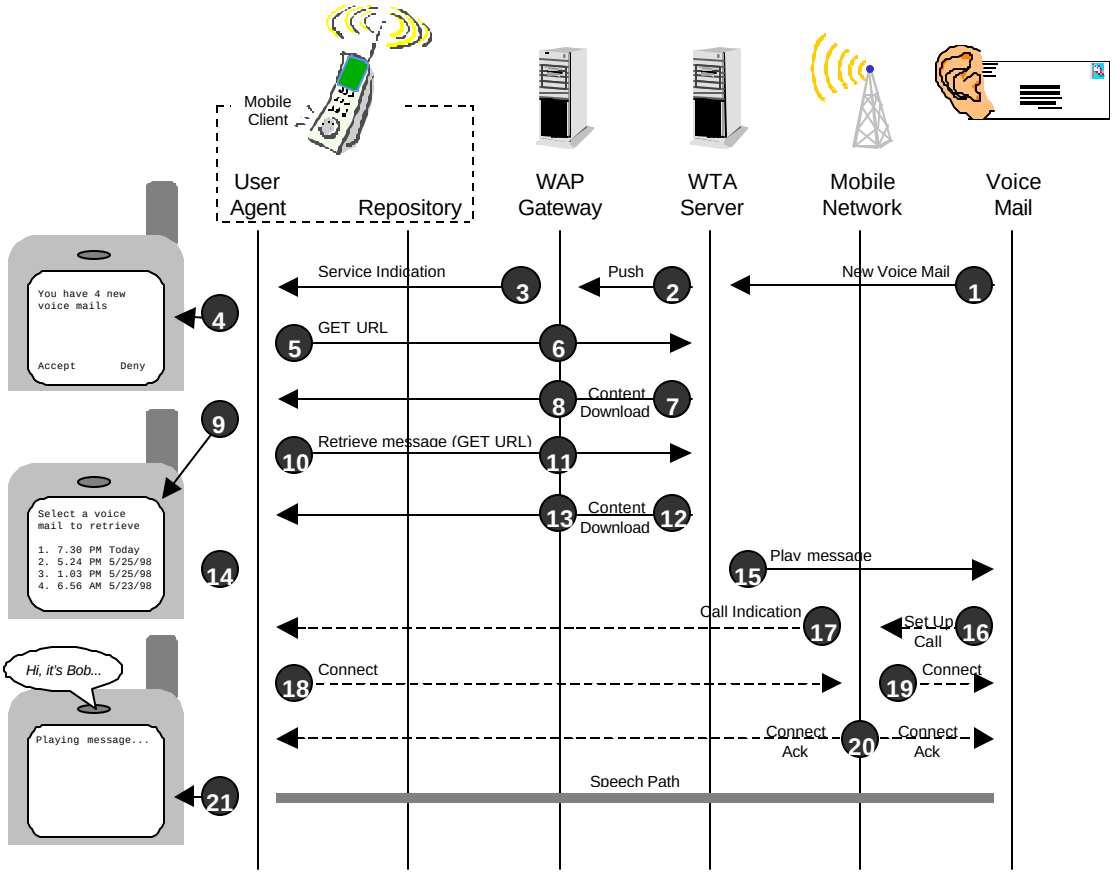
In the example, it is assumed that a valid channel and its associated content are stored in the repository. It is also assumed that the client is not engaged in any other WTA service (i.e. no temporary event bindings exist).



1. The mobile network receives an incoming call and sends a “Call indication” to the mobile subscriber.
2. In the client, the incoming call WTA event (wtaev-cc/ic) is generated. The repository is consulted in order to find a dedicated channel. The channel provides the URL to the “Incoming Call Selection” service stored in the repository.
3. The user-agent requests the content from the repository.
4. The repository returns the requested content.
5. The content is loaded into a clean context and starts executing. The service presents a list of options to the user, from which she can choose how to proceed with the call in progress. In this example she elects to answer the call. The WTAI function “WTAVoiceCall.accept” is invoked.
6. A “Connect” request is sent to the mobile network (the invoked WTAI function communicates with the mobile network).
7. A “Connect Acknowledgement” is generated in the mobile network. A result code indicating the outcome of the call is generated internally in the phone.
8. A speech path between the mobile network and the client is established.

B 1.3 Voice Mail

This example illustrates how a voice mail service could be established within the WTA framework. In the example the user is notified that she has received new voice mails and she chooses to listen to one of them.



1. The Voice Mail System notifies the WTA server that there are new voice mails. A list of them is also sent to the WTA server.
2. The WTA server creates new service content based on the list received from the voice mail system. The content is stored on the server, and its URL is included in a Service Indication that will be pushed to the client. The Service Indication's message could read: "You have 4 new voice mails".
3. The WAP gateway sends the Service Indication to the client using push.
4. The user is notified about the Service Indication by a message delivered with the Service Indication. The user chooses to accept the Service Indication.
5. A WSP "Get" request is sent to the WAP gateway (URL provided by the Service Indication).
6. The WAP gateway makes a WSP/HTTP conversion.
7. The WTA server returns the earlier created voice mail service.
8. The WAP gateway makes a HTTP/WSP conversion.

9. The voice mail service is now executing in the client. The user is presented with a list of voice mails originating from the Voice Mail System (a WML2 “Select List” created in step 2). The user selects a certain voice mail to listen to.

10. Another WSP “Get” request is sent to the WAP gateway. The requested deck identifies the selected voice mail.

11. The WAP gateway makes a WSP/HTTP conversion.

12. The WTA server returns the requested deck. The deck only contains one card with a single WML2 “`go href`” task. The URL is automatically called when the card is executed and it refers to a card in the earlier downloaded voice mail content which binds the incoming call event (wtaev-cc/ic) so that the subsequent call from the Voice Mail System will be answered automatically. Now, also the WTA server is informed about which voice mail the user has chosen to retrieve.

13. The WAP gateway makes a HTTP/WSP conversion.

14. The incoming call event (wtaev-cc/ic) is temporarily bound so that the call from the Voice Mail System will be answered automatically. In order to avoid that the voice mail service answers a call from someone else than the voice mail system, the calling party’s phone number (*callerId* parameter of the wtaev-cc/ic event) is preferably checked.

15. The WTA server instructs the Voice Mail System to play the selected voice mail.

16. The Voice Mail System instructs the mobile network to set up a call to the client.

17. The mobile network sets up a call to the client.

18. The client answers the call automatically due to content loaded in steps 12 to 14..

19. The mobile network informs the Voice Mail System that the client has accepted the call.

20. Acknowledgements are sent to the client and the voice mail system.

21. A speech path is established between the Voice Mail System and the client, and the message is played.

Appendix C. Using Events (Informative)

This annex describes implementation aspects that a WTA content developer should consider when writing WTA services that use WTA events. The two WTA models 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.

C 1.1 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 10**. 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.

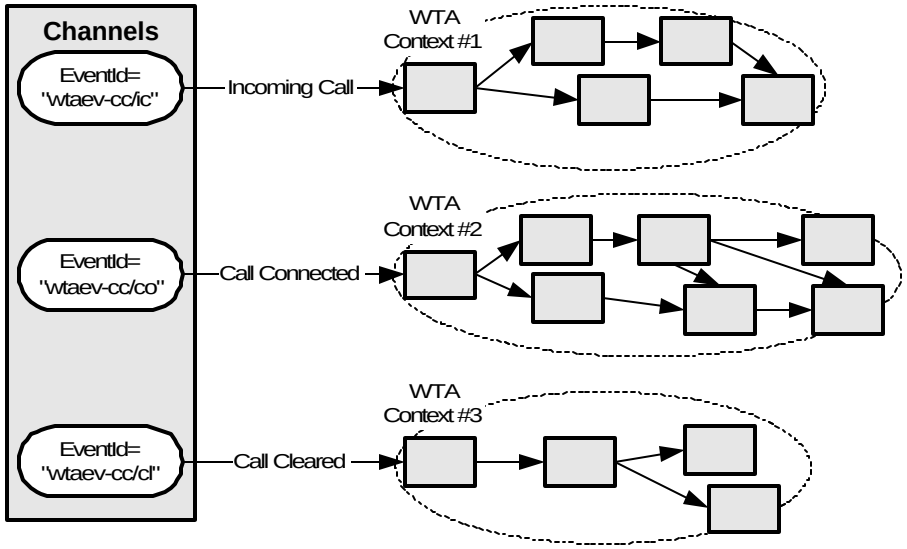


Figure 10 Application Model Using Only Global Bindings

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 9.6).

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.

C 1.2 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 11**. 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.

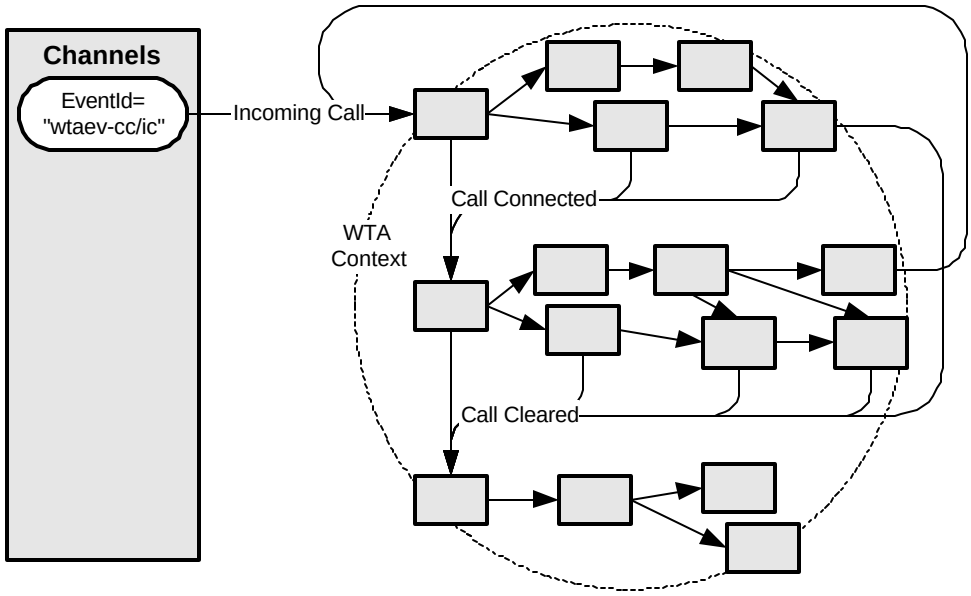


Figure 11 WTA Application Model Using Only One Global Binding

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.

Appendix D. Change History (Informative)

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