WAP WTA

Approved Version 16-Jul-1999

Wireless Application Protocol
Wireless Telephony Application Specification
Version 1.1

Disclaimer:

The document is subject to change without notice.

Contents

1. Scope

Wireless Application Protocol (WAP) is a result of continuous work to define an industry wide specification for developing applications that operate over wireless communication networks. The scope for the WAP Forum is to define a set of specifications to be used by service applications. The wireless market is growing very quickly and reaching new customers and 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*" [WAP].

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. Documents Status

This document is available online in the following formats:

PDF format at URL, http://www.wapforum.org/.

2.1 Copyright Notice

2.2 Errata

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

2.3 Comments

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

3. References

3.1 Normative references

[CACHE] "WAP Caching Model", WAP Forum, 11-Feb-1999. URL: http://www.wapforum.org/

[RFC1630] "Uniform Resource identifiers (URI)", T. Berners-Lee, et al., June 1994. URL: ftp://ds.internic.net/rfc/rfc1630.txt

[RFC1738] "Uniform Resource Locators (URL)", T. Berners-Lee, et al., December 1994. URL: ftp://ds.internic.net/rfc/rfc1738.txt

[RFC2068] "Hypertext Transfer Protocol - HTTP/1.1". R. Fielding,et al. January 1997. URL: ftp://ds.internic.net/rfc/rfc2068.txt

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997. URL: ftp://ds.internic.net/rfc/rfc2119.txt

[SI] "WAP Service Indication", WAP Forum, Draft 08-Jul-1999.

Editor's Note: The referenced Draft Specification for WAP Service Indication is written by the WAP Forum Push Drafting Committee and subject to change.

[WAE] "Wireless Application Environment Specification", WAP Forum, 30-April-1998. URL: http://www.wapforum.org/

[WAP] "Wireless Application Protocol Architecture Specification", WAP Forum, 30-April-1998. URL: http://www.wapforum.org/

[WBXML] "Wireless Binary XML", WAP Forum, 30-April-1998. URL: http://www.wapforum.org/

[WDP] "Wireless Datagram Protocol Specification", WAP Forum, 30-April-1998. URL: http://www.wapforum.org/

[WML] "Wireless Markup Language", WAP Forum, 30-April-1998. URL: http://www.wapforum.org/

[WMLScript] "WMLScript Language Specification", WAP Forum, 30-April-1998. URL: http://www.wapforum.org/

[WSP] "Wireless Session Protocol Specification", WAP Forum, 30-April-1998. URL: http://www.wapforum.org/

[WTAI] "Wireless Telephony Application Interface Specification", WAP Forum, 30-April-1998. 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

4. Definitions and Abbreviations

All non-trivial abbreviations and definitions used in this document are listed in the following sections. The definition section includes description of general concepts that may be fully defined in other documents. The purpose of this section is to advise the reader on the terminology used in the document.

4.1 Definitions

The following are terms and conventions used throughout this specification.

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 by [RFC2119].

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

Cachability – characteristics of locally stored content that can be used to determine if the content needs to be updated.

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.

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

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.

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.

Device OS – is the standard operating system in a device. Not specified by WAP.

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

Implicit Variables – are variables that are created by the user-agent as a result of a WTA event reception.

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.

Temporary Binding – A WTA event binding that is specified using WML content.

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

WTA Event – A notification, in the form of an abstract WTA event that conveys a change of state in the mobile network, e.g. an Incoming call.

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

WTA Server - an origin server, under the control by the mobile network operator that contains 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.

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

4.2 Abbreviations

For the purposes of this specification, the following abbreviations apply.

API	Application Programming Interface
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
SMTP	Simple Mail Transfer Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator [RFC1738]
WAE	Wireless Application Environment
WAP	Wireless Application Protocol [WAP]
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]

5. Introduction

5.1 Architectural Overview

WTA is an application framework for telephony services. The WTA user-agent essentially is a user-agent similar 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 describes one *possible* configuration of the WTA framework. However, this specification solely defines the components contained in the client.

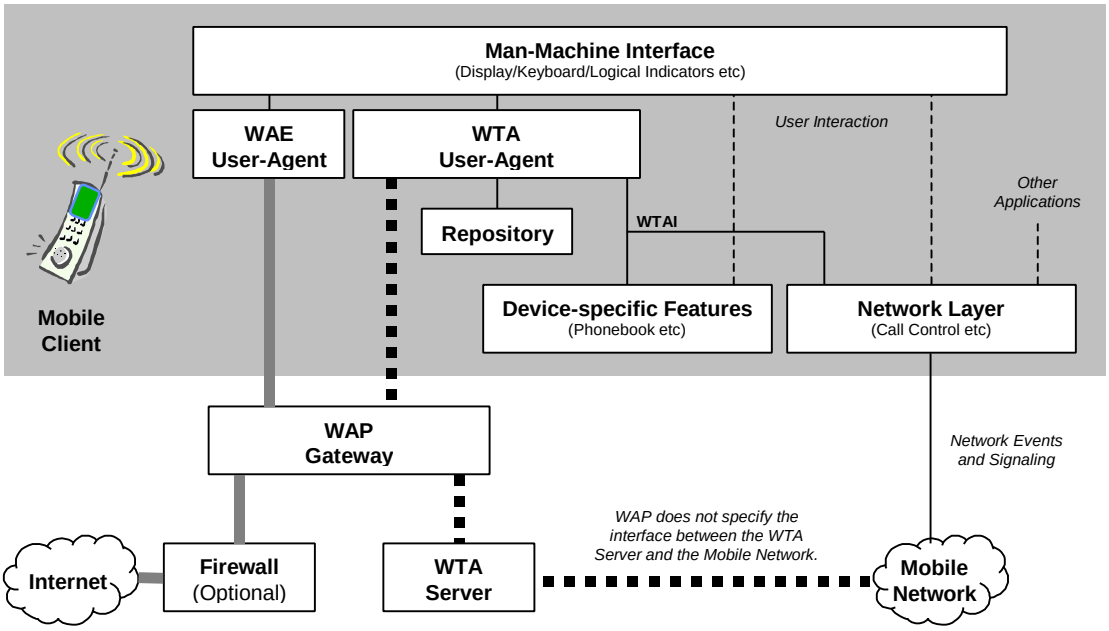


Figure 1, Overview of the WTA architecture

As mentioned above, the WTA framework relies on a dedicated WTA user-agent in the client, briefly described in section 5.1.1. The WTA server is not specified by WAP, but a brief overview is given in section 5.1.2.

5.1.1 The WTA User-Agent

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.

The WAE user-agent only retrieves its content via the WAP gateway. In addition, the WTA user-agent is also able to retrieve content from the repository. Further, 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).

5.1.2 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.

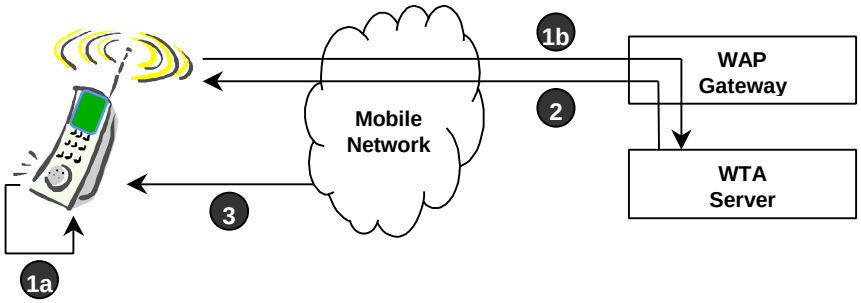
5.2 WTA Services

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. WML, WMLScript etc. The WTA user-agent 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).

5.2.1 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 2, Initiation of WTA services

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

5.3 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 9.1) 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?

5.3.1 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 available services (channels not related to WTA events) in the repository.
- A URL may be given to the user-agent (requested by the end-user, provided in content or delivered by a Service Indication [SI]). The content for this URL may be retrieved from the repository.

6. 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.

6.1 Security delegation

It is within every mobile operator's interest to provide an acceptable level of security in the network. How security is maintained in the operator's domain is not specified by WAP. The mobile telephony service provider can choose

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

6.2 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.

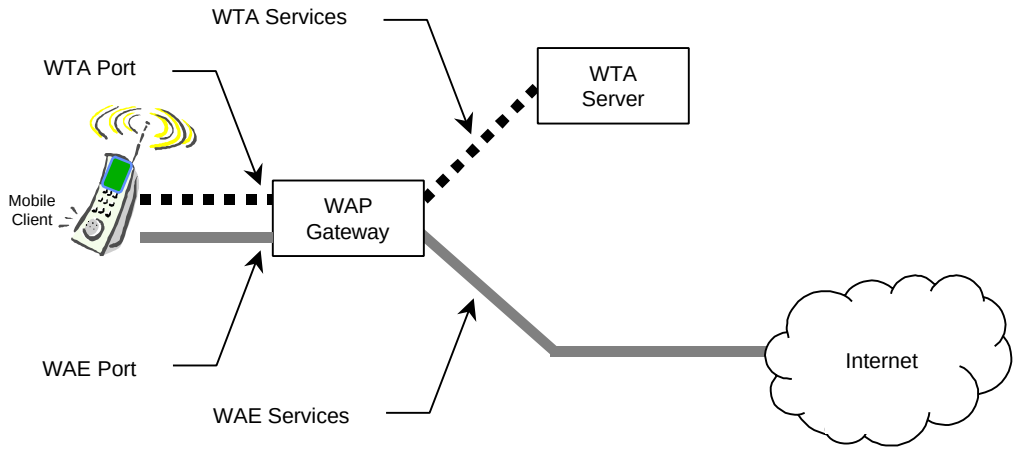


Figure 3, WDP Port Numbers and Access Control

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 shall be discarded.

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 calls WTAI functions.

The user gives permission for a specific WTAI function call by an executable. Support for blanket permission and single action permission is mandatory, but support for session permission is optional (see Table 1).

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

	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.
session permission	The user gives permission to the executable for the specified WTAI function during a specific run time session of an executable, and the executable subsequently uses the user's permission for the identified subsequent WTAI functions whilst the executable session is still running.	Typically such permission would be given at executable run time.	The session permission may be revoked by the user at any time. The user permission no longer applies once the executable run time session 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

6.4 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.

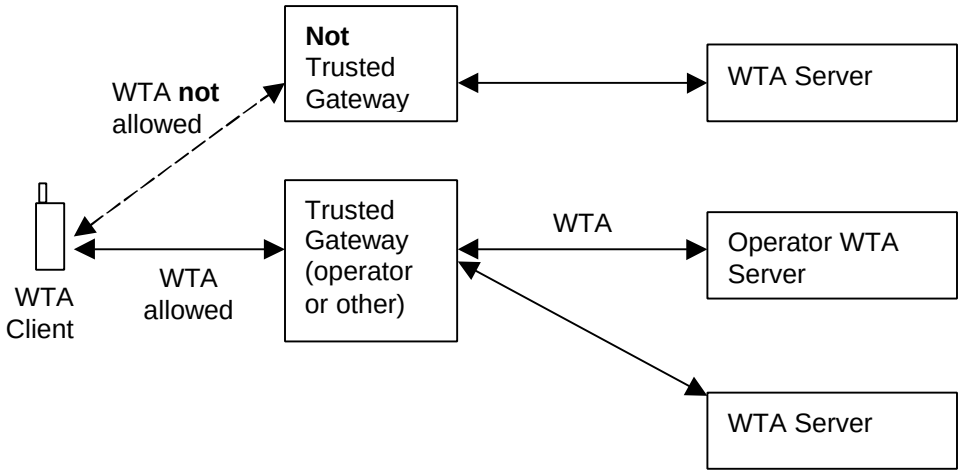


Figure 4, WTA Security Model

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.

7. 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 [WML]. 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 7.3).
- Call state management – the coupling between a call and the context of a WTA service (see section 7.5).
- 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.

7.1 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].

7.1.1 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.

7.2 User-agent Context

Like [WML], 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.

7.2.1 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 [WML] 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.

7.2.2 The Newcontext Attribute

The `newcontext` attribute of the `card` 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 [WML],
- Terminate the call if appropriate to the call mode (see section 7.5),
- Process ongoing calls according to the call mode (see section 7.5),
- Preserve the event parameters (see section7.3), and
- Preserve the contents of the repository (see section 7.3).

7.2.3 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 7.2.1). 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,
- Unload all content associated with current WTA context,
- Terminate the call if appropriate to the call mode,
- Process ongoing calls according to the call mode (see section 7.5),
- 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.

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 9.5.

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).

7.4 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.

7.5 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 (but not the opposite) 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. 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.

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 WML deck), and are stored along with their meta-data (e.g. content type and the HTTP 1.1 *entity-tag* [RFC2068]) 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 [RFC2068]), 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.

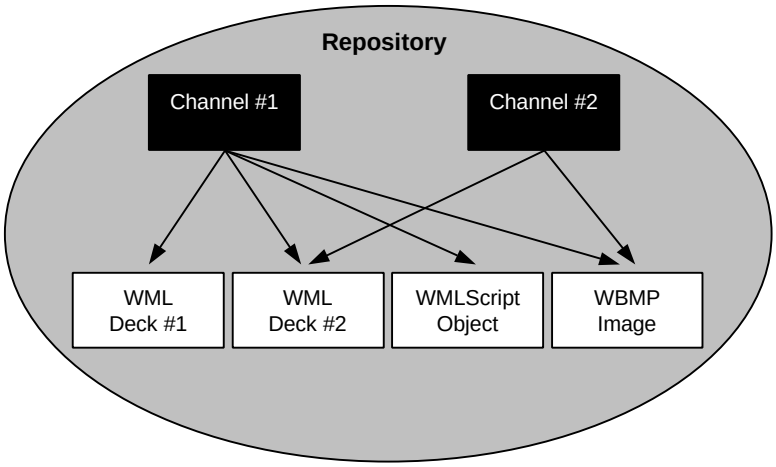


Figure 5, Repository

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). 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, 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 the channel is activated (becomes visible to the WTA user-agent). The channel is stored in the repository, replacing any previously stored channel with the same identity. 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, 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

Editor's note: These identifiers have not yet been registered with the IANA or ISO 9070 Registrar

8.4.1.1.1 SGML Public Identifier

```
-//WAPFORUM//DTD CHANNEL 1.0//EN
```

8.4.1.1.2 Channel Media Type

Textual form:

```
text/vnd.wap.channel
```

Tokenised form:

```
application/vnd.wap.channelc
```

8.4.1.2 The Channel Element

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

<!ELEMENT channel (title , abstract? , resource+ ) >
<!ATTLIST channel
      maxspace    %Number;    #REQUIRED
      base        %URI;       #IMPLIED
      EventId     CDATA       #REQUIRED
      %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 client may opt to not install the channel should the `maxspace` be exceeded.

`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 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 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.

`EventId=CDATA`

The `EventId` attribute specifies the identity of the channel. If the channel is associated with a WTA event the `EventId` must refer to the WTA event by using the predefined naming scheme (see Table 2 below). This identity is used during channel loading and unloading (see sections 8.1and 8.2).

Type	*EventId*	*Description*
Channels related to WTA events	wtaev-xx	(where xx equals the name of the WTA event, e.g. wtaev-cc/ic)
Other Channels	chan-xx	(where xx equals a unique ID for that channel, e.g. chan-MyChan)

Table 2, EventId naming scheme

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
  etagNMTOKEN     #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 [RFC2068]. 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 [RFC2068]. 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 [RFC2068] as the format of the "Etag" header value. The calculations for determining whether the content is still fresh are specified in [RFC2068] 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 [RFC2068] as the format of the "Content-MD5" header value.

If the `md5` attribute is not specified, 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 1.0//EN"
-->

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

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

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

<!ELEMENT channel (title , abstract? , resource+ ) >
<!ATTLIST channel
      maxspace    %Number;    #REQUIRED
      base        %URI;       #IMPLIED
      EventId     CDATA       #REQUIRED
      %ChannelEvent.attribs;
>

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

<!ELEMENT resource EMPTY>
<!ATTLIST resource
  href      %URI;       #REQUIRED
  lastmod   %Number;    #IMPLIED
  etagNMTOKEN     #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

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 1.0//EN"
      "channel.dtd">
<channel
      maxspace="2048"
      base="http://wap.operator.com/"
      EventId="wtaev-cc/ic"
      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.

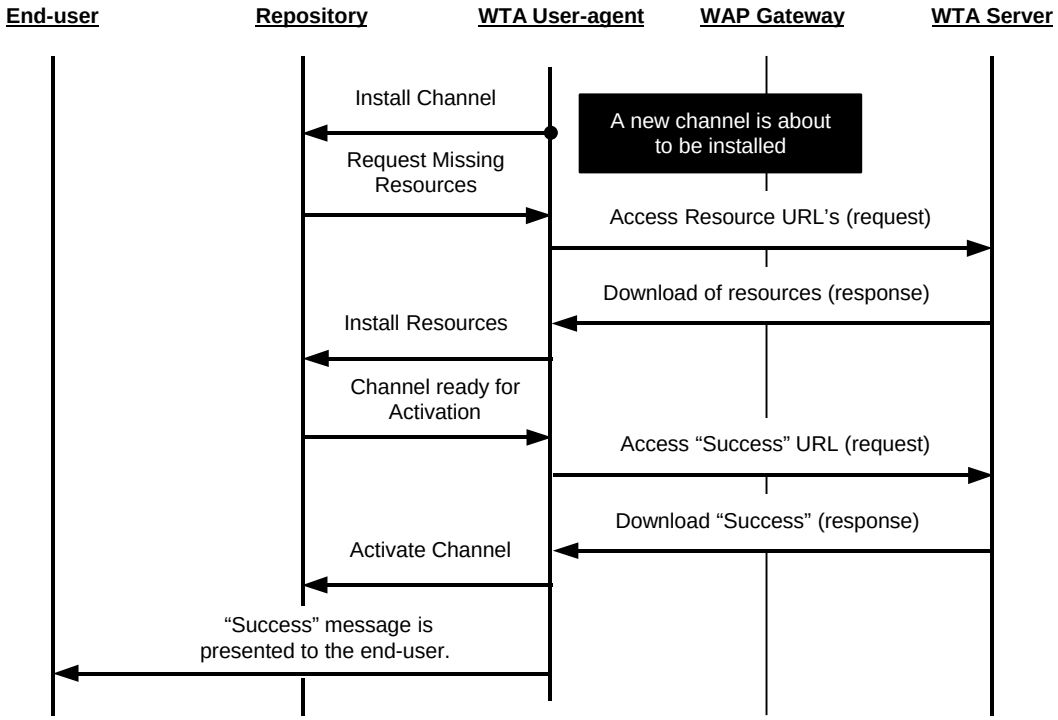


Figure 6, Successful Channel Installation

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

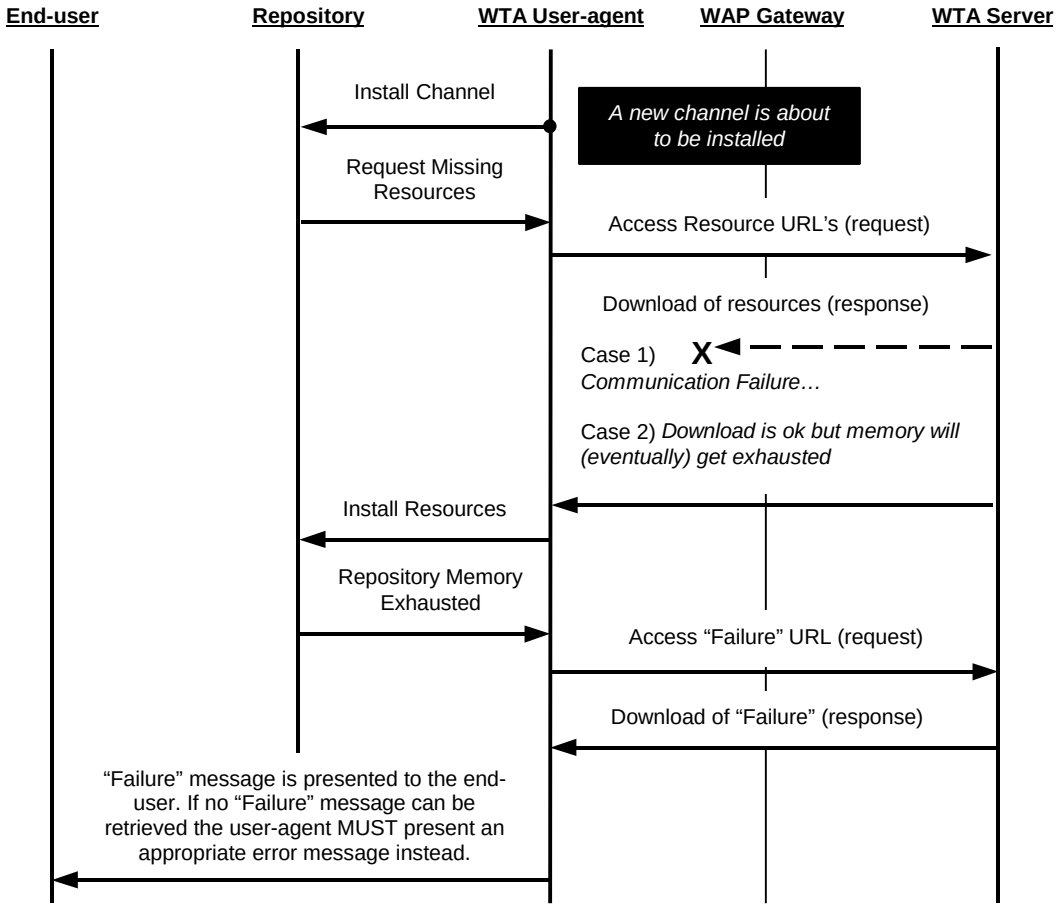


Figure 7, Failure during Channel Installation

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 [RFC2068]). 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

Network events are events that originate from the mobile network. Network events must be transformed from their abstract form into a predefined format called WTA events, suitable to use with the WTA user-agent.

The WTA user-agent can be programmed to take certain actions when receiving a specific WTA event. The actions to take can be controlled from within an executing WTA service or outside the context of a WTA service. Network events are real time events, which implies that a WTA event must be acted upon in a timely manner. Any content involved in handling a WTA event should therefor be available in the client as to not incur any additional delays due to network access during content download. Content to be executed in response to a WTA event should therefor be stored in the WTA repository, see section 8.

A WTA event can be associated with any content in the repository. The `EventId` attribute of a channel uses a predefined naming scheme to link the channel to a specific WTA event (see section 8.4.1.2).

9.2 Event Handling Process

A WTA user-agent can respond to a WTA event using two different methods:

1. **Global Bindings** - The repository may institute a binding between a WTA event and a service. When the event occurs, the corresponding service (already residing in the repository) is initiated.
2. **Temporary Bindings** - Overrides any other event bindings and makes it possible to execute a service using an existing WTA context.

If the WTA user-agent has no context executing, there can not exist any temporary bindings and thus it is sufficient to check the global bindings for a match upon reception of a WTA event.

If a service is already executing in the WTA user-agent upon detection of a WTA event and no temporary bindings exist, it must be determined whether the WTA context is protected from interuption by the global bindings [WTAI] or not in order to resolve if the detected event should be matched against the global bindings.

If the process described above reaches a point where the global bindings are consulted, but no matching binding exists, a fallback procedure to the standard MMI should ensure that the original network event is handled properly (in the same way as if the WTA user-agent were disabled). This is also the case when the WTA context is protected and no temporary bindings exist.

The following order of precedence MUST be applied when processing a WTA event, see flowchart below:

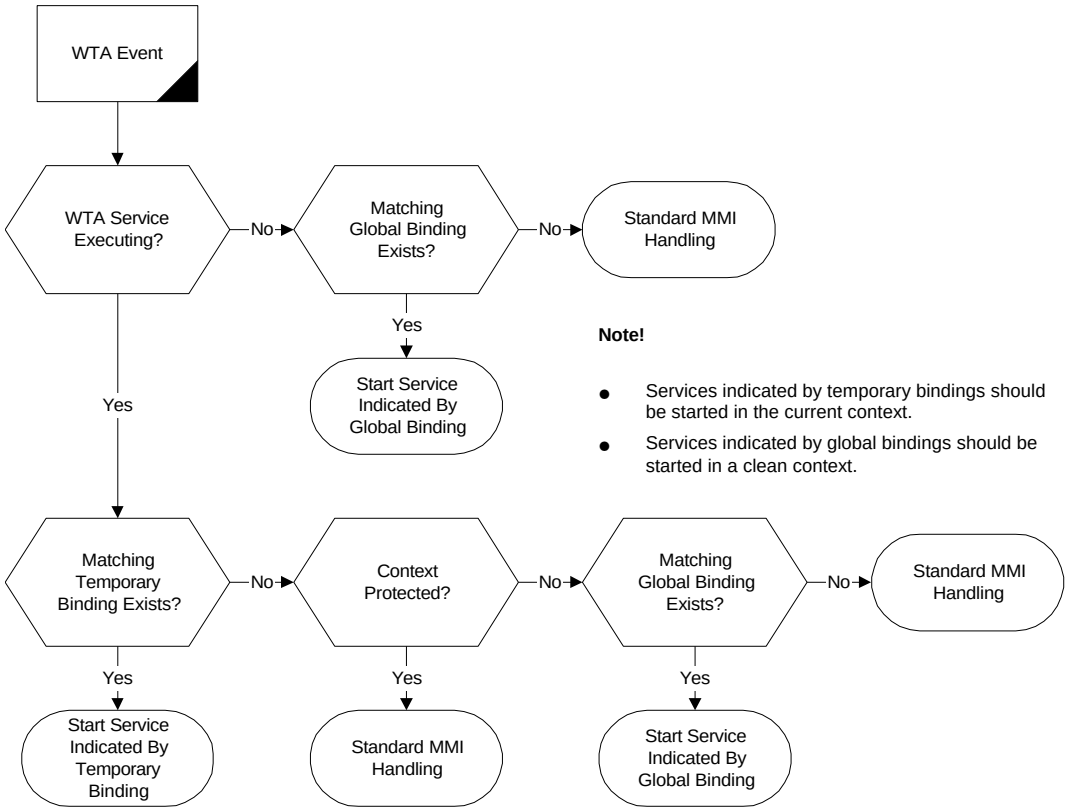


Figure 8, Event Handling Process Flowchart

9.2.1 Interruption of WTA context

If a WTA service that has no temporary bindings is in progress, and its context is not protected [WTAI], the rules according to Table 3 apply when a WTA event is detected. What happens if there is a temporary binding to the detected WTA event is described in section 9.3. In a situation when the WTA service in progress has no temporary binding, and its context is protected, the rules described in section 9.6.1 apply.

WTA USER-AGENT STATE	WTA USER-AGENT ACTION
The WTA user-agent is in a stable state, a card is being displayed	Current WTA context is interrupted. Then process the WTA event.
Execution of WTAI function is in progress.	Wait until execution of WTAI function has been completed. Then the current WTA context is interrupted and the WTA event is processed.
Execution of a WMLScript is in progress.	Interrupt the execution of the WMLScript. Then the current context is interrupted and the WTA event is processed.
A local navigation (to a card in the same deck, to the local cache or to the repository) is in progress.	Wait until navigation has been completed. Then the current context is interrupted and the WTA event is processed.
One or more WSP method requests (GET, PUT etc.) are in progress.	Abort all WSP method requests in progress with the WSP S-Method-Abort service primitive. Then process the WTA event.

Table 3, Interrupting the WTA Context

Note: In the table above ”current context is interrupted” means that the WTA context that is being processed MUST be terminated according to the rules in section 7.2.1, if there is an associated global binding to the detected event. Should there not be any global binding, the fallback procedure to the standard MMI is used and the WTA service in progress MAY be terminated. If the current context is not terminated the entire WTA context and all associated state of the interrupted service must be preserved so that it can be resumed after the interrupting event has been handled.

Implementation Note: It depends on device specific capabilities (such as memory size) whether an interrupted WTA context may be preserved and resumed after the interrupting event has been handled.

This specification does not mention how the processing of WTA events should co-exist with or interrupt other activities, e.g. standard WML–browsing or local MMI functions, in the client. This is implementation dependent. However, it is strongly recommended that a WTA event be handled without any significant delays.

9.3 Temporary Event Bindings

An executing 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. The WML `onevent` element is able to bind a WTA event [WTAI] and associate it with a task. E.g. `<onevent type="cc/ic"><go href=..>`. See details in [WML].

The following steps MUST be carried out in order to process a WTA event with regards to any temporary bindings:

1. The detected WTA event MUST be compared with the event names defined by the temporary event bindings in the WTA user-agent context.
2. In case of a match with any temporary bound event in the WTA user-agent context, the WTA event parameters (if any) are implicitly copied to the event variables (see section 9.5).
3. The task specified in the WML `onevent` construct is executed.

9.4 Global Event Bindings

9.4.1 Repository

The repository is a persistent storage space containing a set of channels. Specifically, a channel contains a list of URLs that identifies the content that should be used by a WTA service in response to a specific WTA event, if the identity of the channel associates to that specific WTA event. Content indicated by URLs listed in a channel is also stored in the repository (see section 8).

9.4.2 Responding to WTA Events

The following steps MUST be carried out in order to process a WTA event:

1. Check the repository for a dedicated channel for the specified WTA event.
2. In case of a match the WTA user-agent starts to process the content indicated by the URL specified by the `href` attribute in the first listed resource element in the channel. Note that this content is loaded into a clean context, i.e. no previous variables or state is associated with the user-agent context. Parameters received with the WTA event are implicitly copied to the WTA context using pre-defined variable names (see section 9.5).

9.5 Event Parameter Reference

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.

WTA allows authors to reference event parameters left to right using an index notation defined by the following BNF-like description used by [WML] to define variable substitution syntax:

```
eventvar =      (   "$" eventvarindex
                  | "$" "(" eventvarindex [conv] ")"
                )
eventvarindex = digit *[ digit ]
```

The `conv` and `digit` symbols are defined in [WML]. The same variable substitution rules defined in [WML] apply to the values referenced by the event parameter indices. Authors may set event parameters. However, such state may be overwritten by the next WTA event (see section 7.3). The following example illustrates how event parameters may be referenced:

```
<wml>
    ...
   <card>
      <onevent type="cc/ca">
         <go href="#cont">
            <setvar name="callerId" value="$2"/>
         </go>
      </onevent><p>
      Calling - please wait.
   </p>
   </card>
   <card id="cont"><p>
      Connected to $(callerId)
   </p></card>
</wml>
```

9.6 Fallback Handling

9.6.1 Fallback Handling for unbound WTA events

Handling of WTA events MUST fall back to the standard MMI when there is no current WTA user-agent context and there is no associated channel stored in the repository, i.e. there is no global event binding.

Handling of WTA events MUST also fall back to the standard MMI when there is a current WTA user-agent context but

1. there is no associated temporary event binding and the context is protected
2. there is no associated temporary or global event binding

In case 1 the current WTA user-agent context MUST remain unaffected. In case 2 the WTA user-agent context is interrupted as described in section 9.2.1.

Implementation Note: WAP does not place any requirements on how a client device handles network events outside the WTA framework.

9.6.2 Fallback Handling for Failed Service

If loading the content bound to a WTA event fails for any reason, the current WTA context (if any) MUST be terminated and event handling MUST then fall back to the standard MMI.

Note: In order to avoid the situation described above, it is recommended that WTA services are designed so that the content indicated by temporary bindings either resides in the same WML deck or in the repository.

10. Examples of WTA Services

10.1 Introduction

WTA services are created using WML 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 WML by using URIs [RFC1630]. 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 [RFC1738].

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.

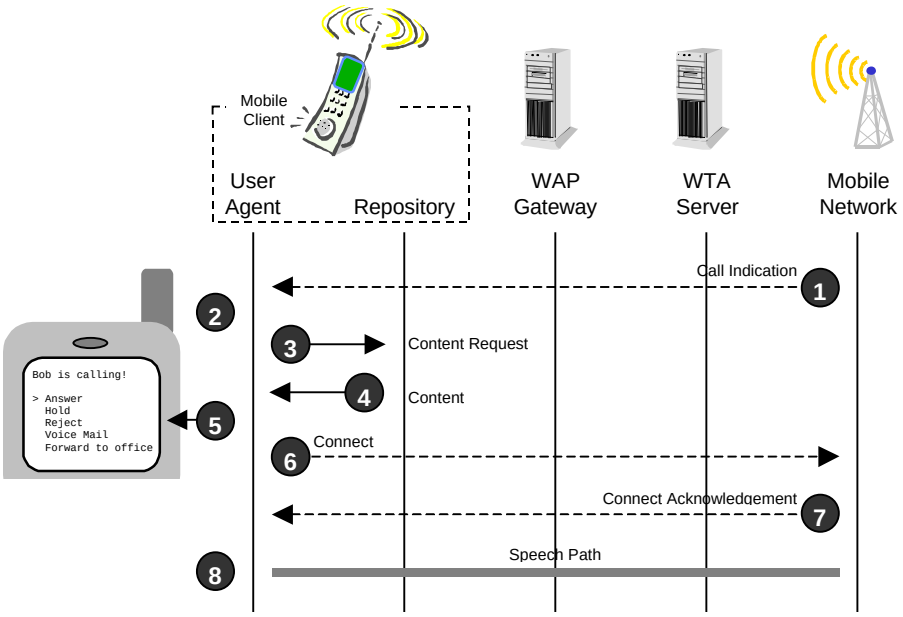
The following two sections show **examples** of how the WTA framework can be used for building telephony related services. Note that these examples only illustrate possible implementations, and are not the only way the architecture can be implemented. The WAP gateway and the WTA server could for example 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.

10.2 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.

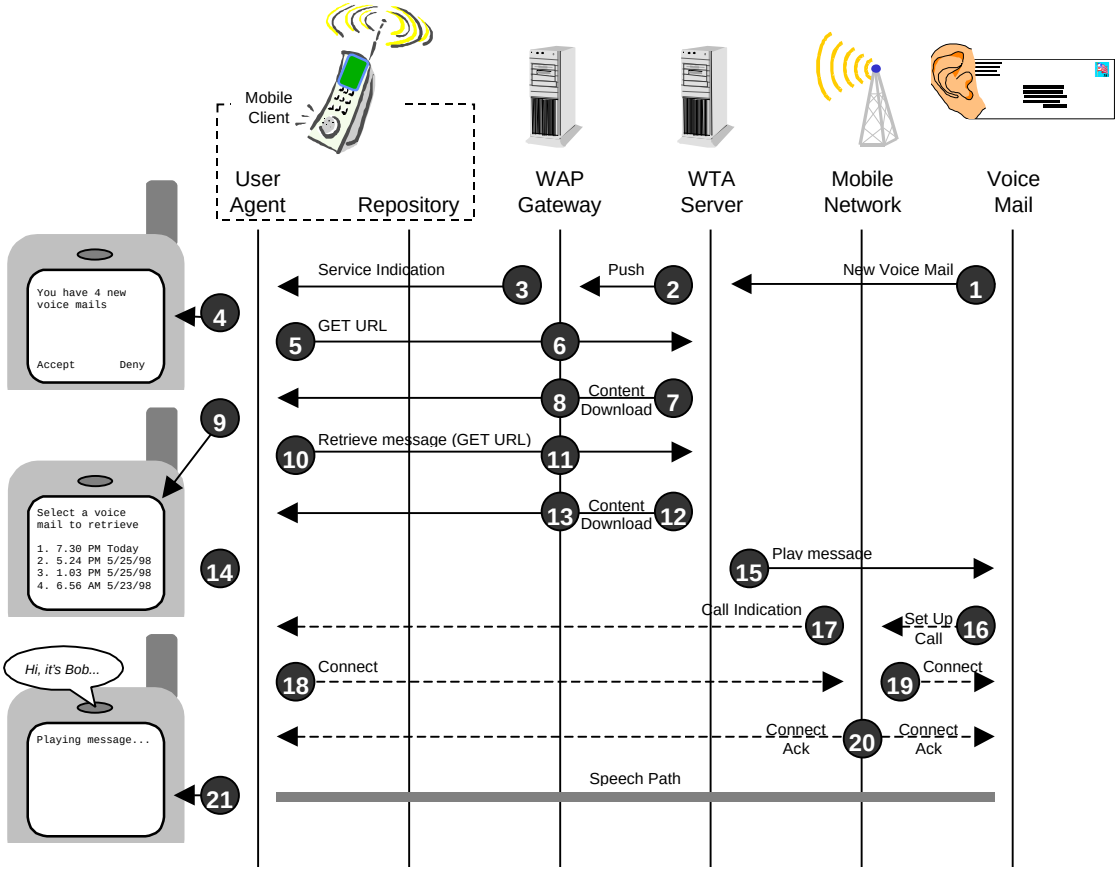
In the example, it is assumed that a valid channel and its associated content are stored in the repository. It is also assumed that the client is not engaged in any other WTA service (i.e. no temporary event bindings exist).



1. The mobile network receives an incoming call and sends a “Call indication” to the mobile subscriber.
2. In the client, the incoming call event (cc/ic) is generated. The repository is consulted in order to find a dedicated channel (wtaev-cc/ic). 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 “vc/ac” 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.

10.3 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 should be pushed to the client using the Service Indication. The Service Indication is sent to the WAP gateway using push. 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 WML "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 WML "`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 (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 (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 (*id*) 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.
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.

Annex 1 - Relation to other WAP Forum specifications

This annex is informative only.

A 1.1 Security

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 is ensured using WTLS class 2 with mandatory use of server certificate authentication. The use of a WTLS session is mandatory for WTA services (non-public).

In the WTLS security model, WTLS certificates are stored in the WAP client, the WTLS session based on these certificates is also used for WTA.

The WTLS security model extends from client to the gateway[2]. The service provider (or an entity delegated by the service provider) shall supervise the gateway in order to ensure that a "secure" trust relationship be maintained between the gateway and the WTA server. The WTLS security model uses certificates stored in the WAP client to enforce that only trusted gateways can be used in order to access telephony resources in the device. Using a transport layer security model, any WTA service can be delivered by the service provider (or an entity delegated by the 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. This is achieved by the identity, authenticated by its certificate, of the specified gateways being an identity specifically allowed for such connections.

Since the WTLS layer (WTLS Session) secures the connection between the client and the WAP gateway, the content (identified by WTLS authentication) can be executed directly by the WTA user-agent. There is no need for any additional processing of the content in order to determine its validity and all operations can be trusted.

A 1.2 Push and Service Indication

The WTA framework assumes that there will exist a push framework and a Service Indication content type. The push framework defines the mechanism that is used by a server initiating content delivery to a WTA user-agent. The Service Indication content type is the actual format in which a notification of a service to be loaded, i.e. content to be retrieved, is transferred and presented to the WTA user-agent.

Editor's note: It is within the WAP Forum Push Drafting Committee that a push framework and a Service Indication content type are being specified. The Service Indication content type serves all kinds of user-agents, including WTA. The WTA specification, in this version, does not intend to suggest any additional features for push or Service Indication to those specified by the WAP Forum Push Drafting Committee.

The WTA framework uses the Service Indication content format as described in [SI].

[2] Using WTLS an end to end solution could be achieved between the WAP client and an entity that acts as the termination point for the WAP protocol stack on the server side.

A 1.3 Persistent Storage

The WAP Forum Arch Persistence Drafting Committee is defining a standard, client based persistent storage system to be used by WAP. The WTA framework includes the Repository, specified in section 8, which is a module providing persistent storage capabilities for content that should be accessible from the WTA user agent. In order to align with the persistent storage work, the WTA group will collaborate with the Persistence Drafting Committee and see to that the WTA specification does not assume or specify mechanisms that will conflict with the intentions of the Persistence Drafting Committee.

A 1.4 Provisioning

How the certificates are provisioned is not a WTA specific problem and so is outside the scope of this specification.

Editor's note: At the time of writing, the provisioning of certificates onto WAP clients is not specified.