Specification Information Note

WAP-191_105-WML-20020212-a

Version 12-Feb-2002

for

Wireless Application Protocol
WAP-191-WML-20000219-a
Wireless Markup Language
Version 1.3, 19-February-2000

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

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

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

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

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

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

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

Contents

1. Scope

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

- WAP-191-WML-20000219-a

It includes changes from the following change requests:

- CR-IBM-WML1.3-20020131
- CR-IBM-WML1.3-20020131-A

with modification to specification reference.

2. Notation

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

3. Addition of SCR for tabindex

3.1 Change Classification

Class 2 – Bug Fixes

3.2 Change Summary

Problem Report (PR) 2413 pointed out that tabindex is optional for WML user agents and yet does not have a separate SCR to indicate its optionality (Section 11.6.1).

This change adds the necessary SCR to clearly illustrate its optionality and help suppliers declare the devices support for tabindex when certifying their devices.

3.3 Change Description

Section 15.1.5

WML-C-76	tabindex support	11.6.1	O	

4. Clarification of the go element for the Post method.

4.1 Change Classification

Class 2 – Bug Fixes

4.2 Change Summary

Problem report (PR) 2414 suggested the text associated with enctype and the use of the post method with multipart/form-data was insufficiently clear with respect to their optionality.

This change clarifies the situation.

Section 9.5.1 is insufficiently clear.

4.3 Change Description

9.5.1 The Go Element

```
<!ENTITY % cache-control “(no-cache)” >
<!ELEMENT go (postfield | setvar)*>
<!ATTLIST go
 href                %HREF;          #REQUIRED
 sendreferer         %boolean;       "false"
 method              post|get)       "get"
 enctype             %ContentType;
           "application/x-www-form-urlencoded"
 cache-control       %cache-control; #IMPLIED
 accept-charset      CDATA           #IMPLIED

%coreattrs;
>
```

The go element declares a go task, indicating navigation to a URI. If the URI names a WML card or deck, it is displayed. A go executes a "push" operation on the history stack (see section 9.2). The UA must ignore all postfield elements of a go element if the target of the go element is a card contained within the current deck and if the cache-control is not specified as “no-cache”. Refer to section 12.5.1 for more information on the semantics of go.

Attributes

href=*HREF*

The href attribute specifies the destination URI, e.g., the URI of the card to display.

sendreferer=*boolean*

If this attribute is true, the user agent must specify, for the server's benefit, the URI of the deck containing this task (i.e., the referring deck). This allows a server to perform a form of access control on URIs, based on which decks are linking to them. The URI must be the smallest relative URI possible if it can be relative at all. For example, if sendreferer=true, an HTTP based user agent shall indicate the URI of the current deck in the HTTP "Referer" request header [RFC2616].

method=*(post|get)*

This attribute specifies the HTTP submission method. Currently, the values of get and post are accepted and cause the user agent to perform an HTTP GET or POST respectively.

cache-control=*no-cache*

If the “cache-control” attribute is present, and the value is set to “no-cache”, the client MUST reload the URL from the origin server. This attribute represents the HTTP “cache-control” header, when this attribute is present, the HTTP “cache-control” header MUST be added to the request with the same value as specified in the attribute.

enctype=*ContentType*

This attribute specifies the content type used to submit the parameter to the server (when the value of method is post). The default value for this attribute is `application/x-www-form-urlencoded`. Currently, only `application/x-www-form-urlencoded` or `multipart/form-data` can be specified. When the field values in a submitted form may contain characters not in the US-ASCII character set, it is recommended that the post method and optional `multipart/form-data` [RFC2388] are used. User agents must explicitly specify content type for each part. If a part corresponds to a `postfield` element, its content type should be `text/plain`. The charset parameter is required when the content contains characters not in the US-ASCII character set.

`accept-charset=`*`cdata`*

This attribute specifies the list of character encodings for data that the origin server accepts when processing input. When the `go` task is executed and `method="post"` has been specified, the user agent should encode the field names and values of all associated `postfield` elements using any one of the specified character sets.

The value of this attribute is a comma- or space-separated list of character encoding names (`charset`) as specified in [RFC2045] and [RFC2616]. The IANA Character Set registry defines the public registry for charset values.

If the `accept-charset` attribute is not specified or is the reserved string `unknown`, user agents should use the character encoding that was used to transmit the WML deck that contains the go element.

Attributes defined elsewhere

- id (see section 8.9)
- class (see section 8.9)

The `go` element may contain one or more `postfield` elements. These elements specify information to be submitted to the origin server during the request. The submission of field data is performed in the following manner:

1. The field name/value pairs are identified and all variables are substituted.
2. The user agent should transcode the field names and values to the correct character set, as specified explicitly by the `accept-charset` or implicitly by the document encoding.
3. If the `href` attribute value is an HTTP URI, the request is performed according to the `method` and `enctype` attribute's value:

Method	Enctype	Process
`get`	`application/x-www-form-urlencoded`	The field names and values are escaped using URI-escaping and assembled into an `application/x-www-form-urlencoded` content type. The submission data is appended to the query component of the URI. The result must be a valid query component with the original query part and the postfields combined. An HTTP GET operation is performed on the resulting URL.
	multipart/form-data	Error.
`post`	`application/x-www-form-urlencoded`	The field names and values are escaped using URI-escaping and assembled into an `application/x-www-form-urlencoded` content type. The submission data is sent as the body of the HTTP POST request. The `Content-Type` header must include the charset parameter to indicate the character encoding.
	`multipart/form-data`	The field names and values are encoded as a `multipart/form-data` content type as defined in [RFC2388]. The `Content-Type` header must include the charset parameter to indicate the character encoding when the part contains characters not in the US-ASCII character set. The submission data is sent as the body of the HTTP POST request.

When enctype attribute's value is `application/x-www-form-urlencoded`, the field names and values must be encoded as follows:

1. The field names and values are escaped using URI-escaping, and listed in the order in which the postfields are presented.
2. The name is separated from the value by `=' and name/value pairs are separated from each other by `&'. See [RFC2396] for more information on the URI syntax and its escape sequence.

It is recommended that user agents submit data with the optional~~an~~ `application/vnd.wap.multipart.form-data` content type when enctype attribute has a value of `multipart/form-data`. Some user agents may only support data submission using the default~~as~~ `application/x-www-form-urlencoded` content type. Such user agents may ignore `enctype` attribute. Thus, it is recommended that an origin server expect either form of submission (i.e., `multipart/form-data` or `application/x-www-form-urlencoded`) when the `enctype` value is `multipart/form-data`. However, the origin server may assume that it will only receive an `application/x-www-form-urlencoded` submission when the `enctype` value is `application/x-www-form-urlencoded`.

For example, the following go element would cause an HTTP GET request to the URL

```
"/foo?x=1":
  <go href="/foo">
    <postfield name="x" value="1"/>
</go>
```

The following example will cause an HTTP POST to the URL `"/bar"` with a message entity containing `"w=12&y=test"`:

```
  <go href="/bar" method="post">
    <postfield name="w" value="12"/>
    <postfield name="y" value="test"/>
  </go>
```

Conformance Rules:

WML-C-29	go	M
WML-C-77	Support for `enctype`	O
WML-C-78	Support for `application/vnd.wap.multipart.form-data`	O

15.1.5 Elements

Item	Element	Reference	Status	Requirement
WML-C-77	Support for `enctype`	9.5.1	O	
WML-C-78	Support for `application/vnd.wap.multipart.form-data`	9.5.1	O	WML-C-77