Specification Information Note

WAP-239_101-WCSS-20020415-a

Version 15-April-2002

for

Wireless Application Protocol
WAP-239-WCSS-20011026-a
WAP CSS Specification
Version 26-Oct-2001

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

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

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

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

Contents

3.1 CHANGE CLASSIFICATION 5
3.2 CHANGE SUMMARY 5
3.3 CHANGE DESCRIPTION 6

1. Scope

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

- WAP-239-WCSS-20011026-a

It includes changes from the following change requests:

- CR-WAP-239-WCSS-MOT-20020107-ACCESSKEY-5
- CR-WAP-239-WCSS-ERICSSON-20020211-ACCESSKEY-7
- CR-WAP-239-WCSS-ERICSSON-20020311-WAPINPUTFORMAT-1
- CR-NOKIA-WCSS-20020403-1
- CR-NOKIA-WCSS-20020403-2

2. Notation

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

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

3. Clarifications and Bugfixes for Access Key and Input Format

3.1 Change Classification

Class 2 – Bug Fixes

3.2 Change Summary

This SCD is derived from five different CRs.

3.2.1 Add Support for the CSS <string> Data Type

The property values of the font-family (section 12.1) and –wap-input-format (section 19.2) properties may be of the CSS2 <string> Data Type (as defined in CSS2 section 4.3.10). Given this, WCSS's Data Types table (section 6.2) must be expanded so that it explictly includes the <string> Data Type.

3.2.2 Fix Typographical Error in Font Family Property

The font-family property (section 12.1) refers to a CSS <font-family> Data Type. This is a typographical error – there is no <font-family> Data Type - it should be the <family-name>.

3.2.3 Clarification of Multiple Assignment for Accesskey

Section 18.1.2 of the WCSS spec describes fallback and multiple selections for access keys. When these two selection algorithms are combined, e.g.

```
a { -wap-accesskey: snd *, #;}
```

it is unclear whether the user agents should make a best-effort or all-or-nothing attempt when assigning the space separated access keys.

This was discussed within the WAE DC in Berlin, with the consensus being:

1. If the user specifies only space separated access keys, the user agent must attempt to assign the available keys (best-effort).
2. If the user specifies space separated access keys followed by a comma and one or more access keys, the user agent must first make a best effort attempt to assign the space separated access keys. Only if none of the space separated access keys is available should it attempt to assign the next comma separated access key(s).

It was also unclear as to the order in which comma separated access keys should be assigned, for example left-to-right or right-to-left. Although the parsing direction is not mandated, some text has been added to help clarify this point. The new text is consistent with that of the font set in the CSS2 specification, which shares the concept of a comma separated, prioritized list.

3.2.4 Improving the Access Key Syntax

In order to be more explicit in the syntax for lists of key combinations a KeyCombinationList is added to the BNF.

The Char rule has been changed from accepting all Unicode Characters in [XML] to include characters that are allowed as start characters in a CSS2 expr (See CSS2 Appendix D) and that in the same time can be present as a single character without generating CSS2 parsing errors.

The VendorKey is removed since the dot (.) would generate a parsing error.

According to the [CSS2] syntax, the “*” and “#”are not allowed characters in an identifier. Therefor they must be coded to their unicode escape sequence.

Another minor editorial change that is to add double colons in front of the equal sign in the BNF for SpecialKey and ModifierKey.

3.2.5 Backslash in Inputformat Must be Escaped

In CSS strings the backslash (\) is regarded as a character escape. The example in section 19.2 uses a single backslash according to the syntax for input masks in CSS2. However a CSS2 parser would treat this character as an escape character and remove the backslash from the string. Therefor all backslashes must be written as double backslashes (\\).

3.3 Change Description

6.2 Data Types

In order to support the WAP CSS Data Types the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
TYPES-1	<number> Integers and real numbers.	[CSS2] section 4.3.1	M	
TYPES-2	<length> The “px”, “em”, and “ex” length units.	[CSS2] section 4.3.2	M	TYPES-1
TYPES-3	<percentage>	[CSS2] section 4.3.3	M	TYPES-1
TYPES-4	<uri>	[CSS2] section 4.3.4	M	
TYPES-5	<color> The 16 HTML 4.0 colours.	[CSS2] section 4.3.6	M	
TYPES-6	<color> Numerical RGB specification.	[CSS2] section 4.3.6	M	
TYPES-7	<string>	[CSS2] section 4.3.10	M	

…

12.1 Font Family

In order to support the WAP CSS Font Family properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
FONTS-FAMILY-1	‘font-family’	[CSS2] section 15.2.2	M	FONTS-FAMILY-2 OR FONTS-FAMILY-3
FONTS-FAMILY-2	<generic-family> Generic font family name value	[CSS2] section 15.2.2	M	

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
FONTS-FAMILY-3	<family-name> Specific font family name value	[CSS2] section 15.2.2	O	TYPES-7

…

18.1 Access Keys: '-wap-accesskey'

'-wap-accesskey'

Value: none | <KeyCombinationList> | inherit
Initial: none
Applies to: all elements
Inherited: no
Percentages: N/A
Media: interactive

Definition: An *access key* is an additional, optional way to activate an element using a keypad key or key combination. What it means to *activate* an element depends on the type of element, and is specified for each element. For elements in WML, see the WML specification [WML2].

The actual list of supported keys and characters that may be used as access keys is platform dependent.

The access key is a key, not a function. The "mail" access key, for example, may be used on the device to launch the email application, but if used in the '-wap-accesskey' property, and assigned by the user agent, it will instead be used to activate the corresponding element - not launch the email application.

Definition: A *valid access key combination* matches the following syntax.

```
KeyCombinationList ::= KeyCombination (Separator KeyCombination)*

KeyCombination ::= Key ('-' Key)*

Separator ::= S+ | S* ',' S*

S ::= See [CSS2], section 4.1.1, a spaceKey ::= 'space' | Char | SpecialKey

SpecialKey ::= ModifierKey | FunctionKey | NavigationKey | EditKey | MiscKey
             | VolumeControlKey | ApplicationKey | PhoneKey | VendorKey

ModifierKey ::= 'accesskey' | CmdKey | OptKey | CtrlKey | ShiftKey | AltKey |
             WinKey | MetaKey | 'fn' | 'fcn' | 'caps'
```

…

```
Char ::= [0-9] | Nmstart

Nmstart ::= a single character as defined by nmstart in [CSS2] section 4.1.1
```

Note: Many of the keys have been adopted from the W3C working draft "User Interface for CSS3" [UICSS3]. Keys common on wireless devices have been added.

Conformance Requirement: The user agent MUST ignore invalid access key combinations.

Since the actual user input for keys is case insensitive, single characters that represent keys must be specified in uppercase or as entities. Special or modifier keys are specified in all lowercase so as to be distinguished from the characters representing keys.
...

18.1.2 Fallbacks and Multiple Assignments

The -wap-accesskey property provides a way to specify a prioritized list of access keys. To deal with the problem that not all access keys are available on all devices, or that access keys may have already been assigned, this property allows authors to specify a list of access keys that are tried in sequence to see if they can be assigned. The list of access keys is processed from the first to last character in the property string.

Conformance Requirement: In a ","separated list of access keys the user agent MUST select the first available access key in the list for assignment, and ignore all other access keys in the list.

In the following example the user agent will first try to assign the "snd" key to the element. If that key is unavailable it will try the "*" (\2a) key, and if that is unavailable it will try the "#" (\23) key.

```
a { -wap-accesskey: snd, \2a , \23 ;}
```

This is the same algorithm that is used for the CSS 'font-family' property.

Note: According to the [CSS2] syntax, the "*" and "#" are not allowed characters in an identifier. Therefor they must be coded to their unicode escape sequence.

Conformance Requirement: In a " " (space, as defined in [CSS2]) separated list of access keys the user agent MUST select all available access keys for assignment.

In the following example the user agent will assign the access keys "snd" and "*" (\2a) if either or both are available.

```
a { -wap-accesskey: snd \2a ;}
```

Conformance Requirement: In an access key list containing both "," and " " (space) operators, the user agent MUST assign the first available access key in the comma separated list. If a comma-separated entity is a list of space-separated access keys, the user agent MUST select all available access keys in the space-separated list for assignment. If none of the space-separated access keys can be assigned, the user agent MUST evaluate the next entity in the comma separated list.

In the following example the user agent will first try to assign the "snd" and "*" (\2a) keys. If either or both are available, the user agent will assign the available keys. If none of the space-separated access keys is available, the user agent will try to assign the "#" (\23) key.

```
a { -wap-accesskey: snd \2a , \23 ;}

...
```

19. WAP CSS Extension: Input

This section is normative.

Note: Input formatting and input validation are not features currently implemented as CSS2 [CSS2] properties.

In order to support the WAP Input CSS extensions the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
WAPEXT-INPUT-GENERAL-1	General concepts	Section 19.1	M	
WAPEXT-INPUT-FORMAT-2	‘-wap-input-format’	Section 19.2	M	TYPES-7
WAPEXT-INPUT-REQUIRED-3	‘-wap-input-required’	Section 19.3	M	

. . .

19.2. Input Format: '-wap-input-format'

'-wap-input-format'

Value: <format>

Initial: "*M"

Applies to: Input elements

Inherited: no

Percentages: N/A

Media: interactive

Definition: An *input format* is a sequence of characters that denotes a set of strings, L (F). The input format constrains the value space of the input data type. Strings in L (F) are valid input values for the input element to which the property is applied.

The <format> value is a *string* as defined in [CSS2] section 4. This means that input formats must be quoted.

Conformance Requirement: The user agent MUST ignore input masks that do not match the "input mask" syntax defined in [WML2].

Note: The "input mask" syntax is case sensitive.

Note: Since the backslash (\) character indicates escaping in CSS2, this character must be escaped with yet one more backslash as shown in the example below.

Here are some examples of input formats:

```
.phonenumber { -wap-input-format: "NNNNN\\-3N"; }

.pincode { -wap-input-format: "NNNN"; }
```