Wireless CSS Specification

Candidate Version 1.1 – 09 Jun 2004

Open Mobile Alliance

OMA-WAP-WCSS-V1_1-20040609-C

Contents

1. Scope

This section is informative.

Wireless CSS specifies a subset of CSS2 [CSS2] and mobile extensions defined by OMA. It can be used to style XHTML Mobile Profile [XHTMLMP] documents, as well as any other XML document.

The CSS2 subset contains core CSS functionality such as inheritance, cascading, selectors, and the syntax. It also contains properties for the box layout model, text, fonts, visual effects, colour, and lists.

The following CSS extensions are specified:

- **Marquee** - properties to create simple animation effects. Typically this will be used to create scrolling text.
- **Input** - properties to specify the format of the user input into a form control. It is the same feature as the WML 'format' and 'emptyok' attributes provide.
- **Accesskey** - property to specify an optional, additional way to activate an element in a document. It is similar to the feature the HTML 'accesskey' attribute provides.

A subset of CSS2 is also specified in the W3C, the "W3C CSS Mobile profile" Candidate Recommendation [CSSM]. Since both CSS2 variants follow CSS user agent semantics, conforming Wireless CSS user agents will accept valid W3C CSS Mobile Profile style sheets.

2. References

2.1 Normative References

[CSS2] "Cascading Style Sheets, level 2", CSS2 Specification, W3C Recommendation.
URL: http://www.w3.org/TR/1998/REC-CSS2-19980512

[IOPProc] "OMA Interoperability Policy and Process". Open Mobile Alliance™. OMA-IOP-Process-v1_0. URL:http//www.openmobilealliance.org/

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

[RFC2318] "The text/css Media Type", H. Lie, B. Bos, C. Lilley, March 1998.
URL: http://www.ietf.org/rfc/rfc2318.txt

[WML2] "WML 2.0", Open Mobile Alliance™. WAP-238-WML.
URL: http://www.openmobilealliance.org/

[XMLCSS] "Associating Style Sheets with XML documents", Version 1.0, W3C Recommendation.
URL: http://www.w3.org/1999/06/REC-xml-stylesheet-19990629

Field Code Changed

2.2 Informative References

[BASIC] "XHTML Basic", W3C Recommendation
URL: http://www.w3.org/TR/2000/REC-xhtml-basic-20001219

[CSSM] "CSS Mobile Profile 1.0", W3C Candidate Recommendation
URL: http://www.w3.org/TR/2002/CR-css-mobile-20020725

[GSM0230] "Digital cellular telecommunications system (Phase 2+); Man-Machine Interface (MMI) of the Mobile Station (MS)", GSM 02.30 version 7.1.0.
URL: http://www.3gpp.org/

[SELECTORS] "CSS3 module: W3C selectors", W3C Candidate Recommendation
URL: http://www.w3.org/TR/2001/CR-css3-selectors-20011113

[UICSS3] "CSS3 module: Basic User Interface", W3C Working Draft
URL: http://www.w3.org/TR/2002/WD-css3-ui-20020802/

[XHTMLMP] "XHTML Mobile Profile 1.1", Open Mobile Alliance™. OMA-WAP-XHTMLMP-V1_1. URL: http://www.openmobilealliance.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].

CSS property, descriptor, and pseudo-class names are delimited by single quotes.

CSS values are delimited by double quotes.

Data types are written as

<data type>

Notes are written as

Note: An informational message to the reader.

3.1.1 Tables

CSS properties and features are defined in tables with the following column names:

- **PROPERTY**
  Comma-separated list of property names.
- **FEATURE**
  Name of a CSS feature.
- **VALUE**
  “|”-separated list of datatypes and property values. (This is not the syntax of the property value.)
- **REFERENCE**
  Normative reference to the place where the property or function is defined.
- **STATUS**
  “M” – Mandatory; “O” – Optional.

3.2 Abbreviations

CSS	Cascading Style Sheets
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WCSS	Wireless Cascading Style Sheets
WML	Wireless Markup Language
XHTML	Extensible Hyper Text Markup Language
XML	Extensible Markup Language

4. Introduction

This section is informative.

Readers of this specification are assumed to be familiar with Cascading Style Sheets (CSS) as defined in [CSS2].

4.1 How to Read This Specification

The major part of this specification is a table with normative references to the CSS2 specification. The last few sections specify the extensions to CSS. Appendix A follows the syntax described in [IOPProc] and is the Static Conformance Requirements (SCR) for the whole specification.

It is recommended that the specification is read in the following way:

1. Read Appendix A to get an overall view of what parts of the specification are mandatory and what parts are optional. The SCR is a list of features of Wireless CSS.

2. For each feature in the SCR, details on conformance can be found in the referenced section.

A reference to CSS includes the referenced section and all sub-sections, unless stated otherwise.

4.2 Conformance

Conformance to this specification is defined in the Static Conformance Requirements (SCR) table in Appendix A. A conforming user agent may support additional CSS features.

This specification does not specify user interface presentation. It is not specified what a dotted line or a solid line should look like or how colors should be represented on the display. The look and feel depends on the device. This limits to what extent certain features can be tested. It is however expected that conformance testing will be carried out in such a way that it is possible to distinguish between a device that has implemented a particular feature and one that has not. It can be done for example by observing just the difference in effect of two different property values. A device that is unable to support a certain feature cannot claim to conform to that feature.

> **Note**: Conformance to this specification is intended to be stronger than for CSS2, where conformance can be claimed even if parts of the specification have not been implemented (see [CSS2] section 3.1). In Wireless CSS conformance can be claimed, in the Implementation Conformance Statement (ICS), only for features that have actually been implemented.

Fortunately, in many cases it is clear how a property value should be presented. Whether a font is “sans serif” or whether the text is indented five pixels can be tested. And if that test fails, the device does not conform to that feature.

Some features are optional and some features are mandatory. Here are some of the principles that have been used to determine what shall be mandatory and what shall be optional:

- For interoperability, most of the core CSS features are mandatory: syntax, selectors, cascading, inheritance, how values are calculated, etc.
- For consistency, when there are variants of the same property, such as ‘border-left’ and ‘border-right’ of the ‘border’ property, then all variants are required.
- In general, property values that are keywords, such as “larger”, “left”, and “smaller”, are required for that property.
- When a property takes values of only one type, that type is required.
- Some properties can take many different types of values. A property requires at least one type of values. The requirement-column in the SCR table is used to express the dependency between a property and the types of values it can take.
- Values expressed as colours always require support for the 16 HTML colour keywords and the RGB specification.

4.2.1 Exceptions when Presenting Documents in Certain Character Sets

There are style sheet properties that are not useful for documents in a certain character set. Most properties can be applied to a document in a Latin character set. Some properties for fonts and text, however, cannot be applied in a meaningful way to a Chinese-Japanese-Korean (CJK) document.

The general principle is that properties that are not useful for a certain character set are optional when a document in that character set is presented.

In the requirements tables, properties which are not useful when CJK documents are presented, are marked as optional ("CJK:O"). A user agent presenting a CJK document is not required to implement these properties, although they are mandatory when presenting a document in another character set.

In this specification, exceptions to the conformance requirements are made for CJK documents. Future versions may take other character sets into account, in addition to CJK.

4.2.2 Shorthand Properties

Shorthand properties allow authors to specify the value of several properties using a single property. It is a syntactical shortcut. All shorthand properties are mandatory. Since some of the properties which can be specified by the shorthand property may be optional – and not supported by the user agent – the properties which actually can be set depend on the user agent.

4.2.3 Colour and B&W Devices

Devices may be characterized as having one or more bits of color depth. Devices that have only one bit of color depth (assumed black and white for the purposes of this definition) should treat all color requests other than white as black, except where both foreground color and background color are specified and they are different. In this case the color with the higher hexadecimal value should default to white, and the color with the lower hexadecimal value should default to black. This prevents text being rendered invisible unintentionally.

4.3 The 'text/css' Media Type

Since Wireless CSS style sheets are either valid CSS2 style sheets [CSS2] or contain extensions expressed in valid CSS syntax, when delivered over the Internet, the 'text/css' [RFC2318] MIME media type can be used to represent Wireless CSS style sheets.

OMA has not defined a MIME media type specifically for Wireless CSS style sheets.

5. Selectors

This section is normative.

5.1 Pattern Matching

In order to support the Wireless CSS Selectors the user agent MUST implement all mandatory items ("M") in the following table.

FUNCTION	REFERENCE	STATUS
Universal selector	[CSS2] section 5.3	M
Type selectors	[CSS2] section 5.4	M
Descendant selectors	[CSS2] section 5.5	M
Link pseudo-class: :link	[CSS2] section 5.11.2	M
Link pseudo-class: :visited	[CSS2] section 5.11.2	O
Dynamic pseudo-class: :active	[CSS2] section 5.11.3	O
Dynamic pseudo-class: :focus	[CSS2] section 5.11.3	M
Class selectors	[CSS2] section 5.8.3	M
ID selectors	[CSS2] section 5.9	M
Grouping	[CSS2] section 5.2.1	M

Note: In order to conform to the W3C CSS Mobile profile the optional ":visited" and ":active" pseudo-classes must be supported, in addition to the mandatory selectors.

6. Syntax and Basic Data Types

This section is normative.

6.1 Syntax and Parsing

In order to support the Wireless CSS Syntax and Parsing the user agent MUST implement all mandatory items ("M") in the following table.

FUNCTION	REFERENCE	STATUS
CSS style sheet syntax	[CSS2] section 4.1	M
Rules for handling parsing errors	[CSS2] section 4.2	M
Algorithm for determining the character encoding	[CSS2] section 4.4	M
The "charset" at-rule in external style sheets	[CSS2] section 4.4	M

6.2 Data Types

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

FUNCTION	REFERENCE	STATUS
<number> Integers and real numbers.	[CSS2] section 4.3.1	M
<length> The "px", "em", and "ex" length units.	[CSS2] section 4.3.2	M
<percentage>	[CSS2] section 4.3.3	M
<uri>	[CSS2] section 4.3.4	M
<color> The 16 HTML 4.0 colours.	[CSS2] section 4.3.6	M
<color> Numerical RGB specification.	[CSS2] section 4.3.6	M
<string>	[CSS2] section 4.3.10	M

7. Assigning Property Values, Cascading, and Inheritance

This section is normative.

In order to support the Wireless CSS Assigning property values, cascading, and inheritance the user agent MUST implement all mandatory items ("M") in the following table.

FUNCTION	REFERENCE	STATUS
Assigning values	[CSS2] section 6.1	M
Inheritance	[CSS2] section 6.2, see text below on the 'inherit' value.	M
The "import" at-rule	[CSS2] section 6.3	M
The cascade	[CSS2] section 6.4	M

Note: [CSS2] section 6.4.4 specifies how non-CSS presentational hints (e.g. the presentational markup found in WML and XHTML) are used in the CSS cascade. It also specifies the same thing for the style rules in the XHTML "style" attribute.

7.1 The 'inherit' Value

In CSS2 the feature of property inheritance is defined in two, somewhat confusing, ways, either through an explicit property modifier ("inherit"), with no default action specified, or via a default (implied) action specification, which may or may not have an explicit property modifier available. In order to eliminate confusion and enhance interoperability, the feature of property inheritance MUST be supported in all cases where CSS2 [CSS2] defines the default inheritance feature to be enabled. Furthermore, the explicit value for overriding the inheritance feature also MUST be supported for those same properties. Support of the inheritance value for other properties, with no defined default, is optional for this profile.

Shortcut properties will follow the same rules. Any shortcut property for which all included properties have an inherit default defined, MUST support the inheritance feature and value. All other shortcut properties MAY optionally support inheritance values as defined in the CSS2 [CSS2] specification.

8. Media Types

This section is normative.

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

FUNCTION	VALUE	REFERENCE	STATUS
@media	handheld	[CSS2] section 7.2.1	M
	all	[CSS2] section 7.3	M

Note: A user agent MAY support additional media types, such as “aural” if it supports aural style sheets.

9. Associating Style Sheets with XML Document

This section is normative.

In order to support the Wireless CSS Associating style sheets with XML documents the user agent MUST implement the mandatory item (“M”) in the following table.

FUNCTION	REFERENCE	STATUS
XML style sheet processing instruction	[XMLCSS]	M

Here is an example of how the XML style sheet processing instruction is used in an XHTML document:

```
<?xml version="1.0" ?>
<?xml-stylesheet href="mobilestyle.css" media="handheld" type="text/css" ?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML MOBILE 1.1//EN"
"xhtml-mobile11.dtd">
<html>
  <head><title>Welcome to XHTML Mobile</title></head>
  <body>
   ...
  </body>
</html>
```

Authors SHOULD use the ”handheld” media type to indicate that the style sheet is appropriate for handheld devices.

> **Note:** Since Wireless CSS is based on CSS2 it does not support XML namespaces. The next version of Wireless CSS will support XML namespaces as a part of “CSS3 Module: W3C Selectors” [SELECTORS]. At the time of writing, this specification is still a W3C Working Draft.

10.Box Model

This section is normative.

10.1 Margin

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

PROPERTY	VALUE	REFERENCE	STATUS
margin-top, margin-right, margin-bottom, margin-left	<margin-width>	[CSS2] section 8.3	M
	inherit	[CSS2] section 8.3	O
margin	<margin-width>	[CSS2] section 8.3	M
	inherit	[CSS2] section 8.3	O

The <margin-width> datatype is defined in [CSS2] section 8.3.

10.2 Padding

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

PROPERTY	VALUE	REFERENCE	STATUS
padding-top, padding-right, padding-bottom, padding-left	<padding-width>	[CSS2] section 8.4	M
	inherit	[CSS2] section 8.4	O
padding	<padding-width>	[CSS2] section 8.4	M
	inherit	[CSS2] section 8.4	O

The <padding-width> datatype is defined in [CSS2] section 8.4.

10.3 Border Width

In order to support the Wireless CSS Border Width properties the user agent MUST implement all mandatory items (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
border-top-width, border-right-width, border-bottom-width, border-left-width	<border-width>	[CSS2] section 8.5.1	M
	inherit	[CSS2] section 8.5.1	O
border-width	<border-width>	[CSS2] section 8.5.1	M
	inherit	[CSS2] section 8.5.1	O

The <border-width> datatype is defined in [CSS2] section 8.5.1.

10.4 Border Colour

In order to support the Wireless CSS Border Colour properties the user agent MUST implement all mandatory items ("M") in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
border-top-color, border-right-color, border-bottom-color, border-left-color	<color>	[CSS2] section 8.5.2	M
	inherit	[CSS2] section 8.5.2	O
border-color	<color> | transparent	[CSS2] section 8.5.2	M
	inherit	[CSS2] section 8.5.2	O

10.5 Border Style

In order to support the Wireless CSS Border Style properties the user agent MUST implement all mandatory items ("M") in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
border-top-style, border-right-style, border-bottom-style, border-left-style	none | solid | dashed | dotted	[CSS2] section 8.5.3	M
	hidden | double, groove | ridge | inset | outset	[CSS2] section 8.5.3	O
	inherit	[CSS2] section 8.5.3	O
border-style	none | solid | dashed | dotted	[CSS2] section 8.5.3	M
	hidden | double, groove | ridge | inset | outset	[CSS2] section 8.5.3	O
	inherit	[CSS2] section 8.5.3	O

10.6 Border Shorthand Properties

In order to support the Wireless CSS Border Shorthand properties the user agent MUST implement all mandatory items ("M") in the following table.

PROPERTY	REFERENCE	STATUS
border-top, border-right, border-bottom, border-left, border	[CSS2] section 8.5.4	M

11.Colours and Background

This section is normative.

11.1 Foreground Colour

In order to support the Wireless CSS Foreground Colour properties the user agent MUST implement the mandatory item ("M") in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
color	<color> | inherit	[CSS2] section 14.1	M

11.2 Background Colour

In order to support the Wireless CSS Background Colour properties the user agent MUST implement all mandatory items ("M") in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
background-color	<color> | transparent	[CSS2] section 14.2	M
	inherit	[CSS2] section 14.2	O

11.3 Background Images

In order to support the Wireless CSS Background Images properties the user agent MUST implement all mandatory items ("M") in the following table and Background Colour.

PROPERTY	VALUE	REFERENCE	STATUS
background-image	<uri> | none	[CSS2] section 14.2	M
	inherit	[CSS2] section 14.2	O
background-repeat	repeat | repeat-x | repeat-y | no-repeat	[CSS2] section 14.2	M
	inherit	[CSS2] section 14.2	O
background-attachment	scroll | fixed	[CSS2] section 14.2	M
	inherit	[CSS2] section 14.2	O
background-position	<length> | <percentage>	[CSS2] section 14.2	O
	top | center | bottom | left | right	[CSS2] section 14.2	M
	inherit	[CSS2] section 14.2	O

11.4 Background Shorthand Property

In order to support the Wireless CSS Background Shorthand properties the user agent MUST implement the mandatory item ("M") in the following table.

PROPERTY	REFERENCE	STATUS
background	[CSS2] section 14.2	M

12.Fonts

This section is normative.

12.1 Font Family

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

PROPERTY	VALUE	REFERENCE	STATUS
font-family	<generic-family> | inherit	[CSS2] section 15.2.2	M CJK:O
	<family-name>	[CSS2] section 15.2.2	O

The <generic-family> and <family-name> datatypes are defined in [CSS2] section 15.2.2.

12.2 Font Style

In order to support the Wireless CSS Font Style properties the user agent MUST implement the mandatory item (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
font-style	normal | italic | oblique | inherit	[CSS2] section 15.2.3	M CJK:O

The “oblique” font style MAY be rendered as “italic”, according to [CSS2] section 15.2.3.

12.3 Font Variant

In order to support the Wireless CSS Font Variant properties the user agent MUST implement the mandatory item (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
font-variant	normal | small-caps | inherit	[CSS2] section 15.2.3	M CJK:O

12.4 Font Weight

In order to support the Wireless CSS Font Weight properties the user agent MUST implement the mandatory item (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	[CSS2] section 15.2.3	M CJK:O

It is reasonable to assume that many mobile devices will not support more than the two font-weights: “normal” and “bold”. Those two are defined in CSS2 [CSS2] to map to the numerical font weight values of “400” for “normal” and 700 for “bold”. It is also defined that a lower numerical font-weight must be lighter or equal to a higher numerical font-weight.

If user agent is unable to support different levels of font weights, it MUST map the keywords to the normal and bold keywords in the following way:

- Map font weight keywords “100”, “200”, “300”, “400”, and “lighter” to the same weight as “normal”.
- Map font weight keywords “500”, “600”, “700”, “800”, “900”, and “bolder” to the same weight as “bold”.

With two font weights the relative value “bolder” always equals “bold” and “lighter” always equals “normal”.

As specified in CSS2 [CSS2], only the strings “100”, “200”, “300”, “400”, “500”, “600”, “700”, “800”, “900” are legal as font weights. These weights are strings, not values.

12.5 Font Size

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

PROPERTY	VALUE	REFERENCE	STATUS
font-size	xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller | inherit	[CSS2] section 15.2.4	M CJK:O
	<length> | <percentage>	[CSS2] section 15.2.4	O CJK:O

12.6 Font Shorthand Property

In order to support the Wireless CSS Font Shorthand properties the user agent MUST implement the mandatory item (“M”) in the following table.

PROPERTY	REFERENCE	STATUS
font	[CSS2] section 15.2.5	M CJK:O

13.Lists

This section is normative.

13.1 Lists

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

PROPERTY	VALUE	REFERENCE	STATUS
list-style-type	disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none | inherit	[CSS2] section 12.6.2	M CJK:O[1]
list-style-position	inside | outside | inherit	[CSS2] section 12.6.2	O
list-style-image	<uri> | none | inherit	[CSS2] section 12.6.2	M CJK:O
list-style		[CSS2] section 12.6.2	M

User agent that does not recognize a numbering system, in 'list-style-type', SHOULD use “decimal”.

Since some list properties are optional, some user agents may not support all values of the shorthand property.

The user agent behaviour when the height of the list style image is greater than the highest inline box in the same line is implementation defined. This means that to guarantee interoperability, images specified by ‘list-style-image’ should be of a height less than or equal to the inline box height in the line in which the image is used.

[1] These values do not exist in Latin1 character set, so images are used instead. However for Japanese, they exist in JIS X 0208 as characters, so images are not used.

14.Text

This section is normative.

14.1 Indentation

In order to support the Wireless CSS Text Indentation properties the user agent MUST implement the mandatory item (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
text-indent	<length> | <percentage> | inherit	[CSS2] section 16.1	M CJK:O

14.2 Alignment

In order to support the Wireless CSS Text Alignment properties the user agent MUST implement all mandatory items (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
text-align	left | right | center | inherit	[CSS2] section 16.2	M
	justify	[CSS2] section 16.2	O

14.3 Decoration

In order to support the Wireless CSS Text Decoration properties the user agent MUST implement all mandatory items (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
text-decoration	none | blink	[CSS2] section 16.3.1	M
	underline	[CSS2] section 16.3.1	M CJK:O
	inherit | overline | line-through	[CSS2] section 16.3.1	O

14.4 Transformation

In order to support the Wireless CSS Text Transformation properties the user agent MUST implement the mandatory item (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
text-transform	capitalize | uppercase | lowercase | none | inherit	[CSS2] section 16.5	M CJK:O

14.5 White Space

In order to support the Wireless CSS Text White space properties the user agent MUST implement the mandatory item ("M") in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
white-space	normal | pre | nowrap | inherit	[CSS2] section 16.6	M CJK:O

15.Visual Effects

This section is normative.

In order to support the Wireless CSS Visual effects properties the user agent MUST implement all mandatory items (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
visibility	visible | hidden	[CSS2] section 11.2	M
	collapse | inherit	[CSS2] section 11.2	O

16.Visual Formatting

This section is normative.

16.1 Display Properties

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

PROPERTY	VALUE	REFERENCE	STATUS
display	none	[CSS2] section 9.2.5	M
	inline | block | list-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | inherit	[CSS2] section 9.2.5	O

16.2 Float Positioning

In order to support the Wireless CSS Float Positioning properties the user agent MUST implement all mandatory items (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
float	left | right | none	[CSS2] section 9.5.1	M
	inherit	[CSS2] section 9.5.1	O

When WCSS is applied to an XHTML Mobile Profile document, the user agent MUST support the 'float' property applied to the 'img' element. A user agent MAY support applying 'float' to other elements, according to [CSS2].

16.3 Float Flow Control

In order to support the Wireless CSS Float Flow Control properties the user agent MUST implement all mandatory items (“M”) in the following table and **Float Positioning**.

PROPERTY	VALUE	REFERENCE	STATUS
clear	left | right | both | none	[CSS2] section 9.5.2	M
	inherit	[CSS2] section 9.5.2	O

16.4 Content Width and Height

In order to support the Wireless CSS Content Width and Height properties the user agent MUST implement all mandatory items (“M”) in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
width	<length> | <percentage> | auto	[CSS2] section 10.2	M
	inherit	[CSS2] section 10.2	O
height	<length> | <percentage> | auto	[CSS2] section 10.5	M
	inherit	[CSS2] section 10.5	O
vertical-align	top | middle | bottom | baseline	[CSS2] section 10.8.1	M CJK:O
	sub | super | text-top | text-bottom	[CSS2] section 10.8.1	O
	<percentage> | <length>	[CSS2] section 10.8.1	O
	inherit	[CSS2] section 10.8.1	O

With WCSS it is possible for a content author to set the width and height of the content area, which is a CSS box, by using the 'width' and 'height' properties.

If the rendered content (see [CSS2]) does not fit into the content area it is modified (scaled, deformed, etc.) to fit the area. The modification method is implementation defined. (In CSS2 the author has some control over this situation by using the 'overflow' and 'clip' properties, which is not part of WCSS.)

Some replaced elements (see [CSS2]) such as <img> and <object> permit the content author to specify an alternative representation. If the content cannot be modified to fit into the content area, the alternative representation is used instead. For example if an image in an <img> element cannot be modified to fit into the content area, the value of the “alt” attribute is used as the rendering of the image. The alternative representation is also used in other situations, for example when the user has disabled images rendering using a user control.

An error is indicated to the user when the rendered content cannot be modified and no alternative representation is available. The user agent shall indicate the nature of the error, e.g. presentation error, resource error or content interpretation error, to the user in a meaningful way.

17.CSS Extension: Marquee

This section is normative.

In order to support the WAP Marquee CSS extension the user agent MUST implement all mandatory items ("M") in the following table.

PROPERTY	VALUE	REFERENCE	STATUS
display	-wap-marquee	Section 17.1	M
-wap-marquee-style	scroll | slide | alternate	Section 17.2	M
-wap-marquee-loop	<integer> | infinite	Section 17.3	M
-wap-marquee-dir	ltr | rtl	Section 17.4	M
-wap-marquee-speed	slow | normal | fast	Section 17.5	O

17.1 The '-wap-marquee' Property Value

The *marquee box* inherits and extends the characteristics of the principal block box ([CSS2] section 9.2.1). The content of the *marquee box* is animated according to the marquee properties defined in this section. As principal block boxes, *marquee boxes* participate in a block formatting context ([CSS2] section 9.4.1).

When the 'display' property is set to "-wap-marquee" the user agent MUST generate one *marquee box* with all marquee style properties set to their respective initial values.

Here is an example of how the "-wap-marquee" property value can be used to cause the text and a pictogram inside a paragraph to scroll horizontally:

```
<p style="display: -wap-marquee">
You have <object data="pict:///core/message/message" />!
</p>
```

And here is one that causes the content of list items to scroll:

```
<style type="text/css">
.scroll { display: -wap-marquee;}
</style>

...

<ol>
 <li class="scroll">Item one</li>
 <li class="scroll">Item two</li>
</ol>
```

17.2 Marquee Style: '-wap-marquee-style'

'-wap-marquee-style'

Value: scroll | slide | alternate

Initial: scroll

Applies to: Marquee elements

Inherited: no

Percentages: N/A

Media: visual

A *marquee style* is one of the following:

scroll
Start completely off one side, scroll all the way across and completely off, and then start again.

slide
Start completely off one side, scroll in, and stop as soon as the text touches the other margin.

alternate
Bouncing back and forth within the screen.

For each supported marquee style, the user agent MUST support plain text and, if supported, pictograms. It MAY, in addition, support any other content permitted by the content model of the element for which the marquee property is specified.

Reference processing models for the different marquee types are specified in the following sections. The text is normative and takes precedence over the illustrations.

17.2.1 Scroll

The "scroll" behaviour is defined by the following reference processing model:

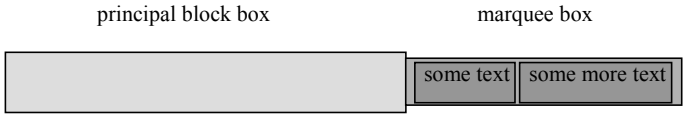
1. Set 'overflow' of the principal block to "hidden".
2. Set 'white-space' of the marquee box to "nowrap".
3. Set 'width' of the marquee box to a value based on the width of the boxes that it encloses.
4. Set the border and padding properties of the marquee box to "0px".
5. Set the number of marquee loops to the value of '-wap-marquee-loop'.
6. For each marquee loop:
   [Repeat until the number of marquee loops is zero.]
   a. For right-to-left direction, decrease 'margin-left' of the marquee box property from 100% to the negative value needed to ensure that the marquee box entirely overflows the principal block box. For left-to-right direction, use 'margin-right' instead.
   b. For right-to-left direction, maintain a constant width for the marquee box by animating 'margin-right' of the marquee box to compensate for changes in the value of 'margin-left'. For left-to-right direction, use 'margin-left' instead.
   c. Decrease the number of marquee loops by one.

The added complexity of keeping a constant width for the marquee box is necessary in case descendent boxes are laid out with respect to the marquee box's width.

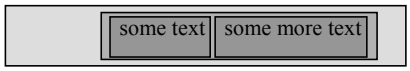
The figure below illustrates the three states for rendering the marquee element:

1. Begin loop state
2. Intermediate state
3. End loop state

Begin Loop State (right-to-left direction)



Intermediate State



End Loop State (right-to-left direction)

The begin loop state describes the presentation at the beginning of each cycle. The intermediate state is where the element content is visible in the principal block box (although some of it may be overflowing the principal block box). The end loop state describes the presentation at the end of each cycle.

17.2.2 Slide

The “slide” behaviour is defined by the following reference processing model:

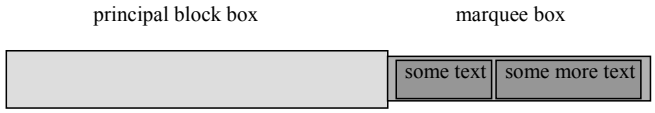
1. Set 'overflow' of the principal block box to “hidden”.
2. Set 'white-space' of the marquee box to “nowrap”.
3. Set ‘width’ of the marquee box to a value based on the width of the boxes that it encloses.
4. Set the border and padding properties of the marquee box to “0px”.
5. Set the number of marquee loops to the value of ‘-wap-marquee-loop’.
6. For each marquee loop:
   [Repeat until the number of marquee loops is zero.]
   a. For right-to-left direction, if ‘width’ of the marquee box is shorter than ‘width’ of the principal box, animate the marquee box's 'margin-left' from 100% to 0%. For left-to-right direction, use 'margin-right' instead.
   b. For right-to-left direction, if ‘width’ of the marquee box is longer than ‘width’ of the principal box, animate the marquee box's 'margin-left' from 100% to the negative value needed to make ‘margin-right’ 0%, and to guarantee that the whole element content can be presented on the screen. For a left-to-right direction, use 'margin-right' and ‘margin-left’, respectively, instead.

c. For right-left direction, maintain a constant width for the marquee box by animating the marquee box's 'margin-right' to compensate for changes in the value of the 'margin-left'. For left-to-right direction, use 'margin-left' instead.

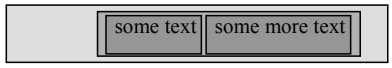
d. Decrease the number of marquee loops by one.

The figure below illustrates the three states for rendering the marquee element:

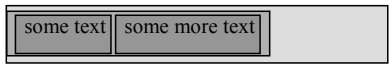
Begin Loop State (right-to-left direction)



Intermediate State



End Loop State (right-to-left direction, if the marquee box's 'width' is shorter than the principal box's 'width')



End Loop State (right-to-left direction, if the marquee box's 'width' is longer than the principal box's 'width')

17.2.3 Alternate

The "alternate" behaviour is defined by the following reference processing model:

1. Set 'overflow' of the principal block box to "hidden".
2. Set 'white-space' of the marquee box to "nowrap".
3. Set the 'width' of the marquee box to a value based on the width of the boxes that it encloses.
4. Set the border and padding properties of the marquee box' to "0px".
5. Set the number of marquee loops to the value of '-wap-marquee-loop'.
6. For each marquee loop:
   [Repeat until the number of marquee loops is zero.]

   a. For right-to-left direction, if the 'width' of the marquee box is shorter than the 'width' of the principal box, animate the marquee box's 'margin-right' from 0 to the value needed until the 'margin-left' property is animated to 0. If the decreased number of marquee loops is a value greater than **0**, the marquee box will go back to right until 'margin-right' is animated to 0 and the number of marquee loops is decreased. For left-to-right direction, use 'margin-right' and 'margin-left' properties, respectively, instead.

   b. For right-to-left direction, if the 'width' of the marquee box is longer than the 'width' of the principal box, animate the marquee box's 'margin-left' from 0 to the value needed until 'margin-right' is animated to 0 to guarantee that the whole element content can be presented on the screen once. If the decreased number of marquee loops is a value greater than 0, the marquee box will go back to right until 'margin-left' is

animated to 0, and the number of marquee loops is decreased. For left-to-right direction, use 'margin-right' and ‘margin-left’ properties, respectively, instead.

c. Maintain a constant width for the marquee box by animating the marquee box's 'margin-right' property to compensate for changes in the value of the 'margin-left' property (in the case of right-left direction). The 'margin-left' property is similarly animated for a left-to-right direction.

d. Decrease the number of marquee loops by one.

e. Change direction for the following loop, so that loops alternate between left-to-right and right-to-left.

The figure below illustrates the three states for rendering the marquee element:

Begin Loop State (right-to-left direction, if the marquee box's 'width' is shorter than the principal box's 'width')

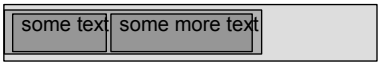
principal block box marquee box

Begin Loop State (right-to-left direction, if the marquee box's 'width' is longer than the principal box's 'width')

principal block box marquee box

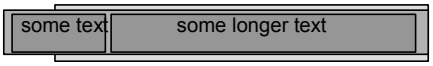
Intermediate State

End Loop State (right-to-left direction, if the number of loops is an odd number and if the marquee box's 'width' is shorter than the principal box's 'width')
)



End Loop State (right-to-left direction, if the number of loops is an even number and if the marquee box's 'width' is shorter than the principal box's 'width')
)

End Loop State (right-to-left direction, if the number of loops is an odd number and if the marquee box's 'width' is longer than the principal box's 'width')
)



End Loop State (right-to-left direction, if the number of loops is an even number and if the marquee box's 'width' is longer than the principal box's 'width')
)

17.3 Marquee Loop: '-wap-marquee-loop'

'-wap-marquee-loop'

Value: <integer> | infinite

Initial: 1

Applies to: Marquee elements

Inherited: no

Percentages: N/A

Media: visual

One *marquee loop* is the transition from the begin loop state to the end loop state. The initial value is "1" iterations. The maximum value is at least "16" iterations. If the user agent enforces a maximum number of loops, it MUST be at least 16. Any value above the maximum MUST be interpreted as the maximum value.

After the user agent has looped the specified number of times, the element MUST be rendered as if the 'display' property was set to "block".

If the value is "0", no looping occurs and the element is displayed as if it had finished looping a specified number of times.

17.4 Marquee Direction: '-wap-marquee-dir'

'-wap-marquee-dir'

Value: ltr | rtl

Initial: rtl

Applies to: Marquee elements

Inherited: no

Percentages: N/A

Media: visual

The *marquee direction* is the direction in which the marquee block should move across the principal block box. It is independent of the writing direction of blocks. The following two directions exist:

ltr
Left-to-right direction.

rtl
Right-to-left direction.

17.5 Marquee Speed: '-wap-marquee-speed'

'-wap-marquee-speed'

Value: slow | normal | fast

Initial: normal

Applies to: Marquee elements

Inherited: no

Percentages: N/A

Media: visual

The *marquee speed* is how fast the marquee box moves across the principal block box. The "normal", "slow", and "fast" values refer to entries in a table of marquee speeds computed and kept by the user agent. The table may be different depending on the content type of the marquee element.

For the marquee speed, the following relationship MUST always be true: slow < normal < fast.

18. CSS Extension: Access Keys

This section is normative.

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

PROPERTY	VALUE	REFERENCE	STATUS
-wap-accesskey	none | <KeyCombinationList> | inherit	Section 18.1	M

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

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.

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

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

CmdKey ::= 'cmd' | 'rcmd' | 'lcmd'

OptKey ::= 'opt' | 'ropt' | 'lopt'
```

```
CtrlKey ::= 'ctrl' | 'rctrl' | 'lctrl'

ShiftKey ::= 'shift' | 'rshift' | 'lshift'

AltKey ::= 'alt' | 'ralt' | 'lalt'

WinKey ::= 'win' | 'rwin' | 'lwin'

MetaKey ::= 'meta' | 'rmeta' | 'lmeta'

FunctionKey ::= 'f1' | 'f2' | 'f3' | 'f4' | 'f5' | 'f6' | 'f7' | 'f8' | 'f9' |
              'f10' | 'f11' | 'f12' | 'f13' | 'f14' | 'f15'

NavigationKey ::= 'tab' | 'esc' | 'enter' | 'return' | 'menu' | 'help' |
              'namemenu' | 'rcl' | 'snd' | ArrowKey | PageKey

ArrowKey ::= 'up' | 'down' | 'left' | 'right'

PageKey ::= 'home' | 'end' | 'pgup' | 'pgdn'

EditKey ::= 'bs' | 'del' | 'ins' | 'undo' | 'cut' | 'copy' | 'paste' | 'clr' |
              'sto'

MiscKey ::= 'prtsc' | 'sysrq' | 'scrlock' | 'pause' | 'brk' | 'numlock' | 'pwr'

VolumeControlKey ::= 'volumeup' | 'volumedown'

ApplicationKey ::= 'memo' | 'todo' | 'calendar' | 'mail' | 'address'

PhoneKey ::=  'phone-send' | 'phone-end' | 'phone-accept' /*See [GSM0230]*/

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 “User Interface for CSS3” [UICSS3]. Keys common on wireless devices have been added.

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.

18.1.1 Access Key Availability and Assignment

An access key is *unavailable* for assignment to an element when it has already been assigned to another element, is used for some other purpose by the user agent, or is not supported by the platform.

For elements whose access key is available, the user agent MUST use the specified access key and assign it to the element.

For elements whose access key is unavailable, the user agent SHOULD select another key and assign it to the element. If no other key is available, the access key MUST be ignored.

User agents SHOULD render the value of an access key in such a way as to emphasise its role and to distinguish it from other characters. For example, by underlining it, enclosing it in brackets, displaying it in reverse video or in a distinctive font.

Note: The author should not refer to the specific access key values in content or in documentation, as the values as specified in the style rule may not be used.

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.

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.

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 ;}
```

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 ;}
```

18.1.3 The XHTML 'accesskey' Attribute

The following conformance requirement is derived from the text in [CSS2] section 6.4.4 that specifies how non-CSS presentational hints are used in the CSS cascade.

If the XHTML 'accesskey' attribute is present on an element, it MUST be translated into the '-wap-accesskey' rule with specificity equal to zero. The rule is assumed to be at the start of the author style sheet and may be overridden by subsequent style sheet rules.

This means that the '-wap-accesskey' property will override the 'accesskey' attribute, if both are present on the same element.

19.CSS Extension: Input

This section is normative.

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

PROPERTY	VALUE	REFERENCE	STATUS
-wap-input-format	<format>	Section 19.2	M
-wap-input-required	true | false	Section 19.3	M

19.1 General Concepts

An *input element* is an XML element that represents a UI control that accepts text input from the user. Typically the element represents a form control such as a textbox.

> **Note**: In XHTML Basic and WML 2.0 the following elements accept text input from the user: textarea and input.

If a WAP Input property is applied to a non-input element, the element MUST be unaffected by the property.

The *input type* is the type of data that the input element accepts. The default data type is a character string. The value space is the set of legal finite-length sequences of characters (in the character set of the document).

The *input value* is the data from the user. An input value that is not in the value space of the input element is said to be *invalid*. The user agent SHOULD reject invalid input values and SHOULD request a new value from the user.

The user agent can use knowledge of the value space to optimise the user interface, for example changing the current input mode according to the specified value space.

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

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.

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"; }
```

19.2.1 The WML 'format' Attribute

The following conformance requirement is derived from the text in [CSS2] section 6.4.4 that specifies how non-CSS presentational hints are used in the CSS cascade.

If the WML 'format' attribute is present on an element it MUST be translated into the '-wap-input-format' rule with specificity equal to zero. The rule is assumed to be at the start of the author style sheet and may be overridden by subsequent style sheet rules.

This means that the '-wap-input-format' property will override the 'format' attribute, if both are present on the same element.

19.3 Required Input: '-wap-input-required'

'-wap-input-required'

Value: true | false

Initial: See requirements text below.

Applies to: Input elements

Inherited: no

Percentages: N/A

Media: interactive

The value space for *required input* is constrained to non-zero length strings.

Here is an example of using required input:

```
input { -wap-input-required: true; }
```

19.3.1 Combining '-wap-input-format' and '-wap-input-required'

The '-wap-input-required' property is combined with '-wap-input-format' property in order to determine the set of valid values of the input element.

When '-wap-input-required' has the value "false", the zero-length string is valid regardless of what is specified in '-wap-input-format'. When the value is "true", the zero-length string is invalid regardless of what is specified in '-wap-input-format'. If the author does not explicitly specify a value for the '-wap-input-required' property, then '-wap-input-format' fully defines the set of valid input values. This is illustrated in table 1.

The '-wap-input-required' property takes precedence over the '-wap-input-format' property when both are applied to the same element. If the '-wap-input-format' allows for zero-length string as a valid input value, then, if input is required according to the '-wap-input-required' property, it is anyway an illegal value.

		-wap-input-required		
		true	**false**	***not present***
-wap-input-format	**Property allows for a zero-length string (e.g. *M)**	According to the format but zero-length string is not allowed (e.g. M*M).	According to the format (e.g. *M).	According to the format (e.g. *M).
	Property requires a non-zero-length string (e.g. NNN)	According to the format (e.g. NNN).	According to the format but zero-length string also allowed.	According to the format (e.g. NNN).

Table 1Combinations of '-wap-input-format' and '-wap-input-required'.

19.3.2 The WML 'emptyok' Attribute

The following conformance requirement is derived from the text in [CSS2] section 6.4.4 that specifies how non-CSS presentational hints are used in the CSS cascade.

If the WML ‘emptyok’ attribute is present on an element it MUST be translated into the ‘-wap-input-required’ rule with specificity equal to zero. The rule is assumed to be at the start of the author style sheet and may be overridden by subsequent style sheet rules.

This means that the ‘-wap-input-required’ property will override the ‘emptyok’ attribute, if both are present on the same element.

Appendix A. Static Conformance Requirements (Normative)

In order to support Wireless CSS the user agent MUST implement all mandatory items ("M") in the following table, and SHOULD implement all optional items ("O").

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
WCSS-SELECTOR-C-001	Pattern matching	Section 5.1	M	
WCSS-SYNTAX-C-001	Syntax and parsing	Section 6.1	M	
WCSS-TYPES-C-001	Data types	Section 6.2	M	
WCSS-CASC-C-001	Assigning property values, cascading, and inheritance	Section 7	M	
WCSS-MEDIA-C-001	Media types	Section 8	M	
WCSS-ASOC-C-001	Associating style sheets with XML documents	Section 9	M	
WCSS-MARGIN-C-001	Margin properties	Section 10.1	M	
WCSS-PADDING-C-001	Padding properties	Section 10.2	M	
WCSS-BORDER-WIDTH-C-001	Border width	Section 10.3	M	
WCSS-BORDER-COLOR-C-001	Border colour	Section 10.4	M	
WCSS-BORDER-STYLE-C-001	Border style	Section 10.5	M	
WCSS-BORDER-SHORT-C-001	Border shorthand property	Section 10.6	M	
WCSS-FCOLOR-C-001	Foreground colour	Section 11.1	M	
WCSS-BCOLOR-C-001	Background colour	Section 11.2	M	
WCSS-BIMAGES-C-001	Background images	Section 11.3	O	
WCSS-BACKGROUND-C-001	Background shorthand property	Section 11.4	M	
WCSS-FONTS-FAMILY-C-001	Font family	Section 12.1	O	
WCSS-FONTS-STYLE-C-001	Font style	Section 12.2	M	
WCSS-FONTS-VARIANT-C-001	Font variant	Section 12.3	O	
WCSS-FONTS-WEIGHT-C-001	Font weight	Section 12.4	M	
WCSS-FONTS-SIZE-C-001	Font size	Section 12.5	O	
WCSS-FONTS-SHORT-C-001	Font shorthand property	Section 12.6	M	
WCSS-LISTS-C-001	Lists	Section 13.1	M	
WCSS-TEXT-INDENT-C-001	Text indentation	Section 14.1	M	

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
WCSS-TEXT-ALIGN-C-001	Text alignment	Section 14.2	M	
WCSS-TEXT-DEC-C-001	Text decoration	Section 14.3	M	
WCSS-TEXT-TRANS-C-001	Text transformation	Section 14.4	O	
WCSS-TEXT-WS-C-001	White space	Section 14.5	M	
WCSS-VEFFECT-C-001	Visual effects	Section 15	O	
WCSS-VFORM-DISP-C-001	Display properties	Section 16.1	M	
WCSS-VFORM-FLOAT-C-001	Float Positioning	Section 16.2	M	
WCSS-VFORM-CLEAR-C-001	Float Flow Control	Section 16.3	M	
WCSS-VFORM-SIZE-C-001	Content width and height	Section 16.4	M	
WCSS-WAPEXT-MARQUEE-C-001	CSS Extension: Marquee	Section 17	M	
WCSS-WAPEXT-ACC-C-001	CSS Extension: Access keys	Section 18	O	
WCSS-WAPEXT-INPUT-C-001	CSS Extension: Input	Section 19	M	

Appendix B. Change History (Informative)

B.1 Approved Version History

Reference	Date	Description
WAP-239-WCSS-20011026-a	26 Oct 2001	WCSS V1.0

B.2 Draft/Candidate Version 1.1 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-WAP-WCSS-V1_1	06 May 2003		The initial version of this document.
	27 Nov 2003		Template update. History changed to this style. URIs changed from www.wapforum.org to www.openmobilealliance.org and owing organisation from WAP Forum™ to Open Mobile Alliance™. Minor editorial corrections in references including changing deleting [CREQ] and replacing with [IOPProc] plus associated change in text.
Candidate Version OMA-WAP-WCSS-V1_1	09 Jun 2004	n/a	Status changed to Candidate by TP TP ref # OMA-TP-2004-0190-Browsing-V2_2-for-Candidate