

Introduction to WAP

The WAP protocol was designed to show internet contents on wireless clients, like mobile phones.

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML
- JavaScript
- XML

If you want to study these subjects first, find the tutorials on [W3Schools' Home Page](#).

What is WAP?

The wireless industry came up with the idea of WAP. The point of this standard was to show internet contents on wireless clients, like mobile phones.

- WAP stands for Wireless Application Protocol
 - WAP is an application communication protocol
 - WAP is used to access services and information
 - WAP is inherited from Internet standards
 - WAP is for handheld devices such as mobile phones
 - WAP is a protocol designed for micro browsers
 - WAP enables the creating of web applications for mobile devices.
 - WAP uses the mark-up language WML (not HTML)
 - WML is defined as an XML 1.0 application
-

The Wireless Application Protocol

The WAP protocol is the leading standard for information services on wireless terminals like digital mobile phones.

The WAP standard is based on Internet standards (HTML, XML and TCP/IP). It consists of a WML language specification, a WMLScript specification, and a Wireless Telephony Application Interface (WTAI) specification.

WAP is published by the WAP Forum, founded in 1997 by Ericsson, Motorola, Nokia, and Unwired Planet. Forum members now represent over 90% of the global handset market, as well as leading infrastructure providers, software developers and other organizations. You can read more about the WAP forum at our [WAP Forum page](#).

WAP Micro Browsers

To fit into a small wireless terminal, WAP uses a Micro Browser.

A Micro Browser is a small piece of software that makes minimal demands on hardware, memory and CPU. It can display information written in a restricted mark-up language called WML.

The Micro Browser can also interpret a reduced version of JavaScript called WMLScript.

What is WML?

WML stands for **W**ireless **M**arkup **L**anguage. It is a mark-up language inherited from HTML, but WML is based on XML, so it is much stricter than HTML.

WML is used to create pages that can be displayed in a WAP browser. Pages in WML are called DECKS. Decks are constructed as a set of CARDS.

What is WMLScript?

WML uses WMLScript to run simple code on the client. WMLScript is a light JavaScript language. However, WML scripts are not embedded in the WML pages. WML pages only contains references to script URLs. WML scripts need to be compiled into byte code on a server before they can run in a WAP browser.

Visit our [WMLScript tutorial](#) to learn more about scripting in WML documents.

Examples of WAP use

- Checking train table information
- Ticket purchase
- Flight check in
- Viewing traffic information
- Checking weather conditions
- Looking up stock values
- Looking up phone numbers
- Looking up addresses
- Looking up sport results

FAQ about WAP

These are frequently asked question about WAP:

- What is WAP?
- Who is WAP for?
- How does WAP relate to standardization bodies?
- How is WAP related to Internet standards?
- What is the status of WAP?
- What is the future of WAP?

We will try to answer most of these questions. In the meantime read the answers at: <http://www.wapforum.org/faqs/index.htm>.

WAP Homepages

WAP homepages are not very different from HTML homepages. The markup language used for WAP is WML (Wireless Markup Language). WML uses tags - just like HTML - but the syntax is stricter and conforms to the XML 1.0 standard.

WML pages have the extension *.wml, just like HTML pages have the extension *.html.

WML Tags

WML is mostly about text. Tags that would slow down the communication with handheld devices are not a part of the WML standard. The use of tables and images is strongly restricted.

Since WML is an XML application, all tags are case sensitive (<wml> is not the same as <WML>), and all tags must be properly closed.

WML Decks and Cards

WML pages are called DECKS. They are constructed as a set of CARDS, related to each other with links. When a WML page is accessed from a mobile phone, all the cards in the page are downloaded from the WAP server. Navigation between the cards is done by the phone computer - inside the phone - without any extra access trips to the server.

Example WML document:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>

<card id="HTML" title="HTML Tutorial">
  <p>Our HTML Tutorial is an award winning
  tutorial from W3Schools.</p>
</card>

<card id="XML" title="XML Tutorial">
  <p>Our XML Tutorial is an award winning
  tutorial from W3Schools.</p>
</card>

</wml>
```

As you can see from the example, the WML document is an XML document. The DOCTYPE is defined to be wml, and the DTD is accessed at www.wapforum.org/DTD/wml_1.1.xml.

The document content is inside the <wml>...</wml> tags. Each card in the document is inside <card>...</card> tags, and actual paragraphs are inside <p>...</p> tags. Each card element has an id and a title.

Decks and Cards

WML pages are often called "decks". A deck contains a set of cards. A card element can contain text, markup, links, input-fields, tasks, images and more. Cards can be related to each other with links.

When a WML page is accessed from a mobile phone, all the cards in the page are downloaded from the WAP server. Navigation between the cards is done by the phone computer - inside the phone - without any extra access trips to the server:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

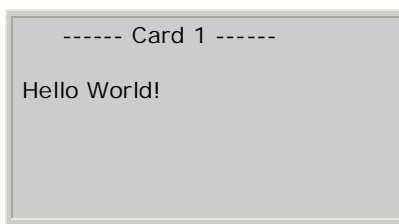
<wml>

<card id="no1" title="Card 1">
  <p>Hello World!</p>
</card>

<card id="no2" title="Card 2">
  <p>Welcome to our WAP Tutorial!</p>
</card>

</wml>
```

The result will look something like this in a mobile phone (note that only one card is displayed at a time):



Paragraphs and Line Breaks

A WML card can be set up to display the paragraph and line break functions of WML:

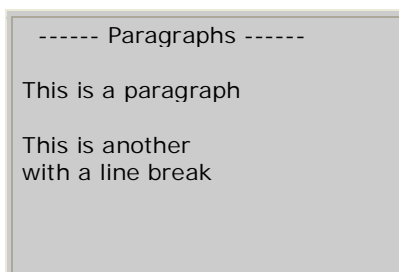
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Paragraphs">

<p>This is a paragraph</p>
<p>This is another<br/>with a line break</p>

</card>
</wml>
```

The result will look something like this in a mobile phone:



Text Formatting

A WML card can be set up to display the text formatting functions of WML:

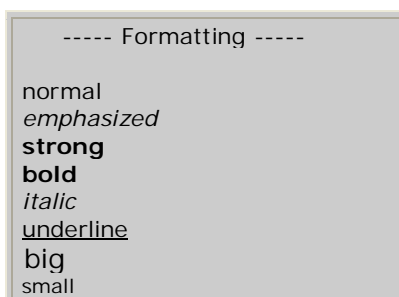
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Formatting">

<p>
normal<br/>
<em>emphasized</em><br/>
<strong>strong</strong><br/>
<b>bold</b><br/>
<i>italic</i><br/>
<u>underline</u><br/>
<big>big</big><br/>
<small>small</small>
</p>

</card>
</wml>
```

The result will look something like this in a mobile phone (don't take it for granted that all formatting tags will render as expected):



Tables

A WML card can be set up to display the table functions of WML:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Table">
```

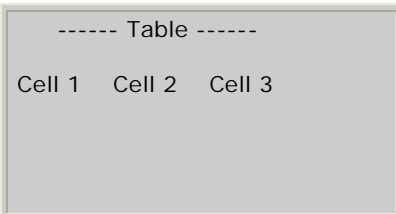
```

<p>
<table columns="3">
  <tr>
    <td>Cell 1</td>
    <td>Cell 2</td>
    <td>Cell 3</td>
  </tr>
</table>
</p>

</card>
</wml>

```

The result will look something like this in a mobile phone:



Links

A WML card can be set up to display the anchor functions of WML.

The <anchor> tag

The <anchor> tag always has a task ("go", "prev", or "refresh") specified. The task defines what to do when the user selects the link. In this example, when the user selects the "Next page" link, the task says "go to the file test.wml":

```

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card title="Anchor Tag">
    <p><anchor>Next page <go href="test.wml"/></anchor></p>
  </card>
</wml>

```

The <a> tag

The <a> tag always performs a "go" task, with no variables. The example below does the same as the <anchor> tag example:

```

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card title="A Tag">
    <p><a href="test.wml">Next page</a></p>
  </card>
</wml>

```

Image

A WML card can be set up to display an image:

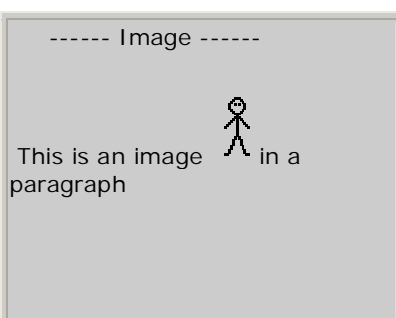
```

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card title="Image">
    <p>This is an image
    
    in a paragraph</p>
  </card>
</wml>

```

The result will look something like this in a mobile phone:



Note that .wbmp is the only image type that can be displayed in a WAP browser.

Input Fields

A WML card can be set up to let a user enter information, as demonstrated in this example:

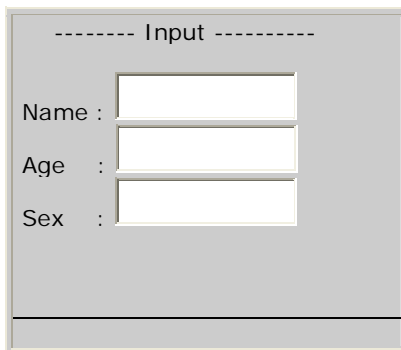
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Input">

<p>
Name: <input name="Name" size="15"/><br/>
Age: <input name="Age" size="15" format="*N"/><br/>
Sex: <input name="Sex" size="15"/>
</p>

</card>
</wml>
```

The result will look something like this in a mobile phone:



----- Input -----

Name :

Age :

Sex :

Select and Option

A WML card, can be set up to display the select and option functions of WML:

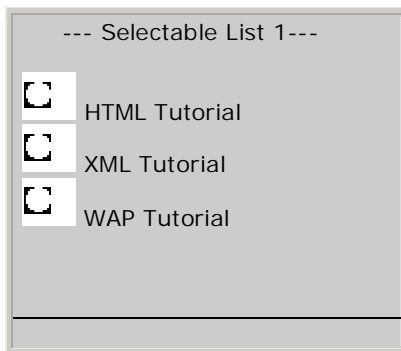
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Selectable List 1">

<p>
<select>
<option value="htm">HTML Tutorial</option>
<option value="xml">XML Tutorial</option>
<option value="wap">WAP Tutorial</option>
</select>
</p>

</card>
</wml>
```

The result will look something like this in a mobile phone:



--- Selectable List 1---

HTML Tutorial

XML Tutorial

WAP Tutorial

In the example below, the result is a selectable list where the user can select more than one item:

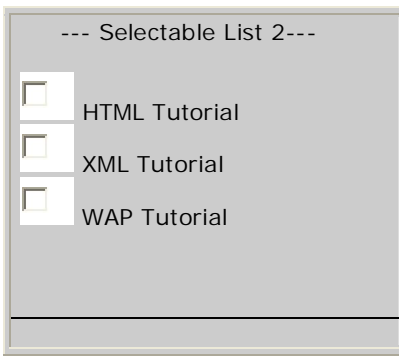
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Selectable List 2">

<p>
<select multiple="true">
<option value="htm">HTML Tutorial</option>
<option value="xml">XML Tutorial</option>
<option value="wap">WAP Tutorial</option>
</select>
</p>

</card>
</wml>
```

The result will look something like this in a mobile phone:



Fieldset

A WML card, can be set up to display the fieldset function of WML:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Fieldset">

<p>
<fieldset title="CD Info">
Title: <input name="title" type="text"/><br/>
Prize: <input name="prize" type="text"/>
</fieldset>
</p>

</card>
</wml>
```

The result will look something like this in a mobile phone:



A task specifies what action to perform when an event occurs.

Go Task

The <go> task represents the action of switching to a new card.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card>
<p>
<anchor>Go To Test<go href="test.wml"/></anchor>
</p>
</card>
</wml>
```

Prev Task

The <prev> task represents the action of going back to the previous card.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card>
<p>
<anchor>Previous Page<prev/></anchor>
</p>
</card>
</wml>
```

Refresh Task

The <refresh> task refreshes some specified card variables. If any of the variables are shown on the screen, this task also refreshes the screen.

The following example uses an `<anchor>` tag to add a "Refresh this page" link to the card. When the user clicks on the link, he or she refreshes the page and the value of the variable `x` will be set to 30:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card>
    <p>
      <anchor>
        Refresh this page
        <go href="thispage.wml"/>
        <refresh><setvar name="x" value="30"/></refresh>
      </anchor>
    </p>
  </card>
</wml>
```

Noop Task

The `<noop>` task says that nothing should be done (noop stands for "no operation"). This tag is used to override deck-level elements.

The `<do>` tag can be used to activate a task when the user clicks on a word/phrase on the screen.

The following example uses a `<do>` tag to add a "Back" link to the card. When the user clicks on the "Back" link, he or she should be taken back to the previous card. But the `<noop>` tag prevents this operation; when the user clicks on the "Back" link nothing will happen:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card>
    <p>
      <do name="back" type="prev" label="Back">
        <noop/>
      </do>
    </p>
  </card>
</wml>
```

Task Elements

Start tag	Purpose	WML
<code><go></code>	Represents the action of switching to a new card	1.1
<code><noop></code>	Says that nothing should be done (noop stands for "no operation"). Used to override deck-level elements	1.1
<code><prev></code>	Represents the action of going back to the previous card	1.1
<code><refresh></code>	Refreshes some specified card variables. If any of the variables are shown on the screen, this task also refreshes the screen	1.1

Timer

A WML card can be set up to use the timer function of WML. The time unit of the timer is 1/10 of a second.

The following example will display a message for 3 seconds, and then take you to the file "test.wml":

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card ontimer="test.wml">
    <timer value="30"/>
    <p>Some Message</p>
  </card>
</wml>
```

Variables

When a user switches from card to card in a deck, we need to store data in variables. WML variables are case sensitive.

Specify a Variable with the Setvar Command

When someone executes a task (like `go`, `prev`, and `refresh`), the `setvar` element can be used to set a variable with a specified value.

The following example will create a variable named `i` with a value of 500:

```
<setvar name="i" value="500"/>
```

The name and value attributes are required.

Specify a Variable through an Input Element

Variables can also be set through an input element (like input, select, option, etc.).

The following example will create a variable named *schoolname*:

```
<card id="card1">
  <select name="schoolname">
    <option value="HTML">HTML Tutorial</option>
    <option value="XML">XML Tutorial</option>
  </select>
</card>
```

To use the variable we created in the example above:

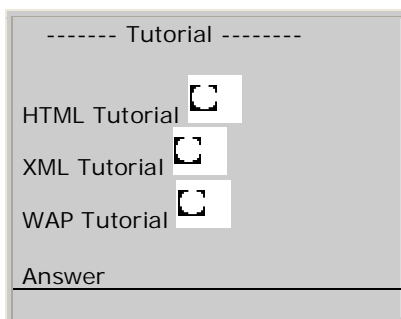
```
<card id="card2">
  <p>You selected: $(schoolname)</p>
</card>
```

A WML deck with two cards - one for user input and one for displaying the result - can be set up, as demonstrated in this example:

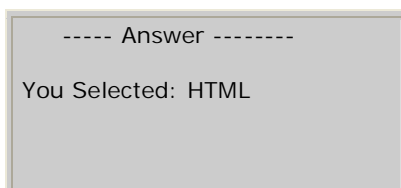
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card id="card1" title="Tutorial">
    <do type="accept" label="Answer">
      <go href="#card2"/>
    </do>
    <p><select name="name">
      <option value="HTML">HTML Tutorial</option>
      <option value="XML">XML Tutorial</option>
      <option value="WAP">WAP Tutorial</option>
    </select></p>
  </card>
  <card id="card2" title="Answer">
    <p>You selected: $(name)</p>
  </card>
</wml>
```

The first card might look something like this in a mobile phone:



The second card might look like this:



Example Explained

The Prolog

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
```

The first lines in the WML document are called the prolog. The prolog defines that this is an XML document, it then defines the XML version, and the DTD to be referenced.

The Deck

```
<wml> ..... </wml>
```

The deck is the WML document itself. It is embedded within `<wml>` tags

The Cards

```
<card> ..... </card>
```

Cards are always displayed one at the time. This WML deck contains two cards - one for user input and one for displaying the result.

The <do> element

```
<do> ... </do>
```

The first card has a <do> element that defines an event to be triggered. The **type="accept"** attribute of the <do> element causes the **label="Answer"** to be displayed in the lower left corner of the display.

The Event

The <go> element triggers when the user clicks the <do> label. The **href="#card2"** attribute of the <go> element causes card2 to be displayed on the screen.

The Variable

Card2 displays the **\$(name)** variable from card1, because variables are valid across cards.

Validating your WML

To help you validate your wml, we have used The Microsoft's XML parser to create a wml validator. Paste your wml in the text area, and validate it by pressing the validate button.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM/DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="card1" title="Card 1">
<p>Hello World</p>
</card>
</wml>
```

NOTE: This example validates the wml file according to its DTD, this validation is only supported by Internet Explorer.

Validating your WML file

You can also validate your wml files, simply by typing the url of your wml file and press the submit button

File Name:

If you want to validate an error free WML file you can paste this address into the name field: http://www.w3schools.com/wap/demo_helloworld.wml

NOTE: This example validates the wml file according to its DTD, this validation is only supported by Internet Explorer.

NOTE: If you get an error message saying "Access denied" when accessing this file, it is because your Internet Explorer security setting do not allow access across domains.

Compiling WML Code

To test and compile your WML code, you can download the [Nokia Mobile Internet Toolkit](#) for free. The Nokia Mobile Internet Toolkit supports the complete WAP 2.0 specification, including XHTML and CSS.

If you place WML code on your IIS or Apache server, you don't need to compile it. This is a job for the WAP Gateway. Simply host your native WML code on your server.



If you have a WAP enabled phone, you can access the WAP pages on www.W3Schools.com.

Turn on your WAP phone and enter the following address:

<http://www.w3schools.com/wap.wml>

The web address above contains links to:

- [W3Schools' HTML 4.01 Reference](#)
- [W3Schools' Web Building Glossary](#)

Deck / Card Elements

Start tag	Purpose	WML
<access>	Defines information about the access control of a deck	1.1
<card>	Defines a card in a deck	1.1
<head>	Contains information about the document	1.1
<meta>	Defines meta information about the document	1.1

<template>	Defines a code template for all the cards in a deck	1.1
<wml>	Defines a WML deck (WML root)	1.1
<!-->	Defines a comment	1.1

Text Elements

Start tag	Purpose	WML

	Defines a line break	1.1
<p>	Defines a paragraph	1.1
<table>	Defines a table	1.1
<td>	Defines a table cell (table data)	1.1
<tr>	Defines a table row	1.1

Text Formatting Tags

Start tag	Purpose	WML
	Defines bold text	1.1
<big>	Defines big text	1.1
	Defines emphasized text	1.1
<i>	Defines italic text	1.1
<small>	Defines small text	1.1
	Defines strong text	1.1
<u>	Defines underlined text	1.1

Anchor Elements

Start tag	Purpose	WML
<a>	Defines an anchor (a link)	1.1
<anchor>	Defines an anchor (a link)	1.1

Image Elements

Start tag	Purpose	WML
	Defines an image	1.1

Event Elements

Start tag	Purpose	WML
<do>	Activates a task when the user clicks on a word/phrase on the screen	1.1
<onevent>	Contains code to be executed when one of the following events occurs: onenterbackward, onenterforward, onpick, ontimer	1.1
<postfield>	Contains information to be sent to the server along with a <go> tag	1.1

Task Elements

Start tag	Purpose	WML
<go>	Represents the action of switching to a new card	1.1
<noop>	Says that nothing should be done (noop stands for "no operation"). Used to override deck-level elements	1.1
<prev>	Represents the action of going back to the previous card	1.1
<refresh>	Refreshes some specified card variables. If any of the variables are shown on the screen, this task also refreshes the screen	1.1

Input Elements

Start tag	Purpose	WML
<fieldset>	Used to group together related elements in a card	1.1
<input>	Defines an input field (a text field where the user can enter some text)	1.1
<optgroup>	Defines an option group in a selectable list	1.1
<option>	Defines an option in a selectable list	1.1
<select>	Defines a selectable list	1.1

Variable Elements

Start tag	Purpose	WML
<setvar>	Sets a variable to a specified value in a <go>, <prev>, or <refresh> task	1.1
<timer>	Defines a card timer	1.1

Character Entities

Result	Description	Entity Name	Entity Number
&	ampersand	&	&
'	apostrophe	'	'
>	greater-than	>	>
<	less-than	<	<
	non-breaking space	 	
"	quotation mark	"	"
	soft hyphen	­	­

The Complete WML DTD:

```
<!--
Wireless Markup Language (WML) Document Type Definition.

Copyright Wireless Application Protocol
Forum Ltd., 1998,1999.
All rights reserved.

WML is an XML language. Typical usage:
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
...
</wml>

Terms and conditions of use are
available from the Wireless
Application Protocol Forum Ltd. web site at
http://www.wapforum.org/docs/copyright.htm.
-->

<!ENTITY % length "CDATA">
<!-- [0-9]+ for pixels or [0-9]+%"
for percentage length -->
<!ENTITY % vdata "CDATA">
<!-- attribute value possibly containing
variable references -->
<!ENTITY % HREF "%vdata;">
<!-- URI, URL or URN designating a
hypertext node. May contain variable references -->
<!ENTITY % boolean "(true|false)">
<!ENTITY % number "NMTOKEN">
<!-- a number, with format [0-9]+ -->
<!ENTITY % coreattrs "id ID #IMPLIED
class CDATA #IMPLIED">

<!ENTITY % emph
"em | strong | b | i | u | big | small">
<!ENTITY % layout "br">

<!ENTITY % text "#PCDATA | %emph;">

<!-- flow covers "card-level" elements,
such as text and images -->
<!ENTITY % flow
"%text; | %layout; | img | anchor | a | table">

<!-- Task types -->
<!ENTITY % task "go | prev | noop | refresh">

<!-- Navigation and event elements -->
<!ENTITY % navelmts "do | onevent">

<!--===== Decks and Cards =====>

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

<!-- card intrinsic events -->
<!ENTITY % cardev
"onenterforward %HREF; #IMPLIED
onenterbackward %HREF; #IMPLIED
ontimer %HREF; #IMPLIED"
>

<!-- card field types -->
<!ENTITY % fields
"%flow; | input | select | fieldset">

<!ELEMENT card (onevent*, timer?, (do | p)*)>
<!ATTLIST card
title %vdata; #IMPLIED
newcontext %boolean; "false"
ordered %boolean; "true"
xml:lang NMTOKEN #IMPLIED
%cardev;
%coreattrs;
>

<!--===== Event Bindings =====>

<!ELEMENT do (%task;)>
<!ATTLIST do
type CDATA #REQUIRED
label %vdata; #IMPLIED
name NMTOKEN #IMPLIED
optional %boolean; "false"
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT onevent (%task;)>
```

```

<!ATTLIST onevent
type CDATA #REQUIRED
%coreattrs;
>

<!--===== Deck-level declarations =====>

<!ELEMENT head ( access | meta )+>
<!ATTLIST head
%coreattrs;
>

<!ELEMENT template (%navelmts;)*>
<!ATTLIST template
%cardev;
%coreattrs;
>

<!ELEMENT access EMPTY>
<!ATTLIST access
domain CDATA #IMPLIED
path CDATA #IMPLIED
%coreattrs;
>

<!ELEMENT meta EMPTY>
<!ATTLIST meta
http-equiv CDATA #IMPLIED
name CDATA #IMPLIED
forua %boolean; #IMPLIED
content CDATA #REQUIRED
scheme CDATA #IMPLIED
%coreattrs;
>

<!--===== Tasks =====>

<!ELEMENT go (postfield | setvar)*>
<!ATTLIST go
href %HREF; #REQUIRED
sendreferer %boolean; "false"
method (post|get) "get"
accept-charset CDATA #IMPLIED
%coreattrs;
>

<!ELEMENT prev (setvar)*>
<!ATTLIST prev
%coreattrs;
>

<!ELEMENT refresh (setvar)*>
<!ATTLIST refresh
%coreattrs;
>

<!ELEMENT noop EMPTY>
<!ATTLIST noop
%coreattrs;
>

<!--===== postfield =====>

<!ELEMENT postfield EMPTY>
<!ATTLIST postfield
name %vdata; #REQUIRED
value %vdata; #REQUIRED
%coreattrs;
>

<!--===== variables =====>

<!ELEMENT setvar EMPTY>
<!ATTLIST setvar
name %vdata; #REQUIRED
value %vdata; #REQUIRED
%coreattrs;
>

<!--===== Card Fields =====>

<!ELEMENT select (optgroup|option)+>
<!ATTLIST select
title %vdata; #IMPLIED
name NMTOKEN #IMPLIED
value %vdata; #IMPLIED
iname NMTOKEN #IMPLIED
ivalue %vdata; #IMPLIED
multiple %boolean; "false"
tabindex %number; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT optgroup (optgroup|option)+ >
<!ATTLIST optgroup
title %vdata; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;

```

```

>
<!ELEMENT option (#PCDATA | onevent)*>
<!ATTLIST option
value %vdata; #IMPLIED
title %vdata; #IMPLIED
onpick %HREF; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT input EMPTY>
<!ATTLIST input
name NMTOKEN #REQUIRED
type (text|password) "text"
value %vdata; #IMPLIED
format CDATA #IMPLIED
emptyok %boolean; "false"
size %number; #IMPLIED
maxlength %number; #IMPLIED
tabindex %number; #IMPLIED
title %vdata; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT fieldset (%fields; | do)* >
<!ATTLIST fieldset
title %vdata; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT timer EMPTY>
<!ATTLIST timer
name NMTOKEN #IMPLIED
value %vdata; #REQUIRED
%coreattrs;
>

<!--===== Images =====>

<!ENTITY % IAlign "(top|middle|bottom)" >

<!ELEMENT img EMPTY>
<!ATTLIST img
alt %vdata; #REQUIRED
src %HREF; #REQUIRED
localsrc %vdata; #IMPLIED
vspace %length; "0"
hspace %length; "0"
align %IAlign; "bottom"
height %length; #IMPLIED
width %length; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!--===== Anchor =====>

<!ELEMENT anchor
( #PCDATA | br | img | go | prev | refresh )*>
<!ATTLIST anchor
title %vdata; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT a ( #PCDATA | br | img )*>
<!ATTLIST a
href %HREF; #REQUIRED
title %vdata; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!--===== Tables =====>

<!ELEMENT table (tr)+>
<!ATTLIST table
title %vdata; #IMPLIED
align CDATA #IMPLIED
columns %number; #REQUIRED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT tr (td)+>
<!ATTLIST tr
%coreattrs;
>

<!ELEMENT td
( %text; | %layout; | img | anchor | a )*>
<!ATTLIST td
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

```

```
<!--== Text layout and line breaks ==-->

<!ELEMENT em (%flow;)*>
<!ATTLIST em
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT strong (%flow;)*>
<!ATTLIST strong
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT b (%flow;)*>
<!ATTLIST b
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT i (%flow;)*>
<!ATTLIST i
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT u (%flow;)*>
<!ATTLIST u
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT big (%flow;)*>
<!ATTLIST big
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT small (%flow;)*>
<!ATTLIST small
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ENTITY % TAlign "(left|right|center)">
<!ENTITY % WrapMode "(wrap|nowrap)" >
<!ELEMENT p (%fields; | do)*>
<!ATTLIST p
align %TAlign; "left"
mode %WrapMode; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ELEMENT br EMPTY>
<!ATTLIST br
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>

<!ENTITY quot "&#34;">
<!-- quotation mark -->
<!ENTITY amp "&#38;#38;">
<!-- ampersand -->
<!ENTITY apos "&#39;">
<!-- apostrophe -->
<!ENTITY lt "&#38;#60;">
<!-- less than -->
<!ENTITY gt "&#62;">
<!-- greater than -->
<!ENTITY nbsp "&#160;">
<!-- non-breaking space -->
<!ENTITY shy "&#173;">
<!-- soft hyphen (discretionary hyphen) -->

<!--
Copyright Wireless Application Protocol
Forum Ltd., 1998,1999.
All rights reserved.
-->
```