



Contec Innovations Inc.

Web Content Developers Guide

How to Leverage the Power of the Hornet Rendering Engine

April 2007

Prepared by: Gordon Du

Contec Innovations Inc.
301 – 2071 Kingsway Avenue
Port Coquitlam, British Columbia
Canada, V3C 6N2

COPYRIGHT AND NON-DISCLOSURE NOTICE

This document is highly confidential and may not be copied, forwarded or disclosed in whole or in part to a third party without the express written consent of Contec Innovations Inc. Failure to comply with this notice may result in prosecution to the fullest extent of the law. © 2004 Contec Innovations Inc. All rights reserved.



Table of Contents

1 OVERVIEW	2
1.1 ABOUT HORNET RENDERING ENGINE	2
2 DEFINITIONS	2
3 USE OF THE HORNET RENDERING ENGINE	3
3.1 WEB CONTENT CUSTOMIZATION FOR MOBILE DEVICES	4
3.1.1 Real World Example.....	4
3.1.2 How do we do it - Hornet Rendering Tag Attributes	5
3.1.3 Sample Page with Tag Attributes and the Results.....	6
3.2 RELOADING OF PAGES IF VISITED BY MOBILE DEVICE	9
3.3 LIMITATIONS OF THE RENDERING ENGINE	10
3.4 TESTING	11
3.5 FEEDBACK	11



1 Overview

This guide is prepared mainly for web developers who need to create web pages with optimal results for both normal and mobile browser use, through Contec's Hornet rendering engine.

1.1 About Hornet Rendering Engine

Hornet rendering engine allows automatic translation of web content from normal HTML code to mobile browser code, which enables your application to be accessed from wireless devices. The advantage of the rendering engine is that it relieves application developers from worrying about device-specific markup language types, image types, memory limitations, and screen-size limitations. It reduces and converts images and web content, optimizing them for each device supported in the device profile database. You can focus on your application's functionality, without the need to consider each mobile device's limitations. The Hornet rendering engine addresses the following issues:

- Translates HTML on the fly to WML, cHTML or xHTML based on the device capabilities
- Converts image types (JPG, PNG, GIF, etc.) on-the-fly
- Resizes images and applies color-depth adaptation
- Provides character set encoding on the fly
- Division of web content into sections for display on mobile devices

The use of the rendering engine will mainly be transparent to the application. However, certain restrictions apply for some specific features and active elements, such as JavaScript or flash elements. These limitations are addressed in detail later in this guide in order to enable you to make best use of the rendering engine.

2 Definitions

Term	Definition
HTML	HyperText Markup Language. A standard language to define web pages.
WML	Wireless Markup Language. A language to define simple text pages that can be accessed by a cell phone.
cHTML	Compact HTML for Small Information Appliances
xHTML	The Extensible HyperText Markup Language
Deck	The maximum displayable screen size of mobile device



3 Use of the Hornet Rendering Engine

The Hornet rendering engine is designed for accessing HTML web content from wireless devices, which have low bandwidth, limited memory and processing power, as well as small screen-size. The rendering engine is able to detect most properties from the wireless device, and generate suitable content given its constraints. The following properties will be considered during the rendering process:

- The markup language supported (WML, cHTML, or xHTML, etc)
- The image type supported (JPEG, GIF, PNG, WBMP)
- The color depth (32 bit down to 1 bit)
- The maximum displayable screen size
- The maximum memory size able to handle for each page
- The character encoding supported (well-tested UTF-8 for multiple languages)

All HTML pages will split into decks to small pages and character encoded appropriately with the following differences:

Device	Code	Image
WAP 1.0 legacy wireless phones	WML	Resized WBMP
WAP 1.0 wireless phones with color support	WML	Resized supported format
WAP 2.0 wireless phones with color support	xHTML	Resized supported format
I-mode wireless phones	cHTML	Resized supported format



3.1 Web Content Customization for Mobile Devices

3.1.1 Real World Example

The following example shows how the content of an HTML web page will display on your mobile device with both normal web browser and mobile browser side by side. Below you will see a standard web page that is split into a banner, a menu and a content section. On a mobile you likely don't want to see the banner. You also likely want to see the menu first and only if your click on a menu item you want to see the associated contents:

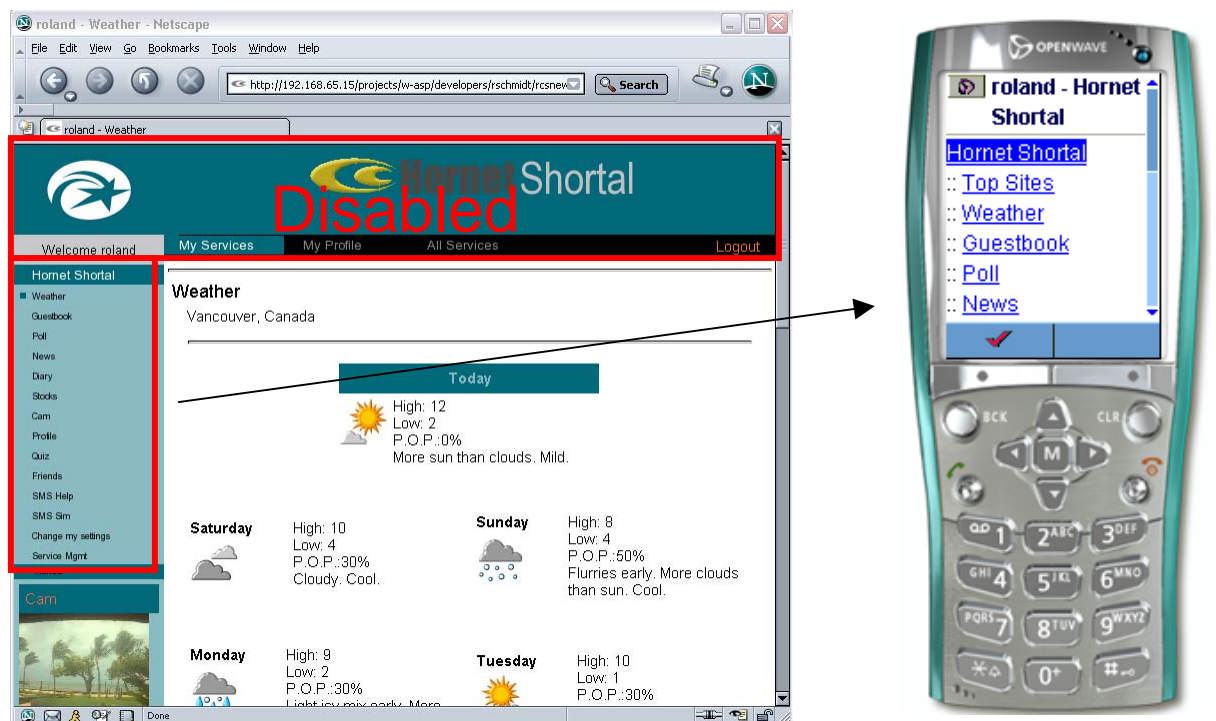


Figure - 1 Initial Display on the Mobile Device

When you use mobile device to access this page, the first display will ignore disabled banner and go to the menu directly subject to your definition.

As in our example, the menu has been defined as section. Suppose you select "weather" from your mobile device, the display will be changed to the outlined section on the following figure. Rendering engine will just render the content section identified by the "weather" menu item to show it on your mobile device.

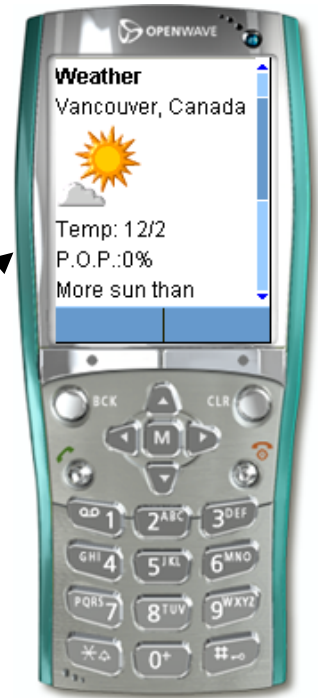
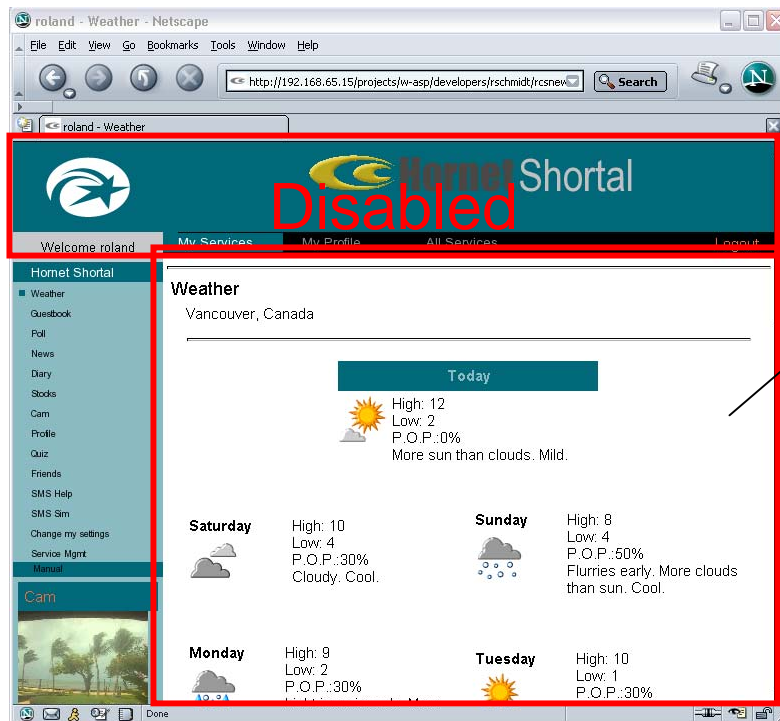


Figure - 2 "Weather" Section on the Mobile Device

3.1.2 How do we do it - Hornet Rendering Tag Attributes

Rendering Tag Attributes provided by Hornet allows an application developer to define pages to get optimal results for mobile devices. These include:

- disabling certain contents for rendering
- dividing the content on a web page
- defining the sections accessed in a specific sequence

Instead of showing all the data on one page (causing extended transmission delays and cost), pages with Rendering Tag Attributes are displayed one section at a time, as specified by the attributes (resulting in faster display and reduced transmission time/cost). Since web browsers ignore Hornet Rendering Tag Attributes, they only affect displays on mobile devices but not on the desktops.

The attributes defined below may be used to instruct the rendering engine to return HTML content in a specific order. They can be inserted in table related tags such as <table>, <tbody>, <td>, <tfoot>, <th>, <tr> and <thead>:



Tag Attributes	Description
<code><tag csi_name=""></code>	Defines a section name for display. This name will be used for referencing and switching between sections. Any tag using this attribute must have this tag set.
<code><tag csi_display="n"></code>	Defines the display sequence for this section of the page. For example, <code><tag csi_display=0></code> will be displayed before <code><tag csi_display=1></code> . The tag can also be used to disable rendering of particular sections to the wireless UI by omitting this tag for such section since only those sections framed by <code><tag csi_display=x></code> will be rendered if the tags exist.
<code><tag csi_target="n"></code>	Defines the section identified by the <code>csi_name</code> attribute (for non-unicode text) or <code>csi_display</code> attribute (for both Unicode and non-unicode text) to be displayed when a link to the page is selected. This allows redundant information such as the Table of Contents to be skipped when a page is accessed via a link.
<code><tag csi_break="true"></code>	Mobile phones support only one form per page served. Using this tag you can split a section of a table into independent pages.
<code><tag csi_no_rw></code>	When a page has been rendered, all the URL references defined in images, anchors, forms, meta, and script tags get rewritten to tell the browser to render the contents from the URL. In some cases, however, it may be necessary to bypass rendering a particular URL. This can be achieved by adding this attribute. CAUTION: Once a user clicks a link and leaves the rendering engine by this tag, all subsequent pages will not be rendered.

3.1.3 Sample Page with Tag Attributes and the Results

The following example shows how the content of an HTML web page is divided into sections:

<pre><tr csi_name="Home" csi_display="0" csi_target="1"> <td> Banner </td> <td> ...link... </td> </tr></pre>	
<pre><tr> <td csi_name="Menu" csi_display="1" csi_target="2"> Table of Contents ...link... ...link... </td></pre>	<pre><td csi_name="Main" csi_display="2" csi_target="2"> Main ...link... </td> </tr></pre>

Figure - 1: Sample Source

The table contains three display sections: 'Home', 'Menu' and 'Main'. Each section has the rendering attributes 'csi_name', 'csi_display', and 'csi_target'.



- The attribute 'csi_name' specifies the name of the section and is mandatory to activate the use of Rendering Tag Attributes.
- The attribute 'csi_display' specifies the presentation sequence. All other sections without this attribute within the page will be disabled when this tag is in use.
- The attribute 'csi_target' defines the section, identified by a csi_display attribute, to be displayed when the link to the page is selected.

Note that the Rendering Engine automatically adds navigation elements at the bottom of the mobile screen to allow easy navigation to section(s) not displayed. The labels for these navigation elements are defined by the 'csi_name' attribute.

Let us look what will happen when you use a mobile device to access this sample page:

- Section (0) will be displayed first because it has the highest display priority.
- In section (0), selecting the 'Menu' at the bottom or any links will bring you to section (1). Selecting the 'Main' button will bring you to section (2).
- In section (1), selecting the 'Main' at the bottom or any links will bring you to section (2), the 'Main' section of the page. This is because the 'csi_target' of 'Table of Contents' is '2' a.k.a 'Main'. Selecting the 'Home' at the bottom will bring you back to section (0).
- In section (2), selecting the 'Menu' at the bottom will bring you to section (1). Selecting any links or the 'Main' button will still reroute you to the same section of the original HTML page. Selecting 'Home' at the bottom will bring you back to section (0).



3.2 Reloading of Pages if visited by Mobile Device

The simplest way to mobilize your application or website is to use your HTML page as base and render it through the rendering engine. Once your cell phone uses a rendering engine all links are rewritten to continue to use the rendering engine - but how do we get the cell phone to go to the rendering engine the first place?

One commonly used way is to check during request processing whether

a) the requesting client is a cell phone and

whether the request comes directly from the phone and not already through the rendering engine.

If both conditions are true the request shall be rerouted through the rendering engine.

During the decision process you could also set a global variable that your website creation code is later checking to enable/disable some mobile specific functionality (for example: no use of JavaScript etc.)

The following code snippet shows an example how such decision and reload functionality is implemented in PHP. The functionality can easily be cloned if the pages are generated by other means, such as .NET, JSP or Cold Fusion.

The location of the rendering engine (in our example <http://wap.contec.ca/C/xC>) may be different depending on your particular installation:

```
<?php
global $is_mobile;

if (isset($_SERVER['HTTP_USER_AGENT']))
    $uAgt=$_SERVER['HTTP_USER_AGENT'];
else
    $uAgt='';
if (isset($_SERVER['HTTP_ACCEPT']))
    $acpt=$_SERVER['HTTP_ACCEPT'];
else
    $acpt='';
// Check if it is a mobilizable client
if (strpos($acpt,'wml')===false &&
    strpos($uAgt,'PDXGW')===false &&
    strpos($uAgt,'portalmmm')===false &&
    strpos($uAgt,'DoCoMo')===false &&
    strpos($uAgt,'Windows CE')===false &&
    strpos($uAgt,'PPC')===false &&
    strpos($uAgt,'Palm')===false &&
    strpos($uAgt,'BlackBerry')===false &&
    strpos($uAgt,'Symbian')===false &&
    strpos($uAgt,'dopod')===false &&
    !isset($_SERVER['HTTP_X_WAP_PROFILE']))
{ // do not use redirect
if(!isset($_SERVER['HTTP_CMS']))
    $is_mobile = false;
else
    $is_mobile = true;
}
```



```
else
{ // this is a mobile client
  if(isset($_SERVER['HTTP_CMS']))
    $is_mobile = true;
  else {
    Header('Location:

```

3.3 Limitations of the Rendering Engine

Due to limited capabilities of mobile devices, not all type of content can be transcoded to the devices. Here are a few suggestions for developing wireless enabled applications:

Recommendations for building a HTML page that can be transcoded properly:

Ensure that all HTML tags are nested properly and HTML syntax/form corresponds to W3C standards because mobile devices' trouble handling abilities are not as strong as normal web browser. Improperly nested or non-standard code may cause rendering engines to produce unexpected results. An online HTML validator at <http://validator.w3.org> is a good tool to verify your pages.

- To ensure that all mobile devices can view your application, keep your user interface as simple as possible. Wherever a <script> is used, it would help to provide a <noscript> equivalent. The same is true for <frame> and <noframe>.
- Since WML does not support multiple forms in a deck, you need to place a <csi_break> attribute within each form's container element (e.g. <td>) if your web pages contain multiple forms. This will help in properly splitting the forms into multiple decks in WML conversion (see section 3.1.2)
- If forms are needed, try to keep the contents as short as possible since most mobile devices do not have enough memory capacity to store and display a lot of large web-based forms. If a form cannot fit within a phone's memory constraints, the user may experience some unexpected results.
- Specify both width and height in the <table> and in pixel. The rendering engine uses those values to split tables and re-size images properly.
- Include a proper <meta http-equiv="Content-Type" content="..." /> nested in the <head> tag. It will help the rendering engine encode the HTML content **correctly**.
- Use csi tags to structure your page to allow the user easy navigation via the mobile device. For a description of available csi tags, see previous section.

Issues for all mobile browsers that should be avoided if possible:

- Do not use complex objects, such as Flash, PDF, Word, Excel, PowerPoint files etc. whenever possible. They cannot be rendered but are forwarded to the mobile device directly. These may cause bad performance or even memory crash.



- Avoid using <frame>; the rendering engine will only output one frameset each time. All links defined inside the <frameset> will be displayed.

Issues for WML mobile browser that should be avoided if possible:

- Do not use <script> to control dynamic content; the rendering engine will remove all code inside the <script> tag.
- Not all HTML tags are supported on mobile devices due to screen size, memory, and throughput limits. Some tags as follows need to be avoided if possible:

Tag	Reason
<link>	Not supported.
<object>	Not supported.
<script>	Not supported.
<applet>, <param>	Not supported
<map>, <area>	Due to limitations, the areas in the image map cannot be selected on the mobile devices. All links defined inside the <area> will be displayed after the image.

Issues that should be used with care:

- The following tags need to be used with care:

Tag	Reason
	The value will not be evaluated properly if it contains JavaScript in reference URL or full path URL.
<input onclick="..." type="button">	The value will not be evaluated properly if it contains JavaScript in reference URL or full path URL.
<form onsubmit="..." action="...">	The value will not be evaluated properly if it contains JavaScript in reference URL or full path URL.

3.4 Testing

When you follow the suggestion from this guide to design and develop your web content, user will immediately differentiate your websites from other sites for its mobile-friendly features. In general, mobile users want a convenient experience (fast access, appropriate, up-to-date but limited amount of information, limited picture size), which Hornet rendering engine will provide. However, the testing is a mandatory step that can ensure your mobile-friendly websites actually works. It is a good idea to test all the pages before putting them online. The testing should be better tried from more than one mobile device.

3.5 Feedback

Contec welcomes all your comments to continuously improve the Hornet rendering engine. All feedbacks can be mailed to info@contec.ca. Thanks!