



Genie Application Style Guide

(For Openwave™, Nokia™ Model 7110™, Model 6210/6250™, and Mitsubishi™ Trium™ WAP™ browsers)

Openwave Systems Inc.
800 Chesapeake Drive
Redwood City, CA 94063
<http://www.openwave.com>

Release 1.0, February 2001

LEGAL NOTICE

Copyright © 2000 –2001 Openwave Systems Inc. and Genie. All rights reserved.

Use other than for internal purposes, reproduction, modification or distribution without prior written authorisation by Genie is strictly prohibited.

Openwave, the Openwave logo and the “UP.” family of terms are trademarks of Openwave Systems, Inc. Nokia, Model 7110, and Models 6210/6250 are registered trademarks or trademarks of NOKIA Corporation and/or its affiliates.

Mitsubishi and Model Trium are registered trademarks of Mitsubishi Electric and/or affiliates.

“WAP Forum” and “W@P” and all trademarks, service marks, certification marks and logos based on these designations are marks of Wireless Application Protocol Forum Ltd. All other trademarks and registered trademarks are the properties of their respective owners.

Contents

1	Style Guide Overview	1
	Organisation of this Guide	1
	Why Specialise?	2
	Testing on SDKs	2
2	Usability Design Philosophies	5
	Creating Usable Applications	6
	Testing the Design	7
3	Navigation Guidelines	15
4	Menu Navigation	25
5	Making Phone Calls from the Browser	33
6	Using Multiple Selection Lists	37
7	Backward Navigation	39
8	Displaying Text	45
9	Data Entry Queries	51
10	Formatted Entry Fields	55
11	Forms	59
12	Icons and Images	65
13	Cache	67
14	Cookies and Subscriber ID	69

15 Labels and Links 71

A Identifying the Browser 73

B Differences between Browser Types in Same Class 77

Style Guide Overview

This document gives developers comprehensive guidelines for developing highly usable applications that run on Genie supported WAP browsers: the Openwave™ browser, the Mitsubishi™ Trium™ browser, and the Nokia™ Models 7110™, 6210™, and 6250™ (Nokia 7110, 6210, and 6250) browser. This guide considers various situations and possibilities, offers the most usable solution, and provides sample code.

Usability tests have shown that it is not possible to write a single usable “generic” version of an application that will run well on different types of mobile browsers. For this reason, this document is written to help developers create optimised applications to run on the Openwave, Mitsubishi, or Nokia browsers.

Organisation of this Guide

This guide begins by outlining some usability philosophies and then gives detailed guidelines on navigation, selection lists, text display, forms, alerts, and many more topics. Each section that gives guidelines looks at the topic from four points of view:

- **Shared feature set for creating WAP applications.** These recommendations apply to the Openwave, Nokia 7110/6210/6250 and Mitsubishi Trium browsers when developing usable applications. These guidelines address the shared feature set, that is, the features that are supported on all of the browsers. Because the browsers interpret the WAP standards differently when displaying data, this section outlines the approach that will work on all of the browsers.
- **Openwave usability.** These guidelines explain how to create the most usable applications for the Openwave browser. Use the Openwave guidelines in conjunction with the shared feature set guidelines. Code samples may employ the Openwave extensions to WML to provide advanced functionality. These extensions are available to applications running on Openwave handsets through the Genie gateway.
- **Nokia usability.** These guidelines explain how to create the most usable applications for the Nokia 7110 and Nokia 6210/6250 browser. Again, use the Nokia guidelines in conjunction the shared feature set guidelines. Although the Nokia examples are based on the Nokia 7110 browser, they apply as well to the Nokia 6210/6250 browser. The few exceptions are explicitly labelled as “Nokia 6210/6250 only”.

- **Mitsubishi Trium Usability.** These guidelines explain how to create the most usable application for the Mitsubishi Trium browser. Again, use the Mitsubishi guidelines in conjunction with the shared feature set guidelines.
- Appendix A provides information on how to distinguish between various browsers.

The information in this document derives from usability tests, knowledge of the capability of the WML, and testing applications on a variety of phone models and SDKs.

Why Specialise?

The WAP language specification for the Wireless Mark-up Language (WML) can be variously interpreted, and devices with browsers from different manufacturers exhibit differences in display and behaviour. Although the WAP Forum is in the process of refining the language specification, developers have an immediate need for how to build the best applications for browser phones on the market today. However, browsers interpret some of the same WML features differently. A main goal of this guide is to help application designers and developers understand and work around these differences.

A usable application is one that lets users achieve a goal with a minimum of keypresses and without incurring extensive charges. The most usable applications are written for a specific browser. “Generic” applications are developed to the “lowest common denominator”, a subset of WML that works on multiple browser phones. However, because this subset is quite small, “generic” applications are more difficult to use than applications customised for a specific browser.

This guide encourages developers to capitalise on the unique features of the Openwave, Nokia and Mitsubishi Trium browsers. The entire industry benefits from excellent usability on all devices.

Testing on SDKs

When developing to the features that work on the Openwave browser and the Nokia browsers, use the two SDKs:

- **The Openwave WAP-compatible SDK.** The UP.SDK 4.1 can be downloaded from <http://developer.openwave.com> <http://www.developer.phone.com/> at no expense.
- **The Nokia WAP Toolkit.** The Nokia WAP Toolkit can be found at <http://forum.nokia.com>. In addition to testing the applications on the SDKs, test the application on the handsets to ensure that it looks right and works correctly.

- **Mitsubishi Trium.** There is little developer documentation on the Mitsubishi Trium WAP browser, and no known full simulator or SDK product available at this time. The device has not been through interoperability tests. The guidance for this browser is therefore based on observation of the behaviour of phones containing the Mitsubishi browser (version 1.1.a) based on how the device renders the content of a sample set of applications. Indeed two different behaviours have been noticed with the two different browsers both returning the same user agent string. The advice found herein will produce applications which are usable on both observed devices. The differences between those devices, and how they may be differentiated in software is provided in Appendix B for those developers who may wish to make their applications more specialised. There may be further versions of the browser reporting the same user agent with different behaviours, thus it is important to test all available versions of the handsets.

Usability Design Philosophies

2

Keep in mind a few design philosophies when building an application for a WAP browser phone. The user's experience with an application may determine whether or how often the user revisits the application.

- **Usability is critical.**

Device constraints limit both navigation and the amount of content that a handheld device can display; further, data entry may be difficult. Take extra care to make the application usable in this constrained environment, or users will not use it.

- **Most devices are phones first.**

Most devices on the market today have added the browser as an afterthought. Neither the hardware nor the user interface reflects reasoned planning for the browser from the inception. Some features may be more difficult to use on one phone than another.

- **Mobile handsets will be used for information retrieval, not browsing.**

Users of WAP browsers will not "browse the Web," as they do with a PC, because the amount and nature of content is scaled down for the handheld device. Phone applications will most commonly be used for quick information retrieval, not for general browsing and research. Phone users tend to be less technical and in more of a hurry to get to the data they seek.

- **A phone is not a PC.**

The phone has features that a PC does not, such as the ability to integrate voice, data, and alert features. Design applications that address user interaction models, screen size, and screen context. When designing the UI, keep in mind that there is greater variability among phones than PCs.

- **Airtime costs money.**

Most networks charge the user by the minute for browsing on the phone. Assume that users will avoid expensive browsing sessions.

- **Minimise or avoid text entry.**

Entering text on the phone keypad is tedious. Try to use alternative methods, such as remembering previously entered text or data, personalization, and selection lists.

- **Most users will avoid complex applications.**

Applications must be well organised with a shallow menu structure so that the user can get to the value quickly.

Creating Usable Applications

When developing applications, these are the most important factors to consider: who the user is, what problems the user is trying to solve, and how to solve them most efficiently. Here are some key principles for creating usable applications:

- **Specialise your application for the specific browsers.**

To create a more usable application, determine the type of browser in the code. With that information, develop customised versions that increase usability by taking advantage of unique browser features.

- **Know your customer.**

Users turn to an application to solve a problem and, in some environments, to communicate or be entertained. For instance, the user's goal might be to purchase an item or to upload and download information while in the field. Build the application to help the user accomplish that goal. If the user's goal is to find a stock quote, display the quote right away. Use the quote display screen as the entry point to any other information the user might want.

- **Get to the value quickly.**

Deeply embedded information can cause the user to forget the goal and become frustrated. This frustration will cause the user to avoid the application in the future. Provide commonly used options quickly rather than requiring users to navigate deep menus.

- **Limit the application to only necessary functionality.**

Remember that the browser does not have the display and navigation capabilities of a PC. In addition, while browsing on the handheld device, the user is looking to find or submit information in the shortest possible time. Scale down the application to meet only the goals of the user, and do not include extras. Provide access to the most commonly used features through menu choices, links, or options.

- **Make the application easy to navigate.**

Minimise the number of steps it takes to access information. Eliminate or combine cards if this can be done without losing important information, choices, or content. Create multiple paths to access information, if possible. For example, if the application provides weather content, allow the user to search by postal code or city. In this way, the user has the choice of entering a short code rather than long strings, which are hard to type on the phone.

- **Make the application consistent.**

Consistent applications are “intuitive” for the user. Make the texts descriptive and easy to follow. Labels should explain the actions they cause. Order lists logically, so that items and links are easy to find. Although images and icons provide added information to help make pertinent information stand out, be careful not to overuse them.

- **Avoid text entry.**

Avoid queries that force the user to enter alphanumeric text. Use menus or partial text searches to avoid or minimise text entry.

- **Personalise the service according to the user.**

Allow an application to retain user information to autofill personal fields. For example, store the login and/or password, billing address, or other information in a cookie or on a server where the application resides. The user can be determined by the subscriber ID.

- **Anticipate situations in which users are likely to make errors.**

Make sure the application does not allow the user to continue a task unless certain requirements have been met. For example, if the user is entering the date, the application should check that the date is 8 digits: 2 digits for the day, 2 digits for the month, and 4 digits for the year.

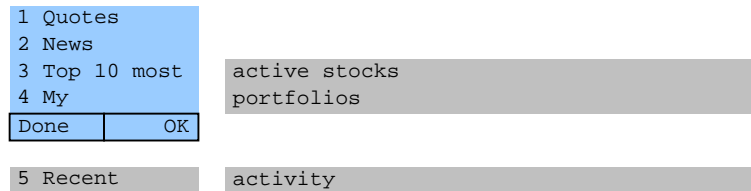
Testing the Design

Once the application is designed, use display templates and navigation flow diagrams to show how the application will behave on each type of browser. Use templates to view the layouts until you are satisfied with the text, wrapping, and overall look and feel. For this guide, templates restricting the lengths of the text were used to demonstrate menus, multiple selection lists, non-wrapping text (Times Square text scrolling), the viewing of text and links, and entries. Examine every layout and display for consistency. Unnecessary differences (for instance the use of scrolling text in one screen and wrapping text in another) can confuse the user.

The sample application diagrams in Appendix B are meant to demonstrate navigation only, not other behavior. In contrast, the templates used throughout the rest of the guide indicate other desired application behaviors (wrapping texts, softkey labels, and so on).

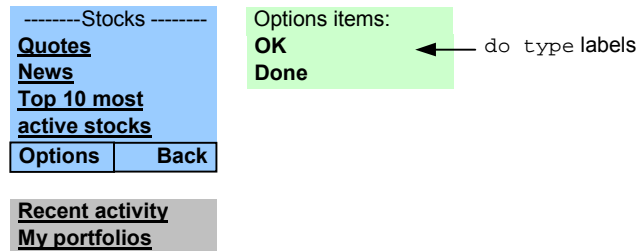
Browser Templates

Openwave Browser Template



The Openwave Browser typically has two programmable softkeys at the bottom; in this example, OK and Done. For these examples, the `<do type="accept">` label appears on the right softkey and the `<do type="options">` label on the left softkey. Text that scrolls horizontally across the screen (Times Square scrolling) appears to the right of the screen, in grey. Below the screen, the template shows additional menu items and text; the user must scroll down to these. Each phone has a fixed key for Back or Back/Clear navigation, not shown.

Nokia 7110 Browser Template

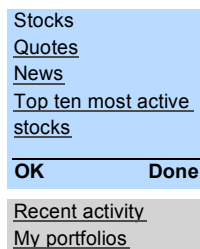


The Nokia 7110 browser has two softkeys with fixed labels: Options and Back. The application `<do>` labels are listed under and accessed from the Options softkey. In these examples, the `<do type="accept">` and `<do type="options">` labels appear to the right of the screen under Options items and can be accessed from the Options softkey.

The Nokia 7110 roller key has various functions depending on the selected item on the display:

- If the selected item is an `<input>` element, activate the input query;
- If the selected item is a `<select>` element, activate the select list;
- If the selected item is an `<a>` or `<anchor>`, activate the link and access the defined URL;
- otherwise, display the Options items (the same menu listed under the Options softkey).

Mitsubishi Trium Browser Template



The Trium handset has two softkeys and a four-way scroll key (supporting the up and down functionality, as well as moving forward and backwards). The Back functionality is available from a press on the left scroll key. The left softkey is a programmable key and will display a `<do>` element if only one is defined for the card (which is not a `prev`). When more than one element is defined, the softkey displays the label `Card` and will display a menu of the defined `<do>` elements in the order of their definition. The right softkey behaves depending on the context, but typically displays the `<prev>` element. When links, such as `<a>` or `<anchor>`, `<input>` and `<select>` elements are highlighted, these will typically replace one of the two softkeys on a context sensitive basis. On one type of device, they will appear on the right hand key, on the other, they will appear on the left hand key ONLY when no conflicting `<do>` element is defined (although they could be accessed through the four-way scroll key). Genie's experience has shown that users (particularly beginners) do not understand the forward and backward behavior of the scroll key, and therefore Genie recommends that its use is not relied upon.

Browser Properties for Phones

Although WAP-compatible phones on the market have a variety of screen sizes, number of keys, and key functionality, design the application to display well on a small screen. Applications designed for small screens look good on larger screens, but not necessarily the other way around.

- **Up to four lines of text or selection items may be visible.**

Some phones have as few as two lines; some have as many as eight lines.

- **Approximately 15 characters can be displayed on one line.**

This number may vary, since many phones support variable-width fonts.

- **All phones have menu up/down navigation.**

- **Right/left navigation may be available.**

- **The browser's home deck may not be easy to access or find.**

- **Some phones support smart-entry methods (for instance, T9 or smart mode).**

- **Most phones can display graphics or images.**

- **Not all phones have a Send/Talk key.**

- **All phones support uppercase and lowercase fonts.**

- **All phones support links, but phones may display them differently.**

- **Most phones do not have separate Back and Clear buttons.**

When queries for entries are presented, the user may need to delete all entered characters in order to return to a previous screen.

Table 2-1 summarises the properties of the Openwave, Nokia 7110 and Mitsubishi Trium WAP browser phones (in the table, "phone" means a WAP browser phone).

NOTE There is some variability among Openwave browser phones; however, this should not affect the user experience.

Table 2-1. Summary of browser phone properties

Property	Openwave	Nokia 7110	Mitsubishi Trium
Number of characters per line	15 characters (mostly variable-width fonts)	18 characters (variable-width fonts)	Variable-width fonts. User can change font size.
Lines of display	2 to 8 lines	4 lines	5 lines (depends on font size)
Image/graphic support	Supports images and graphics	Supports images and graphics	Supports images and graphics. User may disable these.
Link support	Links display as: [link], <link>, or <u>link</u>	Links display as: <u>link</u> Nokia 6210/6250 only: Can be selected by pressing Send key	Links display as: <u>link</u>
Scroll keys	Scroll keys for Up/Down, some support Left/Right scrolling, others do so automatically	Roller key for Up/Down 6210/6250 only: Scroll keys for Up/Down	Four-way scroll key for Up/Down (also supports forward and backward navigation)
<do> labels	Label is associated with softkeys on the phone	Labels appear under Options softkey	Label displays on or under left softkey; see Mitsubishi Trium Browser Properties
Menu navigation	Develop as a <select> element or statement	Develop as a list of links	Develop as a list of links

Table 2-1. Summary of browser phone properties (*continued*)

Property	Openwave	Nokia 7110	Mitsubishi Trium
Back key for navigation	Dedicated Back key available except in entry queries	Programmable Back key is displayed on right softkey except in entry queries	Dedicated back key on left scroll key. Programmable Back key is displayed on right softkey
Clear/Back key in entry queries	Shared. User must delete data in a query to retain Back properties.	Shared. User does not necessarily need to delete data to return to previous card.	Shared. User does not necessarily need to delete data to return to previous card.
Entry queries and multiple selection lists	Displayed as a browser card with one programmable softkey	Displayed as a local application that users must select to access the local application	Displayed as a local application that users must select to access the local application

Openwave Browser Properties

The following properties are unique to the Openwave browser. Capitalise on these to enhance usability. Use the Openwave UP.SDK 4.1 or WAP browser phone to test applications using these features.

- **A label for the highest priority action is viewable and accessible from all cards.**

The `<do type="accept">` label is displayed on the screen and accessed by pressing the primary softkey (the right softkey in the template).

- **A label for the secondary actions is accessible from a second softkey.**

The `<do type="options">` label is mapped to the second softkey (the left softkey in the template) and may require one or more keystrokes to select (depending on the number of `<do type="options">`).

- **Each phone has a fixed key mapped to backward navigation.**

Back functionality, `<do type="prev">`, is always available from a fixed key on the keypad. The default is the last card in the history if no other location is specified. This key may be shared with a Clear key in entry queries.

- **Phones support the ability to select an item from a numbered list.**

Numbers corresponding to `<option>` items precede each menu item.

Nokia Browser Properties

- **Nokia 7110 only: The roller key can be pressed to activate the selected item.**

The selected item (action, anchor, `<option>`, `<select>`, or `<input>` element) can be activated from the roller key and is also listed as the first item when the user presses the Options softkey, “Nokia 7110 Browser Template” on page 8. No label is displayed for the roller key action. If no item on the display is selected, pressing the roller key will display the menu listed under the Options softkey.

- **Nokia 6210/6250 only: The user can press Send key to activate the currently selected anchor, `<option>` element, `<input>` element, or `<select>` element.**

Do not rely on the user discovering this behavior since no label is displayed for the send key action. If no item on the display is selected, pressing the Send key will display the menu listed under the Options softkey.

- **Entry and multiple selection screens are self-contained functions.**

The user cannot access `<do>` element, `<option>` element, or anchors from an input query or multiple selection screen. The phone controls the display of these functions, which are accessible on the card from which the item was selected.

- **The second softkey is reserved for Back.**

The phone displays the `<prev/>` task on the right softkey only if the action defined is `<prev/>`. There is no need to provide a label; the browser will assign the label Back. A defined label will not render on the softkey.

- **The screen displays approximately 18 characters on one line.**

Because the phone supports variable-width fonts, the number of characters per line varies. Check the text on a phone or Nokia WAP Toolkit.

Mitsubishi Trium Browser Properties

- **The left softkey supports the `<do>` elements.**

The left softkey is a programmable key and displays the label for the `<do type="accept">` element when there are no more than one `<do type="accept">` or `<do type="options">` elements are defined. When more than one element is defined, the softkey displays the label Card and will display a menu of the defined `<do>` elements in the order of their definition. See the rule below when a `< prev>` element is defined.

- **The right softkey label supports the `<prev>` element and for some phones may be context sensitive.**

The right softkey supports the following function depending if an item is selected or what type of element is defined:

- If there is no item selected, the softkey returns the user to the previous card in the history.
- For some phones, when a `<prev>` element is defined with a label, the phone will display the defined label. When no label is defined, `prev` or `Back` will display. Backward navigation will replicate that of the left scroll key.
- For some phones, if the item selected is a `<input>` or `<select>` element, the softkey displays the defined label and accesses a card with the `<input>` or `<select>` element. For some phones, if there is a `<do>` element define with an action other than `<prev>`, there is no softkey prompt for the input or select element. The user can the access the `<input>` or `<select>` element from the right scroll key.
- For some phones, if the item selected is a link, the softkey displays the defined label and accesses the URL.

- **The phone has a fixed key mapped to backward navigation.**

- **`<input>` and `<select>` elements.**

Activating a `<select>` or `<input>` element takes the user to a separate card to either select an item or enter data.

- **Tables are supported.**

- **Images and WML Scripts may be turned off by the user.**

The browser has a setting that allows users to enable or disable image and WML Script support. For images, the browser displays the alt text defined. When WML Scripts are turned off, an error message displays causing user confusion; therefore, avoid using WML Scripts.

- **Timers are supported.**

Timers on cards will be re-initialised and activated each time the card is displayed, even if that particular counter had already triggered.

Navigation Guidelines

To navigate Wireless Mark-up Language (WML) content, the user must move through and between cards in one or more decks. The cards can contain many different types of elements, including selection lists (items in a list), displayed information (such as an email message), input fields, and multiple selection lists. To make applications run on multiple browsers, follow a number of general rules.

Shared Feature Set: Navigation Guidelines

- **Do not assign more than one action of `<do type="accept">` in any card.**
- **Map the most commonly chosen action or most intuitive task to `<do type="accept">`.**
- **If there is to be more than one `<do>` task, ensure that the most common one is defined first in the card.**
- **Create an intuitive label for every `<do>` task.**

This label will appear on the softkeys for the Openwave and Mitsubishi Trium browsers and under the Options softkey for the Nokia browser. Note that when a label is assigned to the `prev` task, the Nokia 7110 browser displays a Back label on the right. Always define a label of Back or Done for the `prev` task for the Mitsubishi Trium browser if the task is defined separately.

- **Capitalise only the first letter of labels of `<do>` elements except in words like OK.**
Consistent style throughout all applications enhances usability.
- **Define a backward navigation action for each card.**

Each card should have a `<do type="prev">`.

Example 3-1

```
<do type="prev" label="Back">
  <prev/>
</do>
```

- **Never define a `prev` as having no action.**

Do not bind an action of type `<noop/>` to a task of `<do type="prev">`. This forces the user to return to the home deck, which is not always intuitive and may make users follow a long path to return to the application. Instead, bind the `prev` task to an intuitive place within the application, a starting point within the application or to the home deck when that makes sense.

- **Provide a confirmation card (delete shield) to prevent data loss.**

Do not allow users to back out of an application inadvertently when they have already entered data in an entry query. Create a card asking users to confirm that they want to quit. This spares users the tedious task of re-entering data.

Example 3-2

Openwave Browser

Name (first & last): John Doe	
Alpha	Next

User presses the Back key

Are you sure you want to cancel your order?	
Yes	No

Delete shield

Nokia 7110 Browser

-----Order----- Name (first & last): [John Doe]	
Options	Back

Options items:
Edit
Next

----Confirm order---- Are you sure you want to cancel your order?	
Options	Back

Options items:
No
Yes

Mitsubishi Trium Browser

Order Name (first & last): John Doe	
OK	

User presses the Back key

Are you sure you want to cancel your order?	
Yes	No

Delete shield
[City &

```
<onevent type="onenterbackward">
  <go href="#confirm"> <!-- This goes to a card that asks users
    if they are sure they want to delete all the data they've entered -->
</onevent>
```

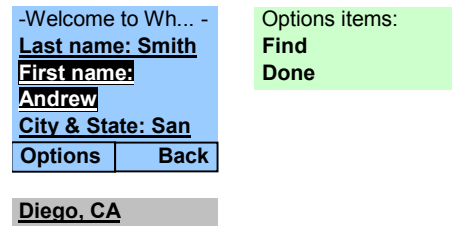
In Example 3-2, the code should be placed on the card that spawns the data entry form. If the user backs out after having entered the name, the confirmation card acts as a delete shield.

- **All cards should have a title attribute of 18 characters or fewer.**

Longer titles may be truncated, obscuring meaning. Test the title to make sure it fits. Not all phones display the title.

Example 3-3

Nokia 7110 Browser



```
<card id="welcome" title="Welcome to White Pages">
```

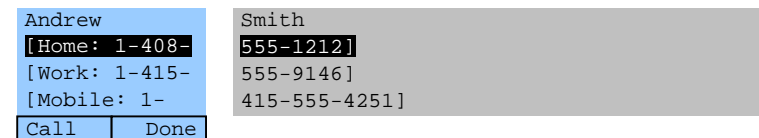
In Example 3-3, the last eight characters of the title are truncated. Test the text length on the Nokia SDK to ensure that the entire title appears on the screen.

- **Whenever a phone number is displayed, make it possible for the user to call it.**

Assign the action href="wtai://wp/mc;<phone number>" to create calls, and assign the label Call. This will not work on Nokia handsets, but users can use the Use Number functionality instead.

Example 3-4

Openwave Browser



```
<a href="wtai://wp/mc;14085551212" title="Call">1-408-555-1212</a>
```

In Example 3-4, the user can call 1-408-555-1212 by pressing the Call softkey.

- **When possible, keep the order of menu items or forms the same.**

Users become familiar with the order and select items without paying much attention.

- **Do not rely on font properties to convey added information.**

Many phones do not support various font properties such as bold, underline, and italic.

Openwave Navigation Guidelines

- **Always define an action for `<do type="accept">`.**

If no action is defined, a task of `<do type="prev">` may be automatically bound to the `<do type="accept">` key with the label OK or Back, which may not be what the developer is intending. If Back is a more appropriate label than OK, use this:

Example 3-5

```
<do type="accept" label="Back">
  <prev/>
</do>
```

- **Limit the length of labels.**

Define `<do>` element, `<option>` element, and anchors labels using five characters or fewer: the screen size on many phones is limited. Longer labels may be truncated and lose meaning.

Example 3-6

```
<do type="accept" label="Find">
  <go href="find.wml"/>
</do>
```

In Example 3-6, the label Find is under five characters.

- **Limit the number of softkey actions to two or fewer, when possible.**

Since most phones do not have more than two softkeys, this limit allows the user to view both softkeys. This simplifies tasks because the softkey labels are accessible with one keypress. When more than two elements are defined, the first element is bound to the primary softkey, and all other options and elements are accessible from the secondary softkey.

Example 3-7**Openwave Browser**

1 U2: Best of	1980 - 1990
2 Actung Baby	
3 Joshua Tree	
4 War	
Menu	View

Options under Menu are:
Buy
Alert
Reset

```
<do type="options" label="Buy">
  <go href="buy.wml" />
</do>
<do type="options" label="Alert">
  <go href="alert.wml"/>
</do>
<do type="options" label="Reset">
  <refresh>
    <setvar name="album" value="" />
  </refresh>
</do>
```

In Example 3-7, more than one option is bound to the `<do type="options">` task. For this reason, the browser renders the label Menu on the secondary softkey. The options under the Menu softkey are Buy, Alert, and Reset. The phone will label the softkey Menu. Although the provided UI is generally understandable and efficient, a developer may wish to exercise more control and build a card for the menu themselves. This could provide more descriptive action items and enable better use of icons and key accelerators. In this case, when more than one option is required, build a card with the appropriate actions and bind this card to the secondary softkey key.

- **Include a descriptive header of 15 characters or fewer.**

This header informs the user of the context if the title is not displayed. On some Openwave browsers, the title may be displayed as well.

Example 3-8**Openwave Browser**

Stock service	
1 Symbol	
2 Company	
3 Portfolio	
Menu	Edit

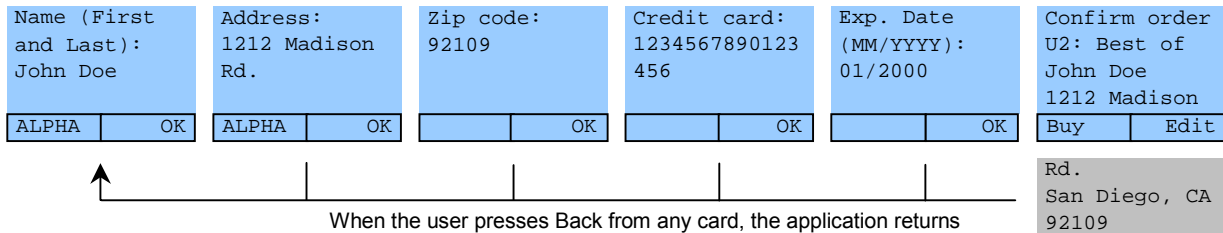
In Example 3-8, the header “Stock service” orients the user to the context.

- Use activities or carefully designed navigation to direct the Back key to the most intuitive page.

Activities best create a start point that allows the application to return to a specified card when user presses the Back key. The intermediate cards (those accessed before the return to the specified card) are removed from the history.

Example 3-9

Openwave Browser



When the user presses Back from any card, the application returns to the first card, not the previous one. In this way, the user can navigate forward and need not delete already entered information.

```
<card id="buy" title="order cd">
  <do type="accept" label="Next">
    <spawn href="#add1">
      <setvar name="album" value="$album"/>
      <setvar name="name" value="$name"/>
      <setvar name="street" value="$street"/>
      <setvar name="zip" value="$zip"/>
      <setvar name="cc" value="$cc"/>
      <setvar name="exp" value="$exp"/>
      <catch name="bail">
        <receive name="street"/>
        <receive name="zip"/>
        <receive name="cc"/>
        <receive name="exp"/>
      </catch>
    </spawn>
  </do>
  <do type="prev">
    <go href="#cnclcrd"/>
  </do>
<p>
  Name (First and Last):
  <input name="name" title="Full Name"/>
</p>
</card>
```

```
<card id="add1" title="order cd">
  <do type="accept" label="Next">
    <go href="#add2"/>
  </do>
  <do type="prev">
    <throw name="bail"/>
  </do>
<p>
  Street Address
```



```
        <input name="street" title="Address:"/>
    </p>
</card>

<card id="add2" title="order cd">
    <do type="accept" label="Next">
        <go href="#cc"/>
    </do>
    <do type="prev">
        <throw name="bail">
            <send value="$street"/>
        </throw>
    </do>
    <p>
        Zip Code
        <input name="zip" title="Zip code:" format="NNNNN"/>
    </p>
</card>

<card id="cc" title="order cd">
    <do type="accept" label="Next">
        <go href="#exp"/>
    </do>
    <do type="prev">
        <throw name="bail">
            <send value="$street"/>
            <send value="$zip"/>
        </throw>
    </do>
    <p>
        Credit Card:
        <input name="cc" title="CC #" format="16N"/>
    </p>
</card>

<card id="exp" title="order cd">
    <do type="accept" label="Next">
        <go href="#confirm"/>
    </do>
    <do type="prev">
        <throw name="bail">
            <send value="$street"/>
            <send value="$zip"/>
            <send value="$cc"/>
        </throw>
    </do>
    <p>
        Exp date (MM/YYYY):
        <input name="exp" title="exp date mmyyyy"
            format="NN\ /\ 2\ 0NN"/>
    </p>
</card>
```

In Example 3-9 the goal is to allow the user to navigate backward through the data input wizard without losing data that was difficult to enter (since many devices map the clear and back functions to the same key) and without corrupting the history stack. This is accomplished by starting out with the `spawn` action, which enables the use of `<throw>` for backward navigation. In the `<throw>` statement for each card, the data that has already been entered is sent back (so it is not lost) and caught in the card where the user entered the name. This way, when navigating forward through the wizard, the user need not re-enter data but can modify or remove it as necessary.

A similar effect (although not quite so efficient) may be achieved with the use of the `onenterbackward` construction, although if data is no longer in the cache this will involve further server hits.

A card containing:

```
<onevent type="onenterbackward">
  <prev/>
</onevent>
```

Will immediately call the card that originally called it, without erasing any data.

Nokia Navigation Guidelines

■ Provide an action or link on every card.

Every card should define either a link or an action bound to a task of `<do type="accept">`, `<do type="options">`, or `<do type="prev">`. Generally, links should be used as a mechanism for forward navigation.

Example 3-10

Nokia 7110 Browser

```
----- Contacts -----
Last name: Smith
First name:
Andrew
City & State: San
Options Back
```

```
Options items:
Edit
Find
```

```
Diego, CA
Zip:
Phone:
Email:
```

```
<do type="accept" label="Edit">
  <go href="edit.wml"/>
</do>
<do type="options" label="Find">
  <go href="find.wml"/>
</do>
```

In Example 3-10, the `<do type="accept">` task calls `edit.wml`, and the `<do type="options">` task calls `find.wml`. On the Nokia 7110 browser, Edit and Find are accessible from the Options softkey.

- **Use up to 18 characters on option labels.**

Since `<do>` labels are items under the Options softkey, labels can be approximately 18 characters long. Limit labels to one or two words.

Example 3-11

Nokia 7110 Browser

-----Contacts-----		Options items: New Contact Done
Last name:- [Smith]		
First name: [Andrew]		
Options	Back	
City & State: [San Diego, CA]		

```
<do type="accept" label="New Contact">
  <go href="newctct.wml" />
</do>
```

In Example 3-11 the menu item “New Contact” can replace New in applications searching a database of information because the label is displayed under the Options softkey.

- **Never define an action of `go` or `noop` to the `prev` task.**

- The Nokia browser will not activate the right softkey. The action will be accessible only under the Options softkey. To control how the application behaves when the `<prev>` action is invoked, use a “shadow” card that contains an `<onevent type="onenterbackward">` element.

Example 3-12

```
<wml>
  <template>
    <do type="prev" label="Back">
      <prev/>
    </do>
  </template>
  <card id="start" onenterforward="#start2"
    onenterbackward="backtome.wml" >
  </card>
  <card id="start2" title="The Start">
  ...
```

Example 3-12 works by defining a dummy card (`id="start"`) that immediately transfers control to the real card (`id="start2"`) and puts itself on the history stack. Entering the card from a backward direction triggers the `onenterbackward`, thus loading `backtome.wml`.

- **It is not necessary to define a label for a `<do type="prev">`.**

The Nokia 7110 browser assigns the label `Back` to the right softkey and ignores the defined label.

Mitsubishi Trium Navigation Guidelines

- **Provide backward navigation for each card.**

Provide an intuitive label, for example Done or Back, to allow the user to use the right softkey for backward navigation. The browser will automatically provide a label of Back assigned to an action of `<prev/>` - if the term "Back" is not suitable, it may be overridden such

Example 3-13

```
<do type="prev" label="Done">
  <prev/>
</do>
```

Example 3-13 provides an alternative labelling for the right hand key.

- **Do not rely on being able to override the default backward navigation**

Trium browsers vary in terms of their handling of the `<do type="prev">` construction. One type will only change the right hand key behaviour if the action is still `<prev/>`; the other type allows `<noop/>` and `<go/>` constructions to be used also. For safety, handle alternate back navigation using a mechanism similar to Example 3-12 above.

- **Do not define a `<do type="accept">` or `<do type="options">` task when a `<select>` element, `<input>` element, `<a>` or `<anchor>` is used.**

A `<do type="accept">` or `<do type="options">` task on some versions of the browser, this will cause the softkey to display the Card label preventing easy navigation to the next card or to the input and select items. Instead, provide a link allowing the user to navigate to the next card.

- **Limit the number of softkey actions to one.**

The browser displays the defined label on the left softkey. This simplifies tasks because the softkey label is accessible with one keypress. When more than one task is defined, the left softkey label may become Card and displays the defined labels. In this case, build a card with the desired tasks and label the softkey Menu. This provides room to display more descriptive labels for each action and enables the use of key accelerators, on some handsets.

- **Do not underline text (`<u>` mark-up tag), because the user may think that the item is a link.**

Not all Trium browsers will do underlining if the tag is used, but if it does, there is clear possibility for confusion.

- **Avoid using WML Scripts.**

Since the user can disable WML Scripts, do not rely on them being activated. You can check to see if the user has disabled scripts by checking the `HTTP_ACCEPT` header – however the user could still turn off scripts while a script is in the cache of the phone.

Menu Navigation

Navigating through menus primarily consists of selecting items or links that, when selected, display a new card or deck or perform some action. Menus can be used for:

- Presenting a list of data (for instance, a list of email messages)
- Navigation (for instance, choices within a financial application)
- Performing an action (for instance, deleting an email message)
- Selecting an option (for instance, picking the day of the week for a scheduled event)
- Changing an option (for instance, allowing the user to change a preference or setting)

Shared Feature Set: Menu Guidelines

- **Sort items contextually.**

Items or links should be sorted logically as content dictates; this might be by type, by date, alphabetically, and so on. If there is no logical order, sort by priority, that is, put the item most likely to be chosen first.

- **Do not put more than 9 items on a single card.**

Limit the amount of scrolling needed on a given card and allow for key shortcuts.

- **Create a More link or `<select>` element as the 9th item if there are more than 9 items in a menu.**

Clicking More should display the next card of menu items.

- **Consider using numbers to help the user remember the application.**

Even if the handset does not provide for key accelerators numbers help a user find their way around an application and remember where to find the information that is important to them. Openwave browsers (when using `<select>` - and automatic numbering) and some Mitsubishi Trium browsers (when using `<a>` or `<anchor>`) provide key accelerators, but numbering is potentially useful even if this is not the case.

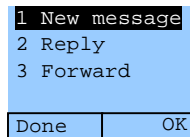
Openwave Menu Guidelines

- **Use the <select> element to get numbers, icons, and items for a menu.**

These provide quick access to an item by including numbers in a list. Instead of using anchors, use a <select> element for each card. In this case, each item in the list becomes an <option onpick=URL> element rather than an anchor.

Example 4-1

Openwave Browser



```
<select>
  <option title="New" onpick="compose.cgi">New Message</option>
  <option title="Reply" onpick="reply.cgi">Reply</option>
  <option title="Fwd" onpick="forward.cgi">Forward</option>
</select>
```

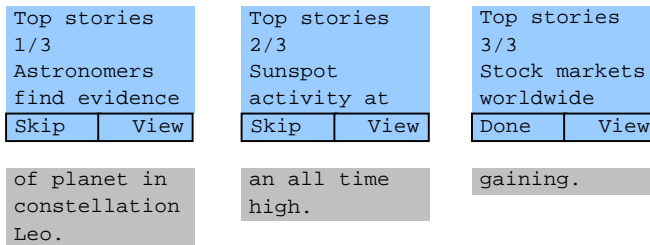
In Example 4-1, menu items can be selected by pressing a number key.

- **Wrap text that takes up multiple lines or multiple display windows.**

For a series of items that occupy multiple lines, allow the lines to wrap and use a link to display the next item. For example, in a list of news headlines, display each headline on a card of its own. Use an anchor with the label Skip at the end so that the user can navigate to the next headline. Map the <do type="accept"> task to a View label so that the user can view the news item. While the user is viewing the headline, map the <do type="accept"> task to a Skip label, so that the user can skip to the next headline.

Example 4-2

Openwave Browser



```
<card id="story1">
  <do type="accept" label="Skip">
    <go href="#story2">
  </do>
  <do type="options" label="View">
    <go href="story1full.wml"/>
  </do>
  <p>
    Top Stories 1/3<br/>
    Astronomers find evidence of planet in constellation Leo.
    <a href="story1full.wml" label="View">View</a>
    <a href="#story2" label="Skip">skip</a>
  </p>
</card>

<card id="story2">
  <do type="accept" label="Skip">
    <go href="#story3">
  </do>
  <do type="accept" label="View">
    <go href="story2full.wml"/>
  </do>
  <p>
    Top Stories 2/3<br/>
    Sunspot activity at an all time high.
    <a href="story2full.wml" label="View">View</a>
    <a href="#story3" label="Skip">Skip</a>
  </p>
</card>

<card id="story1">
  <do type="accept" label="Done">
    <go href="newshome.wml">
  </do>
  <do type="options" label="View">
    <go href="story3full.wml"/>
  </do>
  <p>
    Top Stories 3/3<br/>
    Stock markets worldwide gaining.
    <a href="story3full.wml" label="View">View</a>
    <a href="newshome.wml" label="Done">Done</a>
  </p>
</card>
```

In Example 4-2, a list of items with long text is broken into a card with the header information. The user can skip to the next header by pressing the Skip softkey or the Skip link at the end of the text.

- **An anchor should have a descriptive label of five characters or fewer.**

The label for links leading to the defined URL should be five characters or fewer. It is rendered as the softkey label.

- **Allow users to perform multiple actions on a selected item.**

An item does not need to be selected, only highlighted, before an action can be performed on the item. For example, an email application may display a list of email subjects as a menu. On Openwave browsers, the user should be able to view an item by highlighting the message and pressing the primary softkey, View. However, the user should also be able to reply to, delete, or file the highlighted item. This allows the user to delete the message while viewing the header information, instead of having to retrieve and read the message before deleting.

Example 4-3

```
<wml>
  <head>
    <meta name="vnd.up.markable" content="true" forua="true"/>
  </head>
  <card>
    <do type="accept" label="View">
      <spawn href="?NS=view&U=${U:escape}&MB=${MB:escape}>
        <setvar name="U" value="${U:noesc}"/>
        <setvar name="MB" value="${MB:noesc}"/>
        <setvar name="HO" value="0"/>
        <setvar name="MIB" value="65540"/>
        <catch name="redisplay" onthrow="${NEXT_DEST:unescape}">
          <receive name="NEXT_DEST"/>
          <receive name="MB"/>
          <receive name="II"/>
        </catch>
        <catch name="return"/>
      </spawn>
    </do>
    <do type="options" label="Menu">
      <spawn href="?NS=hdrMenu&MB=${MB:escape}#${U:escape}">
        <setvar name="U" value="${U:noesc}"/>
        <setvar name="MB" value="${MB:noesc}"/>
        <setvar name="HO" value="0"/>
        <setvar name="ZZ" value="1"/>
        <setvar name="MIB" value="65540"/>
        <catch name="redisplay" onthrow="${NEXT_DEST:unescape}">
          <receive name="NEXT_DEST"/><receive name="MB"/>
          <receive name="II"/>
        </catch>
        <catch name="return"/><catch/>
      </spawn>
    </do>
    <do type="delete">
      <spawn href="?NS=del&U=${U:escape}&HO=0&MB=${MB:escape}" sendreferred="true" onexit="${NEXT_DEST:unescape}">
        <receive name="NEXT_DEST"/>
      </spawn>
    </do>
  </card>
</wml>
```



```

        <catch/>
    </spawn>
</do>
<do type="vnd.up.send">
    <spawn href="#str$$$(U:noesc)">
        <setvar name="U" value="$$$(U:noesc)"/>
        <catch/>
    </spawn>
</do>
<do type="prev">
    <exit/>
</do>
<p mode="nowrap">Inbox: 4 unread
    <select name="U" iname="II">
        <option value="65540">
            FW: Lastest news (Patrick Chan)
        </option>
        <option value="65539">
            FW: Status 02 March 2000 (John McPhee)
        </option>
        <option value="65538">
            RE: Lunch? (Adam Smith)
        </option>
        <option value="65536">
            A wonderful message (Francis Bean)
        </option>
    </select>
</p>
</card>
</wml>

<wml>
<card id="menu">
    <p mode="nowrap">
        <do type="options" label="Back">
            <exit/>
        </do>
        <select iname="zz">
            <option onpick="#respond">Respond</option>
            <option onpick="#respond">Delete</option>
            <option onpick="#respond">Compose</option>
            <option onpick="#respond">Change Folder</option>
            <option onpick="#respond">Refresh Inbox</option>
            <option onpick="#respond">Delete All</option>
            <option onpick="#respond">Save Address</option>
            <option onpick="#respond">Save Message</option>
            <option onpick="#respond">Fax Message</option>
            <option onpick="#respond">Msg Info</option>
            <option onpick="#respond">Settings</option>
        </select>
    </p>
</card>
</wml>

```

In Example 4-3, the user can delete a message directly by highlighting the message header and choosing Delete from the Menu softkey. The code in the menu card is vastly simplified to show how the menu items would be rendered on the device. In a real application, the code would be generated dynamically so that the server could be accessed with each key press. It is only through this type of interaction with the server that the behaviour for each individual message can be controlled from the menu.

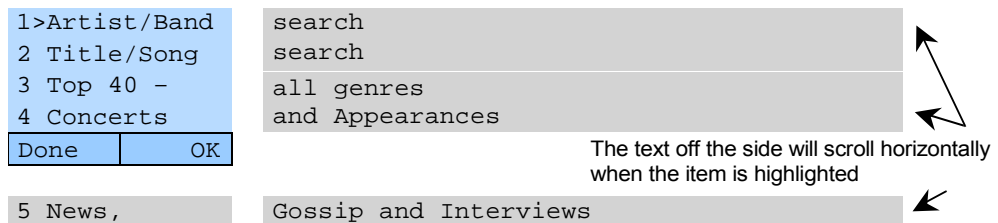
For a real example of how to code for this behaviour, use the UP.Simulator™ to connect to the Developer UP.Link™ and choose the UP.Mail™ application. To view the code, choose the Info menu then the Source option (F5) from the UP.Simulator. For more information on the UP.Simulator and the Developer UP.Link, visit <http://developer.openwave.com>.

- **Do not wrap items on a menu.**

Use `<p mode="nowrap">` to display each menu item on one line. See the “Openwave Menu Guidelines” on page 26 for the exception.

Example 4-4

Openwave Browser



```
<p mode="nowrap">
<select>
  <option onpick="#band" title="find">Artist/Band search</option>
  <option onpick="#song" title="songs">Title/Song search</option>
  <option onpick="#top" title="top">Top 40 - all genres</option>
  <option onpick="#conc" title="live">Concerts and Appearances</option>
  <option onpick="#news" title="news">News, Gossip and
Interviews</option>
</select>
</p>
```

In Example 4-4, `<p mode="nowrap">` prevents the menu items from wrapping. Instead Times Square text scrolling is used.

- **Be aware that links are displayed differently on different phones.**

Some phones may underline links and others may display them in square or angle brackets. Do not underline text or use brackets in text.

Nokia Menu Guidelines

- **Each menu should be a list of anchors.**

Mitsubishi Trium Menu Guidelines

- Each menu should be a list of anchors.
- Define labels of six characters or fewer for each anchor.

Longer labels may be truncated.

Making Phone Calls from the Browser

5

Some applications require the user to make a call or create opportunities for the user to do so, for instance, from a list of contacts, a phone number query, or an order form. This is an extension of general navigation; however, not all phones allow the user to make a call directly from the browser. If the phone does support calls from the browser, develop the application so that the user can retrieve the phone numbers.

Shared Feature Set: Calling Guidelines

- **When displaying more than one phone number on a card, display the primary number first.**

Some browsers pick out phone numbers from the deck automatically and list them in the order listed on the card.

- **Use a line or page break (`
` or `<p>`) between displayed phone numbers.**

Unless a break is added, the numbers may run together when the user selects the “Use number” menu function or presses the Send key.

Example 5-1

Nokia 7110 Browser

```
---- Robert Brown---  
Phone number:  
  
555-1212  
  
Options | Back
```

- **When terminating a call, do not assume that the phone may reinvoke the browser.**
- **Do not assume that the same deck will be redisplayed after the browser is reinvoked.**

The user may need to navigate back to the same deck.

Openwave Calling Guidelines

- **Embed code to make the phone call.**

Example 5-2

Openwave Browser

```
Hi Bob,  
Please call  
me when you  
have a  
Menu    Call
```

```
chance. Tom  
[1 408 555  
1543]
```

```
<do type="accept" label="Call">  
  <go href="wtai://wp/mc;14085551543">  
</do>
```

In Example 5-2 the user can call 1-408-555-1543 by pressing the Call softkey.

- **Provide a Call anchor label.**

This allows users to make a call directly from the browser.

- **Use `href="wtai://wp/mc;<phone number>"` instead of a link, unless multiple actions are desired.**

This allows the user to make the call directly from the card rather than accessing another card to retrieve the number and make the call.

- **Some browsers will return to the previous card when it is reinvoked after the call is terminated.**

To display a card other than the previous card in the history after the call terminates, use `<onevent type="onenterbackward">` in the card that generated the call.

- **Map the Send key action to the action of calling.**

In addition to using the softkey Call use `<do type="vnd.up.send">` so that users can also press the Send key to make the call. However, do not rely solely on this because not all phones have a Send key or map the Send key function.

Nokia Calling Guidelines

- **Include both the name and number on the screen.**
- **Provide a link to a card with the number if only the name is displayed.**

If the application displays only a name and not a number, such as in a search of a contact list, allow the user to place the call from the name list. However, it is also important to give the user the option of accessing an additional card with the name and number.

- **List the numbers for only one contact on one card.**

Do not list numbers for different people on one card. When more than one contact and phone number are listed, the contextual information is lost when the user selects the "Use number" item under the Options softkey or presses the Send key.

Mitsubishi Trium Calling Guidelines

- **Embed code to make the phone call.**

Use links or softkeys to make phone calls as in Example 5-3.

Example 5-3

```
<a href="wtai://wp/mc;14085551543" label="Call">Call 1-408-555-1543</a>
or
<do type="accept" label="Call">
  <go href="wtai://wp/mc;14085551543">
</do>
```

- **Provide a Call anchor label.**

This allows users to make a call directly from the browser.

- **Use `href="wtai://wp/mc;<phone number>"` instead of a link, unless multiple actions are desired.**

This allows the user to make the call directly from the card rather than accessing another card to retrieve the number and make the call.

- **Expect the browser to return to the previous card when it is reinvoked after the call is terminated.**

Using Multiple Selection Lists

6

Use multiple selection lists when the user can select more than one item on a list.

Openwave Multiple Selection List Guideline

- **Create only one <do> element or <do type="accept"> label, using five characters or fewer for the multiple selection list.**

Only one label is displayed; the second label is reserved for the device.

Example 6-1

Openwave Browser

Bob Brown	
X Angelica	Swansen
Dirk	Bigelow
Gary Kinser	
Done	Pick

X Jeff Lingle

```
<do type="accept" label="Done">
  <go href="getnames.cgi"/>
</do>
<select name="recip" multiple="true">
  <option value="bbrown">Bob Brown</option>
  <option value="aclemson">Angelica Clemson</option>
  <option value="dbigelow">Dirk Bigelow</option>
  <option value="gkinser">Gary Kinser</option>
  <option value="jlingle">Jeff Lingle</option>
</select>
```

Example 6-1 demonstrates how to create a multiple selection list. Notice that the code does not use `<do type="options">` because it would render a Menu softkey displaying the defined and prev actions.

- **Do not use <option onpick = href> for items on the selection list.**

Using `<option onpick>` does allow the user to select the item but may automatically display the next URL.

Nokia Multiple Selection List Guideline

- **Provide navigation links to the next URL.**

In addition to the <do> element or <do type="options"> label, create a link leading to the next card. This lets the user navigate via the link rather than from the Options softkey.

Example 6-2

Nokia 7110 Browser

----Send options----		<input type="checkbox"/> Bob Brown	
Names:		<input checked="" type="checkbox"/> Abby Clemson	
[Abby Clemson]		<input type="checkbox"/> Dirk Bigelow	
<u>Next</u>		<input type="checkbox"/> Gary Kinser	
		<input checked="" type="checkbox"/> Jeff Lingle	
Options	Back	Mark	Done

```
Example: <select name="recip" multiple="true">
  <option value="bbrown">Bob Brown</option>
  <option value="aclemson">Angelica Clemson</option>
  <option value="dbigelow">Dirk Bigelow</option>
  <option vlaue="gkinser">Gary Kinser</option>
  <option value="jlingle">Jeff Lingle</option>
</select>
<a href="send.cgi">Next</a>
```

Example 6-2 illustrates how to let the user easily navigate from a URL. The multiple selection list is displayed by the local Nokia 7110 user interface and is not a browser card. Providing a link allows the user to navigate to the next card without pressing the Options softkey.

- **Long text does not wrap.**

Ensure that the text fits on one line; longer text is truncated.

Mitsubishi Trium Multiple Selection List Guideline

- **Avoid using <do> options.**

Provide a link to the next URL instead

- **Long text does not wrap.**

Ensure that the text fits on one line; longer text is truncated.

Backward Navigation

Pay special attention to backward navigation because users tend to use the Back key or softkey to back out of an application. Users are more likely to trust applications with good backward navigation functions. In addition, backward navigation lets users leave the application without returning to the home deck. This may be particularly helpful for applications that are deeply embedded in the browser. For example, if a set of applications is three menus deep from the home card, backward navigation allows the user to return easily to the start of the set without re-navigating through the first two or three menus. In some cases, the design must prohibit navigation behind password-protected cards.

The default behaviour for all three browsers vary amongst each other and depending on the how the code is defined:

Table 7-1. Summary of backward navigation behaviour

	Openwave Browser	Nokia 7110	Mitsubishi Trium
No defined <code><do></code> or <code><prev></code> elements	<p>Primary softkey (labelled OK or Back returns users to the previous card in the history.</p> <p>Dedicated Back key returns the user to the previous card in the history.</p>	<p>Backward navigation is not possible.</p> <p>User must select Home from the Options menu to exit the application.</p>	<p>Right softkey, labelled Back, returns user to the previous card in the history. On some phones, if user navigates over a link, <code><input></code> or <code><select></code> element, the back functionality is removed.</p> <p>Left scroll key returns user to the previous card in the history.</p>

Table 7-1. Summary of backward navigation behaviour (*continued*)

	Openwave Browser	Nokia 7110	Mitsubishi Trium
<pre><prev> element defined with no label <do type="prev"> <prev/> </do></pre>	Dedicated Back key returns the user to the previous card in the history.	Back displays on the right softkey and returns the user to the previous card in the history.	Right softkey, labelled prev or Back, returns the user to the previous card in the history. Left scroll key returns user to the previous card in the history.
<pre><prev> element defined with a label <do type="prev" label="Done"> <prev/> </do></pre>	Dedicated Back key returns the user to the previous card in the history.	Back displays on the right softkey and returns the user to the previous card in the history. Defined label is ignored.	Right softkey labelled with the assigned label and returns the user to the previous card in the history. Left scroll key returns user to the previous card in the history.
<pre><do type="prev" label="Done"> <go href="url"/> </do></pre> <p>Where url takes the user to a defined URL</p>	Dedicated Back key sends the user to the defined URL	Defined label displays under the Options key and sends the user to the defined URL. Note that if a Template has been defined, then the Back key may act a <prev> action.	On some phones, if there is no other <do> element defined, the Done label will appear on the left softkey and send the user to the defined URL. The right softkey displays Back and returns the user back in the history. On other phones, the right hand key will be labelled Done and will send the user to the defined URL.

Shared Feature Set: Backward Navigation

- Always provide some kind of backward navigation.
- Map backward navigation to the next highest or most intuitive menu when direct backward navigation is not suitable.

Backward navigation is not always practical. For example, suppose the user confirms an action, such as making a purchase or deleting an item. Pressing the Back key should not take the user to the card confirming the purchase or the list with the deleted item. Instead, the application should either return to the application's home card, a login card, or a card that intuitively lets the user continue through the application or exit easily. Use an `<onevent type="onenterbackward">` task either to prevent backward navigation or to take the user past screens they should not revisit.

Example 7-1

```
<template>
  <do type="prev" label="Back">
    <prev/>
  </do>
</template>
<card id="start" onenterforward="#start2"
  onenterbackward="backtome.wml">
</card>
<card id="start2" title="The Start">
...
```

Example 7-1 shows how to prevent the user from navigating back through a card by catching the reverse entry into the card. In this case, the user is taken back an additional card.

- **Always provide a label for a `<do type="prev">`.**

The Nokia browser assigns the label `Back` instead of the defined label to the right softkey. If no label is defined, the label `Back` is automatically assigned to the right softkey. On some Mitsubishi Trium browsers, if a label is not defined, the right key will be labelled `prev` – which is not intuitive.

- **Create a second way to navigate backward when moving back in the history stack is not desirable.**

Provide backward navigation via `<do type="options">` if backward navigation should return the user to a higher menu or the top menu of the application. Use the label `Done` to indicate that backward navigation will take the user back more than one step. The label `Done` appears on the secondary softkey on the Openwave browser and also under the `Options` softkey on the Nokia 7110 browser.

Example 7-2**Openwave Browser**

Results 1/322
[Brown, A]
[Brown, Alex]
[Brown, Andrew]
Done View

Nokia 7110 Browser

----Search results---	Options items:
Results 1/322	View
<u>Brown, A</u>	Done
<u>Brown, Alex</u>	
<u>Brown, Andrew</u>	
Options Back	

Mitsubishi Trium Browser

Search results
Results 1/322
<u>Brown, A</u>
<u>Brown, Alex</u>
<u>Brown, Andrew</u>
Done View

```
<do type="options" label="Done">
  <go href="home.wml"/>
</do>
```

Example 7-2 shows how a URL helps the user navigate more easily. The Done and Back keys take the user to the application's home card.

- **Empty the application history when direct backward navigation is not possible.**

When it is not possible to go backward (for instance, after the user confirms an action), consider erasing the application's history list.

Example 7-3

```
<do type="options" label="Done">
  <go href="#clear"/>
</do>

...
<card id="clear" newcontext="true">
  <onevent type="onenterforward">
    <go href="home.wml"/>
  </onevent>
  <p>
  </p>
</card>
```

Example 7-3 shows how to empty the history for applications requiring passwords or secured data. This will delete variables in the handset. If this is not desirable, use an `onenterbackward` to catch the backward navigation.

■ **Save the values of variables when needed.**

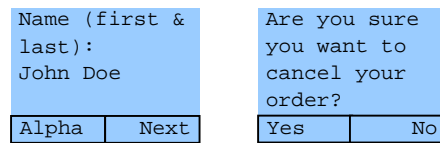
If the user exits an entry field, it may be helpful to temporarily save the values of all or some of the variables. This can reduce the amount of information the user must enter in the future. For example, it may be helpful to retain non-secure information entered in an order form, such as the name and address, so that the user does not have to re-enter it in the same browsing session. See “Cookies and Subscriber ID” later in this guide for information on using cookies for this purpose.

■ **Provide delete shields.**

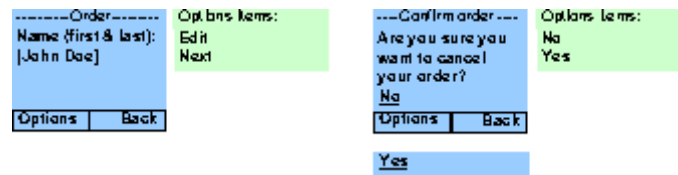
If backing out of an application may cause data loss, provide a card asking the user to confirm the action.

Example 7-4

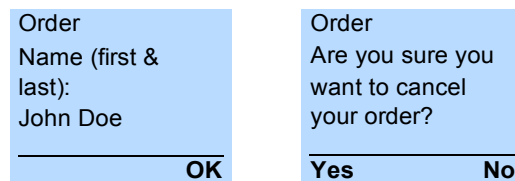
Openwave Browser



Nokia 7110 Browser



Mitsubishi Trium Browser



Example 7-4 shows how the confirmation card acts as a delete shield if the user backs out of the card with the Name prompt preventing loss of data.

■ **Retain data in wizard forms.**

Many devices have shared Clear and Back keys for text input. Therefore, if the user wants to edit or change the data in one of the forms, return to the first entry card so that the user will not have to delete already entered data. When the user navigates through each card, display the data already entered for that card. It is much easier for users to navigate forward through each card than to delete what they already entered in order to navigate back through previous cards.

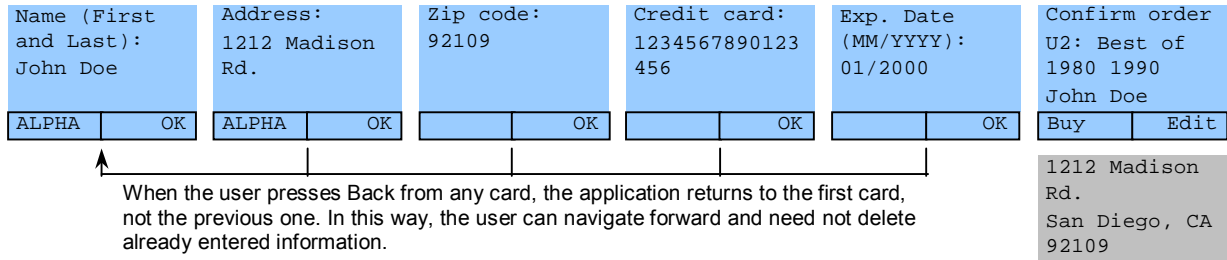
Openwave Backward Navigation

- Use activities when needed.

This way, the user does not have to repeatedly press the Back or Clear key, clearing already entered data.

Example 7-5

Openwave Browser



Example 7-5 shows the navigation of activities. See Example 3-9 for a complete description of the code.

Mitsubishi Trium Backward Navigation

- Define an appropriate label describing backward navigation when Back is not intuitive.

Define a label for the <prev/> event when Back is not intuitive. For example, if the previous card has an onenterbackward event.

Displaying Text

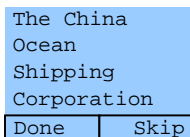
Some cards contain mostly text. For instance, applications that display email messages, news items, stock quotes, and confirmation or informative notices are text intensive. These cards often contain limited number of selection choices and are not used for text entry.

Shared Feature Set: Text-Display Guidelines

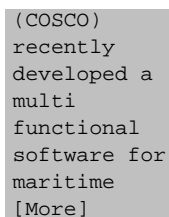
- Display about 500 to 800 characters per card.
- Define a More link if more information is available.

Example 8-1

Openwave Browser



```
The China  
Ocean  
Shipping  
Corporation  
Done Skip
```



```
(COSCO)  
recently  
developed a  
multi  
functional  
software for  
maritime  
[More]
```

- **Wrap the text.**

Do not use `<p mode="nowrap">`. The exception is a short header or text of little relevance to the user. For example, when displaying news for a specific company, use the `<p mode="nowrap">` mode to display the company name.
- **Define the primary label for navigation.**

Use the primary softkey or a link for forward navigation (accessing the next set of data); use the secondary labels for alternative navigation or other functions related to the application.

- **Define a Skip link to go to next related item.**

When displaying a series of related data, such as news stories or email messages, use a Skip link to allow the user to skip the current item and retrieve the next one. Do not use Next. Usability tests show that users tend to think it means “go to the next page” and not “go to the next item.”

- **Define labels for links.**

Create an appropriate label that reflects the action of the link. When the user selects the link, the label should change accordingly, for example, from View to Skip. Limit labels to five characters.

- **Use links sparingly.**

It is popular to put links at the end of display cards for alternative navigation, but do not define more than two or three links per card. User tests show that links at the end of the card often make it difficult for users to navigate out of the card. The reason is that labels corresponding to the link replace the labels for the primary action. This forces the user to scroll back up to access the primary action. The last link should match the default action for the card so that the user does not have to scroll up again.

- **Keep the text of links short.**

This avoids links that wrap beyond one line.

- **Place navigation links only at the top and bottom of the card.**

Do not embed navigation links in displayed text (unless it is context sensitive), because users will not understand that the links are not related to the data. Limit the length of navigation links to one line each, 13 characters if possible (the brackets will use 2 of the 15 characters).

Example 8-2

Openwave Browser

```
INTC
Last: 81 1/4
Chg: -1
Vol:
Done   OK
```

```
1,361,800
[News]
```

Openwave Text-Display Guidelines

- **Use mobile originated prefetch to access the next card.**

This shortens the time needed to retrieve the next set of information. While the user is reading one card, the next card can be retrieved and put into the cache.

Example 8-3

```
<head>
  <link href="page2.wml" rel="next" />
</head>
<card id="page1">
  <do type="accept" label="Page2">
    <go href="page2.wml" />
  </do>
  <p>
    Page 1 of 2<br/>
    ...
  </p>
</card>
```

Example 8-3 shows how to use prefetch to access the next card in a deck.

- **Do not use links on cards used to display results.**

On cards that ask the user to confirm an action (for instance, deleting a contact from an address book), that provide error explanations, or that display other results, use the `<do type="accept">` and `<do type="options">` labels instead of links. Map the safest or most common response to the `<do type="accept">` label.

- **Incorporate Done softkeys when possible.**

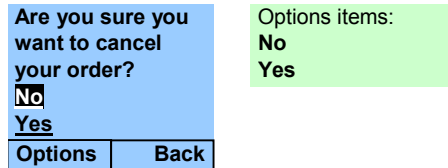
If no secondary functions on the displayed text are needed, add a Done softkey that returns the user to the next highest level within the application. This is often used in conjunction with activities so that the user can pop out of the current activity and return to the activity that spawned it.

Nokia Text-Display Guidelines

- Include extra navigation links, and allow users to use the Options softkey for additional navigation, when needed.

For example, when confirmation cards are used, add links after the question and place confirmation labels (such as Yes and No) under the Options softkey.

Example 8-4



```
<card id="cncl" title="Really Cancel">
  <do type="accept" label="No">
    <prev/>
  </do>
  <do type="options" label="Yes">
    <go href="cdapp.wml#start">
      <setvar name="name" value=""/>
      <setvar name="street" value=""/>
      <setvar name="cc" value=""/>
      <setvar name="exp" value=""/>
      <setvar name="zip" value=""/>
    </go>
  </do>
  <p>
  Are you sure you want to cancel your order?<br/>
  <anchor>
    <prev/>
    No
  </anchor>
  <anchor>
    <go href="cdapp.wml#start">
      <setvar name="name" value=""/>
      <setvar name="street" value=""/>
      <setvar name="cc" value=""/>
      <setvar name="exp" value=""/>
      <setvar name="zip" value=""/>
    </go>
    Yes
  </anchor>
  </p>
</card>
```

In Example 8-4, the links on the card allow the user to easily see and select the available choices in response to the query.

Mitsubishi Trium Text-Display Guidelines

- Create links to navigate off the card.
- Define appropriate softkey labels for confirmation cards.

Provide intuitive labels, such as Yes and No, for cards that require the user to confirm an action. For example, to confirm a purchase transaction or acknowledge data loss, define a `<do type="accept">` task with a Yes label to render on the left softkey and a `<prev>` task with the label No to render on the right softkey. Do not provide links on this type of card as they may be difficult to select.

Example 8-5

```
<card id="makechoice" onenterforward="#makechoice2"
onenterbackward="http://www.myservice.com/homepage.wml">

</card>
<card id="makechoice2" title="Your choice">
  <do type="accept" label="Yes">
    <go href="http://www.myservice.com/somethingnew.wml"/>
  </do>
  <do type="prev" label="No">
    <prev/>
  </do>
  <p>
    Are you sure you want to exit?
  </p>
</card>
```

Example 8-5 shows how to define a `<prev>` task with a softkey labelled No. When the user presses the No softkey the `onenterbackward` event rejects the confirmation and returns the user to the previous card.

Data Entry Queries

If the user must enter numeric or alphanumeric information, the application can use data entry queries to elicit it.

Shared Feature Set: Data Entries Guidelines

- **Minimise the number of input fields requiring alphanumeric entry.**

It is difficult and time consuming to enter text on mobile phones. Use selection lists, when possible, to avoid data entry. Other strategies are to store the user's entries and reuse them when appropriate.

- **Include a descriptive label for the `<do type="accept">` task.**

For example, use the label Find in applications that allow the user access information from a database.

- **Include both a descriptive title of 18 characters or fewer and an `<input>` element of 15 characters or fewer.**

On some phones, the title lets the user know what to enter. On others, the `<input>` element is used.

Example 9-1**Openwave Browser**

First and last name: John Doe	ALPHA	Send
-------------------------------------	-------	------

Nokia 7110 Browser

----- Order Info ----- First and last name: []	ABC First & last name: John Doe		
Options	Back	OK	Clear

```
<card id="ninput" title="Order Info">
  <do type="accept" label="Send">
    <go href="send.cgi"/>
  </do>
  <p>
    First and last name:
    <input name="fname" title="First & last name:"/>
  </p>
</card>
```

Example 9-1 shows how to create a card requesting data. Note that only one action label can be defined.

- **Limit input elements to 254 characters.**
- **Make password fields numeric only, when possible.**
It is easier to enter numbers than letters or symbols.
- **Do not mask alphanumeric passwords.**
Do not mask the entry. It is easier for the user to hide the display from others than to type with masked characters.
- **User names can be no longer than 32 characters; passwords, no longer than 20.**
Discourage long passwords by limiting the length.

Openwave: Data Entries Guidelines

- **Define only one action label for entry cards.**
Use only one `<do type="accept">` action. Additional options will hide primary navigation from the user.
- **Do not have text or links following a `<select>` or `<input>` element.**
This will render on what feels like a separate card, and is inefficient for the user.

- **Limit text before a `<select>` or `<input>` element.**

The browser will automatically ensure that the initial selected item or the input field is visible. If there is significant information above this then the user may need to scroll UP in order to see it. Text prior to these elements should be considered as a prompt for the element itself.

Nokia and Mitsubishi Trium: Data Entries Guidelines

- **Define a link to allow the user to navigate to the next card following a card with an `<input>` element.**

Provide a link following all the `<input>` and/or `<select>` elements on a card to allow the user to easily navigate to the next card.

Formatted Entry Fields

In some applications, the data entry queries can provide specialised format fields. These guide the user in entering the required information. For example, if user must enter a credit card number of 16 digits, the entry field can be formatted to accept 16 characters exactly. Other formatting is also possible; for example, the browser can be limited to accept only numeric entries.

Shared Feature Set: Formatted Input

- **Create informative titles.**

For example, if the field requires a date in a specific format, the field title should indicate what is required (mmyyyy) or “5 digit zip code”.

Example 10-1

Openwave Browser



```
<card id="date" title="Order CD">
  <do type="accept" label="OK">
    <go href="#confirm"/>
  </do>
  <do type="prev">
    <prev/>
  </do>
  <p>
    Date (mmyyyy):
    <input name="exp" title="Date (mmyyyy):" format="NNNNNN"/>
  </p>
</card>
```

Example 10-1 shows how to create a card requesting formatted data. The field accepts only numeric input of six digits. The label also informs the user what input is expected.

- **Force the entry type to numeric or alpha, if appropriate.**

For example, in some environments, postal codes are numeric only. For those countries, force the entry type to numeric so that the user cannot enter alphabetic characters.

- **Restrict the length of the string, if required.**

This is helpful for phone numbers or credit card entries. For example, use `maxlength="N"`; where "N" is the maximum number of characters to limit the length of the entry. Use NNN if exactly 3 numbers are required along with the `maxlength` attribute.

Example 10-2

```
Zip code:<br/>

```

Example 10-2 shows how to define a maximum length for an input string.

- **Prefill known data.**

With the Openwave browser, Mitsubishi Trium and Nokia 6210/6250 (but NOT the 7110 with browser version prior to 5.0) including forced characters in the format can help the user as they will need to enter less information. For example in an expiry date for a credit card field the format of `NN\2\0NN` could be used – this would only require the user to enter four digits for any month from January 2000 to December 2099.

In addition, with ALL browsers specifying a default value where one is sensible (or can be derived from personalisation information) avoids large quantities of data input. For example where a login and a password is absolutely required in some m-commerce sites consider pre-filling the username (which the user can always change) but not the password.

Openwave Formatted Input

- **Set the default input mode to numeric when the entry typically is numeric or starts with a number, but also allow the user to enter letters or symbols.**

For example, phone numbers generally consist of numbers only, but in some cases the user may also need to enter a special character, such as + or #. This applies to postal codes for some countries, purchase orders, ticket confirmations, and part numbers.

Example 10-3

```
<input name="phnum" type="phonenum"/>
```

Example 10-3 shows how to use `type="phonenum"` so that the input mode is initially numeric but so that users can change to alphanumeric or symbol mode.

- **Include symbols in the format to simplify text entry.**

Use symbols (such as / : ; - or even spaces) in the formatting string to indicate the expected input.

Example 10-4

```
NI no:


```

Example 10-4 shows how to embed special characters in the input field. This provides better visual information to the user. For example, all UK National Insurance numbers are of the format two letters, six digits and a letter, but the six digits are normally written in three groups of two. This helps the user to recognise information as they enter it, and helps them avoid mistakes.

- **When using the format `nN` or `nM`, keep in mind that the user can enter from 0 to `n` digits or characters.**
- **Set the proper case (lowercase or uppercase).**

Example 10-5

```
State (2 Letter Code)


```

Example 10-5 shows how to set the appropriate case when the user first accesses the input field. Here the letter A is for a letter whose case cannot be changed.

- **Use `M*m` formatting to obtain sentence-like formatting.**

Example 10-6

```
Last Name:


```

Example 10-6 shows how to set the case.

Nokia Formatted Input

- **Do not include symbols in the formats.**

Some phones will not accept symbols and may even fail if symbols (such as / : ; -) are supplied in the format.

- **In phone number entries, use appropriate formatting.**

If the field requires the user to enter a special character (such as + or #), do not define a format. Also, remember that phone numbers can vary in length (7, 10, 11 digits), so do not format for length.

- **When using the format `nN` or `nM`, keep in mind that the user must enter exactly `n` number of digits or characters.**

- **Use the `maxLength` attribute to prevent users from losing entered data.**

When the user exits a data query formatted with 3N (NNN) or 3M (MMM), the data is lost when the user does not enter exactly 3 numbers or characters. Set the `maxLength` attribute and state the number of required number of characters in the title to inform the user and avoid lost data.

- **The `emptyok="false"` attribute has no effect.**

This attribute does not prevent the user from leaving an input field empty.

Mitsubishi Trium Formatted Input

Formatted text entry is similar to the Openwave browser with the following exception:

- **Always provide a short descriptive title as part of the element**

This will be displayed when the user is editing the data

- **Handsets vary with their treatment of format strings**

One handset type requires a format of `M*m` to contain at least two characters, another type will accept a single character input with a format of `Aa*a` but will add a line feed character to the data that is entered to make it two characters long.

- **The `emptyok="false"` attribute has no effect.**

This attribute does not prevent the user from leaving an input field empty.

Forms can encompass one or more display or action types on a card or set of cards. There are two primary types of forms: forms that allow the user to enter information sequentially (wizard forms) and forms that show all of the fields in one list and allow users to choose what data to enter (elective data forms).

Shared Feature Set: Wizard Forms

- **Use a wizard form whenever possible.**

User tests show that wizards are always easier to use. Elective data forms require the user to focus on navigation instead of entering data, and mistakes are often made. This is particularly the case for Openwave browsers – the navigation features within the browsers make wizard forms very quick for users to navigate. The different presentation model of the Nokia and Mitsubishi browsers means that presenting all information in one card is sometimes more efficient for the user.

- **Link cards in a logical order.**

Provide a logical order that suggests what needs to be entered or selected and why.

- **Include descriptive text and titles for all `<input>` or `<select>` elements.**

- **Place each `<input>` or `<select>` element on a card of its own.**

- **The primary action should access the next card in the sequence.**

Bind the action of `<do type="accept">` to the next card with a required field.

- **The primary label for all cards in the sequence, except the final card, should be Next.**

The label for the last card in the wizard should explain the final action, such as Save, Send, Order, or Buy. Limit the label to five characters.

- **Create a final verification card displaying all entered or selected values.**

Allow the user to change the values if necessary.

- **Return to the first card in the wizard form for backward navigation.**

Many devices have shared Clear and Back keys for text input. For this reason, the user must delete the information in the card before returning to the previous card. To prevent this, map the Back key to the first card in the form. In this way, users can revisit the cards without deleting the information they already entered. See “Backward Navigation” on page 39 for more information.

Openwave Wizard Forms

- **Separate cards are not needed for each of the `<input>` elements.**

It is not necessary to place separate `<input>` elements on individual cards; however, doing so ensures breaks between the elements and allows for control of backward navigation. See the section “Backward Navigation” on page 39 for details. If all `<input>` elements are placed on the same card, the label OK is automatically bound to the `<do type="accept">` key, and pressing OK takes the user to the next defined `<input>` element. When the last `<input>` element is reached, the label defined for the `<do type="accept">` action is used.

- **The primary label for all cards in the sequence, except the final card, should be OK.**

The label for the last card in the wizard should explain the final action, such as Save, Send, Order, or Buy. Limit the label to five characters.

- **Text and links and links following a `<select>` or `<input>` element renders on a separate card.**

Cards that have a `<select>` or `<input>` element followed by a link or text will automatically render on another card when the user accepts the input or makes the selection. This may cause unnecessary extra keystrokes.

- **Limit the number of characters preceding a `<select>` or `<input>` element to 30.**

Long text preceding the option or input field will scroll off the screen.

Nokia Wizard Forms

- **Provide a link following the `<input>` or `<select>` element to the next card in the sequence.**

In addition to binding the `<do type="accept">` action, include a link after the required input.

Example 11-1

Nokia 7110 Browser

----- Order Info -----		ABC	
Name (first & last):		Name (first & last):	
[]		John Doe	
<u>Next</u>			
Options	Back	OK	Clear

In Example 11-1, the Next link allows the user to easily navigate to the next card in the sequence without having to select Next from the Options menu.

- **The label for the last link in the wizard should be no longer than 18 characters.**

Mitsubishi Trium Wizard Forms

The guidelines for the Mitsubishi Trium are similar to the Nokia, with the following exception:

- **Provide a link following the `<input>` or `<select>` element to the next card in the sequence.**

Do not provide a `<do>` action as well.

Example 11-2

```
<card id="empdata" title="Database">
  <p>
    Name (First Last)
    <input name="employee" title="First Last"/>
    <a href="#empdata1" label="Next">Next</a>
  </p>
</card>

<card id="empdata1" title="Database">
  <p>
    Date of birth (DDMMYYYY)
    <input name="dob" title="DDMMYYYY" format="NNNNNNNN"/>
    <a href="#empdata2" label="Next">Next</a>
  </p>
</card>

<card id="empdata2" title="Database">
  <p>
    Phone extension (4 digits)
    <input name="ext" title="Extn. (NNNN)" format="NNNN"/>
    <anchor label="Send">Send
    <go href="commit.cgi" method="post">
      <postfield name="empname" value="$(employee)"/>
      <postfield name="dateofbirth" value="$(dob)"/>
      <postfield name="phoneext" value="$(ext)"/>
    </go>
    </anchor>
  </p>
</card>
```

Example 11-2 shows how to create a link labelled Next that will allow users navigate from one card to the next. Provide an intuitive label for the final link in the sequence to confirm the action (for example "Buy", "Save", "Send").

Shared Feature Set: Elective Data Forms

- **Always try to find alternatives to using elective data forms.**

Use a wizard to link sequences of input queries and selections within an application. For example, if the user must supply a city, state, or zip code in a phone number search, use a menu from which the user selects one of the three. Then use a wizard to elicit the input.

- **Provide an appropriate label for the primary action.**

Label the `<do type="accept">` element with the desired result. For example, in a search query, define the `<do type="accept">` label as Find.

- **Display a final card showing the selected or entered results for all fields.**

When the application saves users data for future access, provide a card displaying the data entered and the elements on the form. For example, in an address book application, it shows the user all the data entered for that contact.

Openwave Elective Data Forms

- **Display the user's entered data on the form.**

Provide a card with the elements and variable strings to display the entered or selected values. Build a `<select>` statement with an `<option>` element for each item on the form. Within the `<option>` element, use the `<onpick>` action to access a second card containing either a `<input>` or `<select>` element (as desired). Include the name of the `<input>` or `<select>` element in the text associated with the option.

Example 11-3

Openwave Browser

```
1 First name:
2 Last name:
3 Zip: 92109
```

```
John
Doe
```

```
Find Edit
```

```
<option title="edit">
  <onevent type="onpick">
    <spawn href="#fin">
      <setvar name="f" value="$(f:noesc)"/>
      <setvar name="label" value="First name"/>
      <receive name="f"/>
    <catch/>
  </spawn>
</onevent>
First name: $f
</option>
```

Example 11-3 shows how to add a variable string to the end of a menu item to display the entered data to the user.

Nokia and Mitsubishi Trium Elective Data Forms

- After the final field, add a link performing the final function.

For example, if the card is a contact search form, add a Search link that allows the user to search on the entered and/or selected fields.

Example 11-4

Nokia 7110 Browser

----White Pages ----	Options items:
Last name:	Edit
[Smith]	Search
First name:	Done
[Andrew]	
Options	Back
Zip:	
[92109]	
Search	

The Search link in Example 11-4 allows the user to search on the entered fields without having to select the Options softkey and then the Search item.

Icons and Images

Images can enhance or support the displayed information so that the user can quickly review a list of items or see a trend. For example, a weather report can display a date along with an icon of the predicted weather. Likewise, an up/down arrow can precede a stock quote.

Shared Feature Set: Image Support

- Images need to be in wbmp format.
- Always include descriptive alt text for devices that do not support images.
- If the phone talks to an UP.Link Server Suite, the server will compile 1-bit bmp images to wbmp form.
- The WAP Forum does not currently define an animated image format.
- Do not define associate functions for areas within an image.

There is no way to associate an area within an image to an action (that is, there is no image map function).

- Be careful using images on cards with a timer element, because the timer may expire before the image is loaded.

Openwave Image Support

- Images larger than the display size are scrollable vertically, but not horizontally.

Thus, make images no wider than 40 pixels.

- Use preloaded images.

Openwave offers `localsrc` images that are preloaded into the devices that support images. The use of these images shortens network access time and creates a consistent user experience. A list of the `localsrc` images is available in the “Images” section of the WML reference that ships with the UP.SDK from Openwave, which can also be accessed at <http://developer.openwave.com>. See the “Image” section in the *UP.SDK Developer’s Guide* for `localsrc` images.

- Images will be aligned according to the attribute of the `<p>` element.
- Images can be displayed inline (along with text or a link).

- **Images can be included in an `<option>` element**, if the extension DTD is being used.

This will allow an icon to be displayed on the same line as a menu item.

Example 12-1

```
<option onpick="my_url">Email</option>
```

Example 12-1 shows how to embed an image inline with a menu item. In this case, an envelope is displayed before the text (✉ Email).

- **When delivering a deck that calls images, consider the use of a digest so that the image is displayed when the card has finished loading.**

This will load the deck and image simultaneously. The maximum digest size must be less than the MAX PDU, which is device specific but is approximately 2000 bytes.

This technique is also useful for pre-loading images that will be needed later in an application as the images are put into the cache of the handset.

Nokia Image Support

- **The maximum display size for an image is 96 x 44 pixels.**
- **Images are centred and displayed with nothing else on the line.**

Images wider than 96 pixels are left aligned and cropped on the right. Images longer than 44 pixels are sometimes scrollable, depending on the other content of the card. A link immediately below a tall picture can mean that it is not possible to see all of the picture.

6210/6250 handsets (and the 7110 with a browser model 5.00 or later) left align the image rather than centring it.

Mitsubishi Trium Image Support

- **Images can be displayed inline (along with text or a link).**

Example 12-2

```
Email</option>
```

Example 12-2 shows how to embed an image inline with a menu item. In this case, the envelope.wbmp is displayed before the text (✉ Email).

- **Ensure that the alt tag deck has meaning when image support is disabled by the user.**

The alt text will show in angle brackets – for example `<test>`

- **The gif image format is supported.**

Cache management is important for allowing quick access to previously viewed cards and controlling the display of time-sensitive content.

Shared Feature Set: Caching

- **Do not leave time-sensitive data, such as stock quotes, in the cache.**
- **Use a cache-control directive to specify how long a deck should persist in the cache of a device.**

A cache-controlled directive can prevent users from accessing outdated time-sensitive information, such as weather, traffic, or a stock quote. Cache-control directives are at a deck level rather than a card level.

Example 13-1

```
<meta http-equiv="Cache-control" content="max-age=600" forua="true"/>
```

In Example 13-1, the value 600 represents the number of seconds the data should be marked as valid the cache.

- **Do not build an application that relies on information residing in the cache.**

Openwave Caching

- **The default TTL (time to live) for a deck is 30 days or until memory is exhausted.**
- **Allow the browser to prefetch the next deck when the user is likely to access the next card.**

For applications providing textual information that users are likely to access in sequence, such as news or email.

Example 13-2

```
<head>
  <link href="page2.wml" rel="next" />
</head>
<card id="page1">
  <do type="accept" label="More">
    <go href="page2.wml" />
  </do>
  <p>
    Page 1 of 2<br/>
    ...
  </p>
</card>
```

Example 13-2 shows how to prefetch the content of the next card. When the user loads the deck, `page2.wml` is automatically loaded in the background as soon as the current deck is completed loading. This ensures that `page2.wml` is already in the cache when the user presses the `<do type="accept">` key.

- **Force the reloading of the deck for dynamic data.**

Example 13-3

```
<meta http-equiv="Cache-control" content="no-cache" forua="true" />
<meta http-equiv="Cache-control" content="must-revalidate" forua="true" />
```

Example 13-3 shows how to force a reloading of the deck each time the card is accessed in the forward or backward direction.

Nokia Caching

- **Force the reloading of a deck for dynamic data.**

Example 13-4

```
<head>
  <meta http_equiv="Expires" content="0" forua="true" />
</head>
```

Example 13-4 shows how to force a reloading of the deck each time the card is accessed in the forward or backward direction.

- **The maximum cache size is 40KB, and the maximum size for a single compiled deck is 1397 bytes.**
- **The default TTL for a deck, if one is not defined, is one day.**

Mitsubishi Trium Caching

The cache behaviour is unknown; however, the default cache time is approximately one hour.

Use cookies to store data, thereby reducing the amount of information the user must enter. The subscriber ID may also be used to personalise a service.

Shared Feature Set: Cookies

- **Cookies are not stored in the phone.**

However, the phone can access cookies if it is connected to an Openwave UP.Link Server. If it is unknown whether the device will be accessing an UP.Link Server, the information should be stored on the server where the application resides.

- **Use cookies as needed.**

Cookies may save the user from continuously needing to enter data from the keypad. If specific information needs to be stored, provide a login so the application can validate the user and access that user's personal information. Dynamically configure the login menu item so that it is displayed only if the application cannot identify the user. Although the Genie gateway supports cookies, note that there is no guarantee that all network gateway/browser combinations will universally support them.

- **If session information needs to be retained, consider the use of URL rewriting.**

Shared Feature Set: Subscriber ID

- **Use the Subscriber ID to allow the user to personalise the application.**

The Genie Gateway delivers the subscriber ID in every HTTP request with the following HTTP header:

```
HTTP_X_UP_SUBNO
```

A database can be used to identify subscriber and personalise the application based on what the user is doing.

Labels and Links

Depending on the type of application and type of information displayed, consistent labels should be used within the application and across other applications. Only the first letter of the label or link should be capitalised unless the word is always capitalised, such as OK.

The following labels apply to the Openwave and Mitsubishi Trium browsers since both browsers can display the title or label attribute as the softkey labels. However, these labels can also be used for the Nokia 7110 browser.

Accepted Labels and Links

- **OK: Used to select a menu choice in a choice card.**

Can also signify agreement to an operation, such as sending an email message.

- **Done: Used to allow the user to cancel the operation.**

Used to return the user to the Home deck, main menu, or intuitive card within the application.

- **Skip: Used to lead to similar data, such as the next news article or email message.**

Skip should always be the first softkey when used. May be repeated as a link at the end of a page as well.

- **View: Used to select an item in a menu of similar data, such as a list of stock quotes or email messages.**

Must provide additional detail. If Times Square scrolling was used to display the list data, pressing View should redisplay the data.

- **Details: Used as a link to get details of an item, such as a news article that is summarised by a headline.**

If information continues on another page, then use More instead of Details.

- **More: Used as a link at the end of a page of data to see the next page of related and similar data.**

- **Back: Not used for entry query cards.**

Back returns the user to the previous card in the history list. All three browsers have a dedicated Back key. Avoid defining a Back softkey on for the Openwave and Mitsubishi Trium browsers since users tend to rely on that rather than the dedicated Back key. Back on the primary softkey should be assigned to the `<do type="accept">` task only if no label for the `<do type="options">` is defined. If required, Back should be assigned to the `<do type="options">` label only if no other `<do type="options">` labels are required.

Conflicting Labels and Links

The following labels may conflict with items on a browser menu and may mislead the user.

- **Exit: May imply exiting the browser.**

- **Next: Often misinterpreted by users when used as a link.**

However, Next may be used as a softkey label in some wizard forms for the Nokia 7110.

- **Home: May imply the browser's home card.**

- **Bookmark: May conflict with the browser menu.**

Identifying the Browser

Several classes of web clients could potentially access your site, but for the sake of simplicity, this appendix addresses four possible situations:

- 1 A client that expects HTML
- 2 A Nokia 7110 browser
- 3 A device with the UP.Browser v3.1 or 4.x that supports WML 1.1
- 4 A Mitsubishi Trium browser

To identify which client is accessing your site, investigate two different HTTP headers, `HTTP_ACCEPT` and `HTTP_USER_AGENT`. While neither of these is part of the WAP specifications, they are both standard HTTP headers defined in RFC1945 (see <http://www.rfc-editor.org/rfcsearch.html>).

The first step is to look the `HTTP_ACCEPT` header and parse it for the inclusion of `text/vnd.wap.wml`:

■ Perl

```
#!/usr/local/bin/perl

$acc = $ENV{"HTTP_ACCEPT"};
$ua = $ENV{"HTTP_USER_AGENT"};
if ($acc =~ "wml"){
    deliver wml
}
else{
    print' Location: http://mysite.com/index.html'."\n\n";
}
```

■ Java

```
public void doGet (HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException{
    String acc = req.getHeader("Accept");
    String ua = req.getHeader("User-Agent");
    ServletOutputStream out = res.getOutputStream();

    if (acc.indexOf("wml") != -1){
        deliver wml
    }
    else {
        res.setHeader(res.SC_MOVED_TEMPORARILY);
        res.setHeader("Location", "http://mysite.com/index.html");
    }
}
```

■ ASP

```
<%response.buffer="true"
    Dim accstring
    Dim uastring
    uastring = request.ServerVariables("HTTP_USER_AGENT")
    accstring = request.ServerVariables("HTTP_ACCEPT")
    If (InStr(accstring,"wml")) Then
        Deliver wml
    Else
        Response.Redirect("/index.html")
    End If
%>
```

In all of these cases, it has been established that the client sends WML in the HTTP_ACCEPT header and can thus assume that WML should be delivered. If WML is not found in the HTTP_ACCEPT list, HTML is delivered. This ensures that HTML is delivered to web browsers and to spiders (crawlers, site indexers).

Once a WML device is found, it is possible to discriminate further to see exactly which device is accessing the site. Three possibilities are accounted for in this code: (1) the UP.Browser from Openwave, (2) the Mitsubishi Trium (3) any other device (including the Nokia browser).

■ Perl

```
if ($ua =~"UP.B" || $ua =~"UP/") {
    Print "Location: /opwv/index.wml \n\n";
}
elseif($ua =~"Mitsu") {
    print "Location: /mitsu/index.wml \n\n";
}
else {
    print "Location: /nokia/index.wml \n\n";
};
```

■ Java

```
if ((ua.indexOf("UP.B") != -1) || (ua.indexOf("UP/") != -1)){
    res.setHeader(res.SC_MOVED_TEMPORARILY);
    res.setHeader("Location", "/opwv/index.wml");
}
elseif(ua.indexOf("Mitsu") != -1){
    res.setHeader(res.SC_MOVED_TEMPORARILY);
    res.setHeader("Location", "/mitsu/index.wml");
}
else{
    res.setHeader(res.SC_MOVED_TEMPORARILY);
    res.setHeader("Location", "/nokia/index.wml");
};
```

■ ASP

```
If((InStr(uastring, "UP.B")) || (InStr(uastring, "UP/")))
    Response.Redirect("/opwv/index.wml")
ElseIf(InStr(uastring, "Mitsu"))
    Response.Redirect("/mitsu/index.wml")
Else
    Response.Redirect("/nokia/index.wml")
```

A

Identifying the Browser

Differences between Browser Types in Same Class

B

Genie strongly recommends that developers provide content that is tailored for the Openwave, Nokia and Mitsubishi browsers as detailed in the information above. The recommendation may become mandatory in time for content to be considered for the Genie Portal.

Genie understands that some developers may wish to take further advantage of the particular features offered by certain browsers. This is acceptable provided that all phones in that class are supported to at least the level defined above. It is not acceptable, for example to deliver content that is tailored for the features of a Nokia 6210 but which is very difficult to use on a Nokia 7110.

The following tables show observed differences between two Nokia variants and two Mitsubishi variants. These have been identified by observation only – if you use them you should check on real handsets that the behaviour is what you expect.

Table B-1. Nokia browser types

Behaviour	Nokia 7110 (browser version 4.xx)	Nokia 6210, 6250 and 7110 with version 5.00 or greater
Differentiation	HTTP_USER_AGENT string contains Nokia7110/1.0 (04)	
Layout tags	All text is left aligned and wraps	Text may also be centre and right aligned. <p mode="nowrap"> can have an effect, but it is recommended that this is NOT used as reading the text can be difficult
Tables	Every cell on a table is laid out on a line of its own. It is recommended that tables NOT be used	Tables are formatted acceptably by the browser

Table B-1. Nokia browser types (continued)

Behaviour	Nokia 7110 (browser version 4.xx)	Nokia 6210, 6250 and 7110 with version 5.00 or greater
<code><input format="..." /></code>	The browser does not enter fixed characters (those preceeded by a backslash) automatically, and may not handle the change between character type A and character type N sensibly. It is recommended that forced characters NOT be used A format such as 3N will only accept three digits (so NNN3N requires six digits)	Forced characters are inserted by the browser at appropriate points during the data input. The browser also handles changes between character types in input statements acceptably. Forced characters may be used. A format such as 3N will accept between 1 to 3 digits (so NNN3N will accept between 4 and 6 digits)
Images	Images are centre aligned	Images are left aligned
Non-breaking space character <code>&nbsp;</code> ;	Typically ignored, and not written as a space	Correctly represented on screen

Table B-2. Mitsubishi Trium browser types

Behaviour	Earlier browser type	Later browser type
Where found	Believed to be only in Trium Geo@ handset	Believed to be only in Trium Mars@ handset
Differentiation	Both contain identical HTTP_USER_AGENT string. HTTP_ACCEPT header does not contain text/x-vCard	HTTP_ACCEPT header contains text/x-vCard
Softkey behaviour	Left hand softkey is fixed as a labelled <code><do></code> action if exactly one non- <code><prev/></code> one exists, nothing if none exist, or Card (providing a menu) if more than one exists. Right hand softkey provides Back functionality (executing <code><prev/></code>) which may be relabelled. The right hand key changes context depending on what is highlighted on screen – a link will cause the title (or link) to be displayed, and <code><select></code> and <code><input></code> items can be highlighted and entered.	Right hand softkey provides fixed Back function which can be relabelled (and repurposed – see below) If no <code><do></code> actions of type not being "prev" are defined, left hand key is context sensitive, allowing links, <code><select></code> and <code><input></code> items to be entered. If exactly one <code><do></code> action of type not being "prev" is defined the left hand key is bound to this action and labelled and is not context sensitive (links, <code><select></code> and <code><input></code> must be accessed by other means). This is NOT recommended. If more than one <code><do></code> action of type not being "prev" is defined the left hand key says Card, and links to a menu of the actions in the order of their definition. As above, links, <code><select></code> and <code><input></code> must be accessed by other means.

Table B-2. Mitsubishi Trium browser types (*continued*)

Default action labelling	Unlabelled <do> is labelled with the type (truncated to fit, if necessary)	Unlabelled <do type="accept"> is labelled OK – all others are labelled with the type (truncated to fit if necessary)
Back key handling Example of	The back key can only have the action <prev/> – any attempt to make it have a different action will leave the key in place. The example code will show up as an action labelled "Main" on the left hand key (or down a Card menu) – "Back" will still show on the right hand key.	The back key may be redefined with any action. The example code will label the right hand key Main, and will go to home.wml
Layout tags	<p mode="nowrap"> has no effect. <small> text is smaller than normal text. <u> does not underline text	<p mode="nowrap"> truncates text to the screen width. There is no obvious mechanism to read the rest of the text. Use of this is not recommended. <small> text is the same size as normal text. <u> underlines text – not recommended as looks like a link.
Key accelerators	There are none	If a user presses a number key on a card which is a list of links, the phone navigates to the corresponding link. For example, if the user presses key "4" the 4 th link is followed. The browser does not provide numbers.
<input format="..." /> example <input format="Aa*a" />	The "*" prequalifier requires at least one character to be accepted. The example would accept only three or more characters of input.	The "*" prequalifier will match zero or more characters. Where Alphabetic characters are required, the browser may pack incomplete entries with an underscore character. The example would accept two or more characters of input. If the user enters only ONE character, an underscore character will be inserted by the browser to make up the required characters.
sendreferer in <go> statements	The full absolute URL is sent, irrespective of whether a relative one could be used.	Part of the referring URL is sent. This behaviour has not been completely characterised and may be incomplete. Where a relative URL could be provided, only the leafname was returned (even if this did not provide the proper path). For a URL which would need to have been provided as a full absolute URL the referrer has been presented as //www... (i.e. missing the http). Developers relying on this for security should check what gets sent with real handsets and verify that it meets their security objectives.

