OPENWAVE™

GSM Application Style Guide

For markets with both Openwave™ and Nokia™ Model 7110™ and Model 6210/6250™ WAP™ browsers

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

Part Number GSMS-01-006
February 2001

LEGAL NOTICE

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

Use other than for internal purposes, reproduction, modification or distribution without prior written authorization by Openwave Systems Inc. is strictly prohibited.

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

Style Guide Overview 1

This document gives developers for the GSM markets comprehensive guidelines for developing highly usable applications that run on two WAP™ browsers: the Openwave's 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 on different types of mobile browsers. For this reason, this document is written to help developers create optimized applications to run on either the Openwave or Nokia 7110/6210/6250 browser.

Organization 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 three points of view:

- **Shared feature set for creating for WAP applications for both the Openwave UP.Browser and the Nokia 7110/6210/6250 devices.** These guidelines address the shared feature set, that is, the features that are supported on both browsers. Because the browsers interpret the WAP standards differently when displaying content, this section outlines the approach that will work on both.
- **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.
- **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 guideline in conjunction with 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 because they are so similar. The few exceptions are explicitly labeled "Nokia 6210/6250 only."

The guide has two appendixes:

- Appendix A provides information on how to distinguish between various browsers.
- Appendix B provides sample applications. A design review and summary, and a technical summary are also provided. The sample applications and specific content may not apply to all environments. The samples do not specify how applications should work; they simply illustrate good design practices.

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 Specialize?

The WAP language specification for the Wireless Markup Language (WML) can be variously interpreted, and devices with browsers from different manufacturers exhibit differences in display and behavior. Although the WAP Forum is in the process of refining the language specification, developers have an immediate need: understanding how to build the best applications for browser phones on the market today. Fortunately, Openwave is working to ensure a consistent approach to the interpretation of WML across the 25+ Openwave handset licensees. However, other browsers, such as the Nokia 7110 browser, 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 efficiently with a minimum 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 customized for a specific browser.

Although authored by Openwave, this guide encourages developers to capitalize on the unique features of both Openwave and Nokia browsers. Openwave believes that 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` 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 browser handsets to ensure that it looks right and works correctly.

Why for the GSM Markets Only?

This document is written only for markets where both the Openwave and Nokia 7110 browsers are sold. It does not apply to other markets, because their requirements differ. Please refer to Openwave's web site for style guidelines for other markets. This guide focuses on the GSM markets because they have the most problems with differences among WAP browsers.

The following phones and SDKs were used to verify the recommendations in this guide: the Nokia WAP Toolkit v2.0, the Nokia 7110, the Openwave UP.SDK™, and a variety of phone models using the Openwave browser.

Usability Design Philosophies 2

Keep in mind a few design philosophies when building an application for a WAP browser phone in the GSM markets. 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.

- **Minimize 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 organized 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:

- **Specialize your application for Openwave and Nokia browsers.**

  To create a more usable application, determine the type of browser (Nokia or Openwave) in the code. With that information, develop customized 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.**

  Minimize 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 zip code as well as by city or state. 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 minimize text entry.

- **Personalize 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).

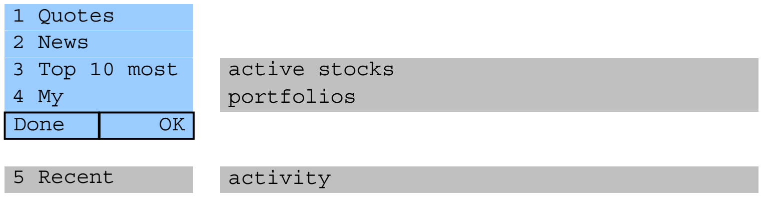
Browser Templates

A Word template has been created to help determine the look and feel of the design:

http://demo.openwave.com/styleguide/gsm/Template_Screenshots_gsmv11.zip

This template document provides various layouts for the different content displays.

Openwave Browser Template

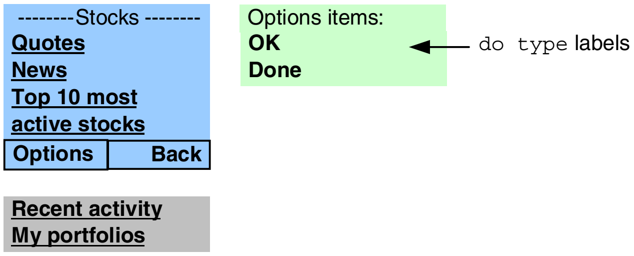


```
1 Quotes
2 News
3 Top 10 most        active stocks
4 My                 portfolios
Done      OK

5 Recent             activity
```

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 gray. 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 Back softkey supports the `prev` type and is not displayed if the application provides a label. In that event, the `prev` type appears under the Options softkey. Additional items and text are shown below the screen; the user and can access these by scrolling down. Text length for the Nokia 7110 browser should be tested on the Nokia SDK.

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 Service Options menu listed under the Options softkey).

Browser Properties for Phones in the GSM Markets

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 summarizes the properties of the Openwave and Nokia WAP browser phones (in the following 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
Number of characters per line	15 characters (mostly proportional fonts)	18 characters (proportional fonts)
Lines of display	2 to 8 lines	4 lines
Image/graphic support	Most phones support images and graphics	Supports images and graphics
Link support	Links display as: [link], `<link>`, or link	Links display as: link Nokia 6210/6250 only: Links can be selected by pressing the Send key
Scroll keys	Scroll keys for Up/Down, some support Left/Right scrolling	Roller key for Up/Down 6210/6250 only: Scroll keys for Up/Down

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

Property	Openwave	Nokia 7110
`<do>` **labels**	Label is associated with softkeys on the phone	Labels appear under Options softkey
Menu navigation	Develop as a `<select>` element or statement	Develop as a list of links
Back key for navigation	Dedicated Back key available except in entry queries	Programmable Back key is displayed on right softkey except in entry queries
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.
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
Alerts	Alerts are supported	Alerts are not supported

Openwave Browser Properties

The following properties are unique to the Openwave browser. Capitalize on them 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.

- **Bookmarking of sites is supported when connect through an Openwave gateway.**

  Users can bookmark a card from the browser menu unless the application specifies otherwise.

- **Most phones allow a press-hold to access the first nine bookmarks.**

- **Alerts are supported.**

  Priority levels should also be specified with the push notifications.

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

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

Nokia 7110 Browser Properties

- **The roller key can be pressed to activate the highest priority action.**

  The highest priority 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 (see “Nokia 7110 Browser Template”).

  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 Send key can be pressed to activate a link,** `<input>` **element, or** `<select>` **element.**

  Do not rely on the user discovering this behavior since no label is displayed for the send key action.

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

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

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

  The phone displays the `<do type="prev">` task on the second softkey only if the action defined is `<prev/>`. Do not provide a label; the browser will assign the label Back. If a label is defined, the softkey will not use it.

- **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 SDK.

Navigation Guidelines 3

To navigate Wireless Markup 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">`**.**

- **Create an intuitive label for every task except** `prev`.

  When a label is assigned to the `prev` task, the Nokia 7110 browser displays a Back label on the right softkey and renders the defined label under the Options softkey. The Openwave browser will not display a label for the `prev` task.

- **Capitalize only the first letter of labels of** `<do>` **elements except in words like OK.**

  Consistent style throughout all applications enhances usability.

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

- **Define a backward navigation action for each card.**

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

  Example 3-1

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

  In Example 3-1, there is no label for the `prev` element, and backward navigation is mapped to the previous card. On the Nokia 7110 browser, Back is rendered on the right softkey. On the Openwave browser, the `prev` navigation is mapped to the fixed Back key.

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

- **Do not bind an action of type `<noop/>` to a task of `<do type="prev">`.**

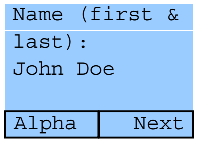
  This forces the user to return to the home deck, which is not always intuitive and may make users follow a long path to return to the application. Instead, bind the `prev` task to an intuitive place within the application, a starting point within the application, or to the home deck when that makes sense.

- **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 reentering data.

Example 3-2

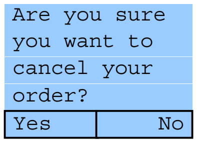
Openwave Browser



```
Name (first &
last):
John Doe

Alpha      Next
```

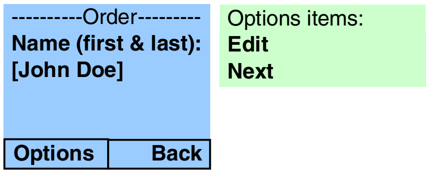
User presses the Back key



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

Delete shield

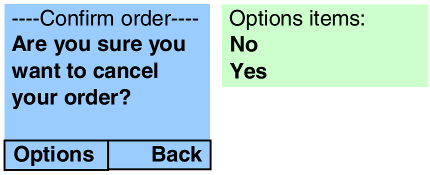
Nokia 7110 Browser



----------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

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

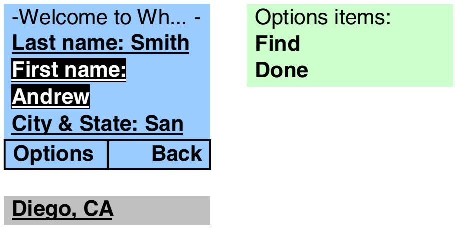
In Example 3-2, the code should be placed on the card that spawns the data entry form. If the user backs out after having entered the name, the confirmation card acts as a delete shield.

- **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 may display the title.

  Example 3-3

  Nokia 7110 Browser

  -Welcome to Wh... -
  Last name: Smith
  First name:
  Andrew
  City & State: San
  Options **Back**

  Diego, CA

  Options items:
  Find
  Done

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

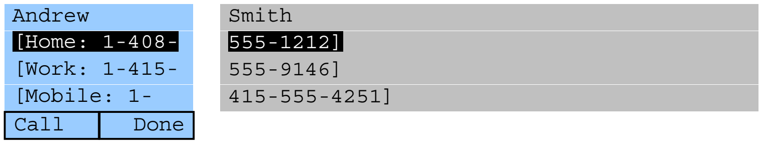
  In Example 3-3, the last eight characters of the title are truncated. Test the text length on the Nokia SDK to ensure that the entire title appears on the screen.

- **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.

  Example 3-4

  Openwave Browser

  ```
  Andrew
  [Home: 1-408-
  [Work: 1-415-
  [Mobile: 1-
  Call      Done
  ```

  ```
  Smith
  555-1212]
  555-9146]
  415-555-4251]
  ```

```
<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 may 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, which may not be appropriate for backward navigation. 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>` and `<option>` elements, and anchor 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>
  ```

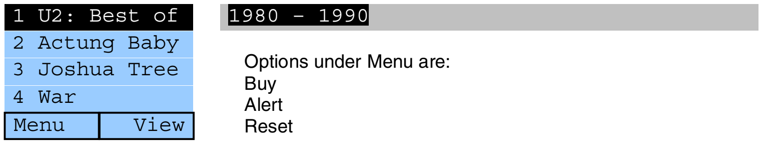
  In Example 3-6, the label Find is under five characters.

- **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



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>
```

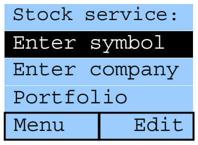
In Example 3-7, more than one option is bound to the `<do type="options">` task. For this reason, the browser renders the label Menu on the secondary softkey. The options under the Menu softkey are Buy, Alert, and Reset. In this case, when more than one option is required, build a card with the appropriate actions and bind this card to the `<do type="options">` key. Label the `<do type="options">` task Menu. This provides more descriptive action items and enables the use of key accelerators.

- **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



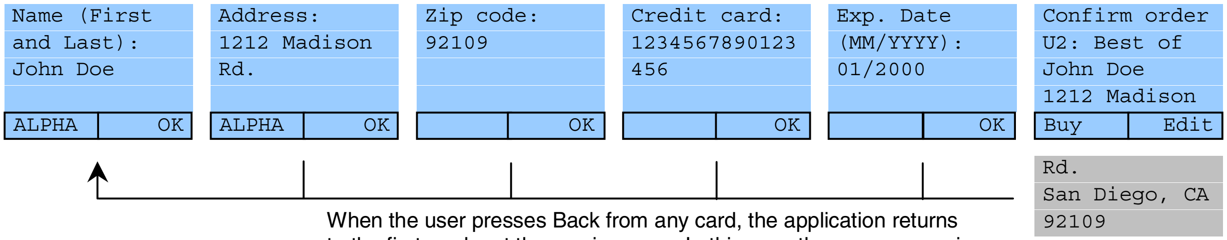
In Example 3-8, the header “Stock service” orients the user to the context.

- **Use activities to direct the Back key to the most intuitive page.**

  Activities 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 and the cache.

Example 3-9

Openwave Browser



```
Name (First
and Last):
John Doe

ALPHA        OK
```

```
Address:
1212 Madison
Rd.

ALPHA        OK
```

```
Zip code:
92109


             OK
```

```
Credit card:
1234567890123
456

             OK
```

```
Exp. Date
(MM/YYYY):
01/2000

             OK
```

```
Confirm order
U2: Best of
John Doe
1212 Madison
Buy        Edit
Rd.
San Diego, CA
92109
```

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 reenter data but can modify or remove it as necessary.

- **Make bookmark titles context sensitive.**

  Users should not need to change the title of the bookmark. For instance, instead of using the title “Stock Quote,” use “OPWV Quote.”

- **Ensure that the necessary data is available from a bookmarked item.**

  If the user accesses a bookmarked card, make sure that the appropriate navigation and data are included. For example, if the user bookmarks a stock quote, retain the ticker symbol even though the quote information is updated.

  Example 3-10

  ```
  <meta name="vnd.up.bookmark" forua="true"
  content="http://stock.com/quote.cgi?OPWV"/>
  ```

  In Example 3-10, when the user bookmarks the site, the label URL is stored regardless of what the URL for the current deck is. This allows for bookmarking the appropriate location of time-sensitive data.

- **Allow users to bookmark entry screens that lead to information.**

  For example, allow users retrieving a stock quote to bookmark the entry screen requesting the ticker symbol or company name.

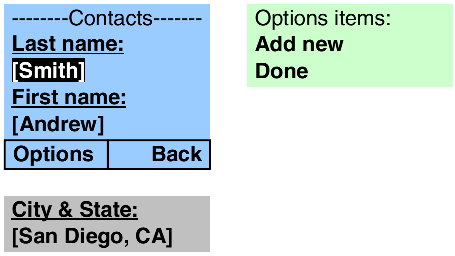
Nokia 7110 Navigation Guidelines

- **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="option">`, or `<do type="prev">`. Generally links should be used as a mechanism for forward navigation.

Example 3-11

Nokia 7110 Browser



--------Contacts-------
Last name:
[Smith]
First name:
[Andrew]
Options **Back**

Options items:
Add new
Done

City & State:
[San Diego, CA]

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

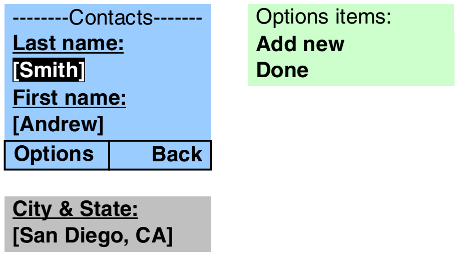
In Example 3-11, the `<do type="accept">` task calls `edit.wml`, and the `<do type="options">` task calls `find.wml`. On the Nokia 7110 browser, Edit and Find are accessible from the Options softkey.

- **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 3-12

Nokia 7110 Browser



--------Contacts-------
Last name:
[Smith]
First name:
[Andrew]
Options **Back**

Options items:
Add new
Done

City & State:
[San Diego, CA]

```
<do type="accept" label="Add new">
  <go href="newctct.wml"/>
</do>
```

In Example 3-12, the menu item “Add 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` to the `prev` task.**

  The Nokia 7110 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 3-13

```
<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 3-13 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 `onenterbackward`, thus loading `backtome.wml`.

- **Do not 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.

Menu Navigation 4

Navigating though 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.

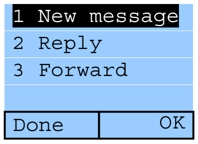
Openwave Menu Guidelines

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

  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=href>` element rather than an anchor.

  Example 4-1

  Openwave Browser

  1 New message
  2 Reply
  3 Forward
  Done | OK

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

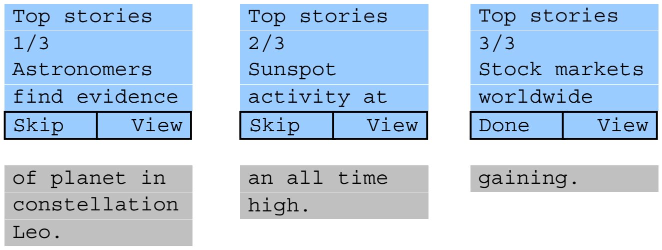
  In Example 4-1, menu items can be selected by pressing a number key.

- **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

  Top stories
  1/3
  Astronomers
  find evidence
  Skip | View
  of planet in
  constellation
  Leo.

  Top stories
  2/3
  Sunspot
  activity at
  Skip | View
  an all time
  high.

  Top stories
  3/3
  Stock markets
  worldwide
  Done | View
  gaining.

```
<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 multiple actions to be performed 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&amp;U=$$(U:escape)&amp;MB=$$(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="$$(NEXT_DEST:unesc)">
          <receive name="NEXT_DEST"/>
          <receive name="MB"/>
          <receive name="II"/>
        </catch>
        <catch name="return"/>
        <catch/>
      </spawn>
    </do>
    <do type="options" label="Menu">
      <spawn href="?NS=hdrMenu&amp;MB=$$(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:unesc)">
          <receive name="NEXT_DEST"/><receive name="MB"/>
          <receive name="II"/>
        </catch>
        <catch name="return"/><catch/>
      </spawn>
    </do>
    <do type="delete">
      <spawn href="?NS=del&amp;U=$$(U:escape)&amp;HO=0&amp;MB=$$(MB:escape)" sen
        dreferer="true" onexit="$$(NEXT_DEST:unesc)">
        <receive name="NEXT_DEST"/>
        <catch/>
      </spawn>
    </do>
    <do type="vnd.up.send">
      <spawn href="#str$$(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">
          <img localsrc="20" src="" alt="*"/>FW: Lastest news (Patrick Chan)
        </option>
        <option value="65539">
          <img localsrc="20" src="" alt="*"/>FW: Status 02 March 2000 (John McPhee)
        </option>
        <option value="65538">
          <img localsrc="20" src="" alt="*"/>RE: Lunch? (Adam Smith)
        </option>
        <option value="65536">
          <img localsrc="20" src="" alt="*"/>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 keypress. It is only through this type of interaction with the server that the behavior for each individual message can be controlled from the menu.

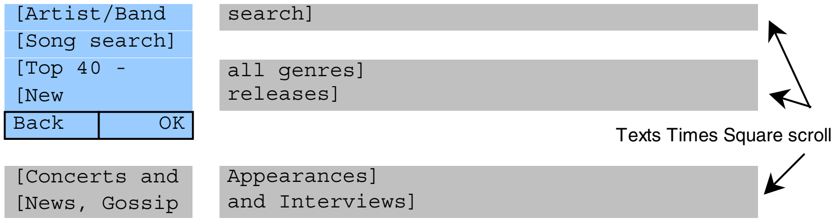
For a real example of how to code for this behavior, use the UP.Simulator™ to connect to the Developer UP.Link™ and choose the UP.Mail™ application. To view the code, choose the Info menu then the Source option (F5) from the UP.Simulator. For more information on the UP.Simulator and the Developer UP.Link,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

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

  Use `<p mode="nowrap">` to display each menu item on one line. This has no affect on the Nokia browser. See the Openwave Menu Guidelines for the exception.

  Example 4-4

  Openwave Browser



```
<p mode="nowrap">
  <option onpick="#band">Artist/Band search</option>
  <option onpick="#song">Title/Song search</option>
  <option onpick="#top">Top 40 - all genres</option>
  <option onpick"#new">New releases </option>
  <option onpick"#goss">News, Gossip, and Interviews </option>
</p>
```

  In Example 4-4, `<p mode="nowrap">` prevents the menu items from wrapping on the Openwave browser. 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 7110 Menu Guidelines

- **Each menu should be a list of anchors.**
- **Define labels of 18 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.**

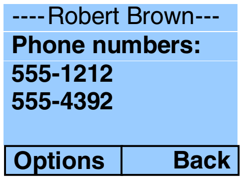
  Some browsers pick out phone numbers from the deck automatically and list them in the order listed on the card.

- **Use a line or page break (**`<br/> 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 numbers:

555-1212

555-4392

Include a line break between the phone numbers.

- **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.**

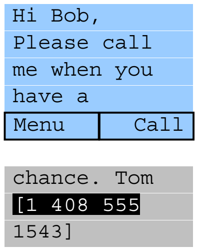
  The user may need to navigate back to the same deck that was displayed before the call.

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.

- **Expect the browser to 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 7110 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.

Using Multiple Selection Lists

6

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

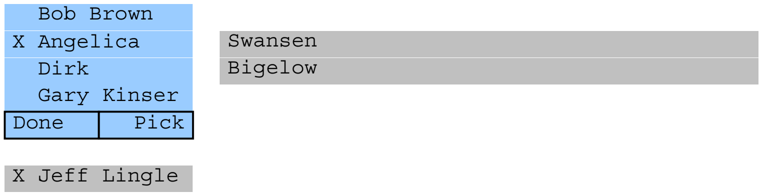
Openwave Multiple Selection List Guideline

- **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
  Dirk
  Gary Kinser
Done      Pick
```

```
Swansen
Bigelow
```

```
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 vlaue="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.

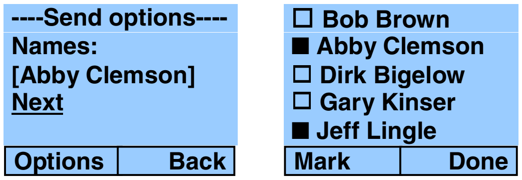
Nokia 7110 Multiple Selection List Guideline

- **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----
Names:
[Abby Clemson]
Next

Options | Back

□ Bob Brown
■ Abby Clemson
□ Dirk Bigelow
□ Gary Kinser
■ Jeff Lingle

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.

Backward Navigation 7

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. Also, 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 renavigating through the first two or three menus. In some cases, the design must prohibit navigation behind password-protected cards.

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

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

  Example 7-1 shows how to prevent the user from navigating back through a card in a password-protected area. In this case, the user would be taken back an additional card.

- **Do not provide a label for the `<do type="prev">` task.**

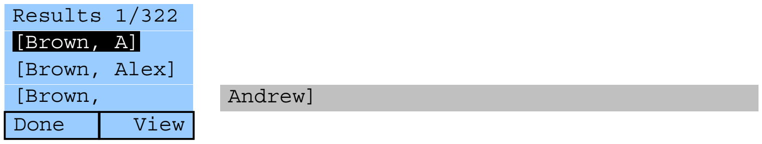
  The Nokia 7110 browser will assigns the label Back instead of the defined label to the right softkey. If no label is defined, the label Back is automatically assigned to the right softkey.

- **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.

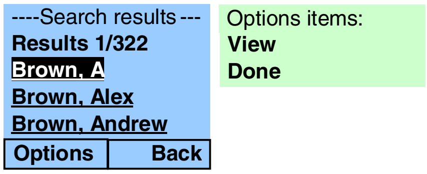
  Example 7-2

  Openwave Browser



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

  Nokia 7110 Browser



----Search results---
Results 1/322
Brown, A
Brown, Alex
Brown, Andrew
Options **Back**

Options items:
View
Done

```
<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 items take the user back 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), erase 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.**

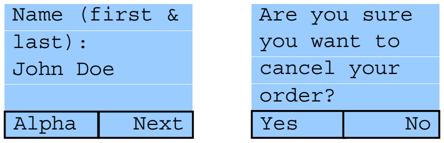
  If the user exits an entry field, it may be helpful to temporarily save the values of all or some of the variables. This can reduce the amount of information the user must enter in the future. For example, it may be helpful to retain nonsecure information entered in an order form, such as the name and address, so that the user does not have to reenter it in the same browsing session. See "Cookies" 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

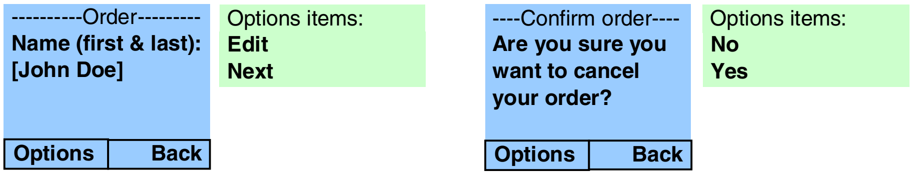


```
Name (first &
last):
John Doe

Alpha      Next
```

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

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

For both browsers in Example 7-4, the confirmation card acts as a delete shield preventing loss of data if the user backs out of the card with the Name prompt.

- **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.

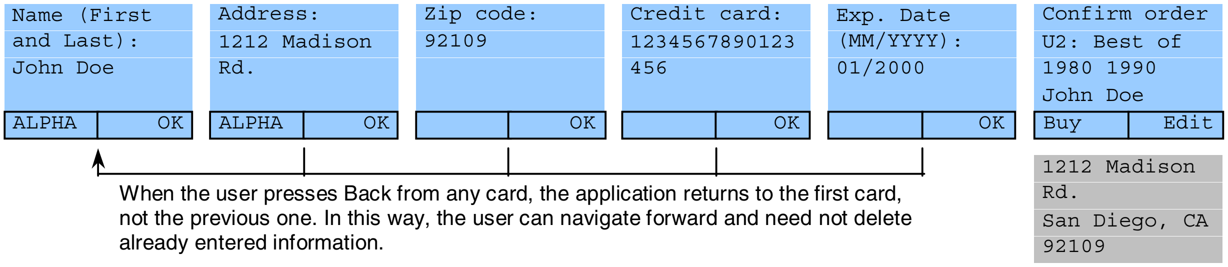
Openwave Backward Navigation

- **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



```
Name (First
and Last):
John Doe
ALPHA      OK
```

```
Address:
1212 Madison
Rd.
ALPHA      OK
```

```
Zip code:
92109
           OK
```

```
Credit card:
1234567890123
456
           OK
```

```
Exp. Date
(MM/YYYY):
01/2000
           OK
```

```
Confirm order
U2: Best of
1980 1990
John Doe
Buy      Edit
1212 Madison
Rd.
San Diego, CA
92109
```

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.

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

Displaying Text

8

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.

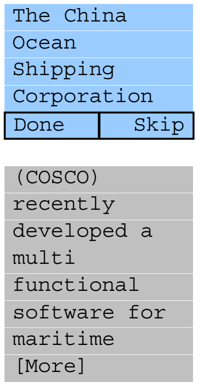
Shared Feature Set: Text-Display Guidelines

- **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.**

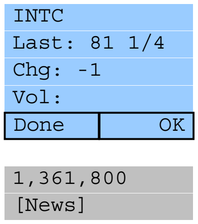
  This avoids links that wrap beyond one line.

- **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:
```

```
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.**

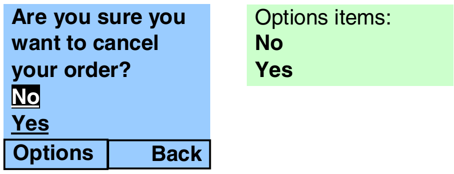
  If no secondary functions on the displayed text are needed, add a Done softkey that returns the user to the next highest level within the application. This is often used in conjunction with activities so that the user can pop out of the current activity and return to the activity that spawned it.

Nokia 7110 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



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

Options items:
No
Yes

```
<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 , the links on the card allow the user to easily see and select the available choices in response to the query.

Data Entry Queries 9

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

Shared Feature Set: Data Entries

- **Minimize 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.

- **Define only one action label for entry cards.**

  Use only one `<do type="accept">` task. Additional options will hide primary navigation from the user.

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

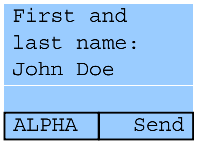
  For example, use the label Find in applications that allow the user access information from a database.

- **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

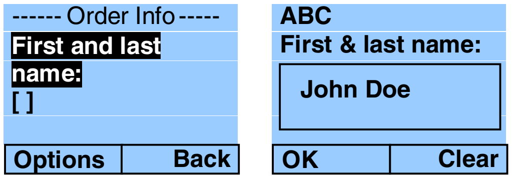
Openwave Browser



First and
last name:
John Doe

ALPHA Send

Nokia 7110 Browser



------ Order Info -----
First and last
name:
[ ]

Options Back

ABC
First & last name:
John Doe

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> element s 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.

Formatted Entry Fields 10

In some applications, the data entry queries can provide specialized 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.

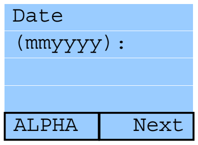
Shared Feature Set: Formatted Input

- **Create informative titles.**

  For example, if the field requires a date in a specific format, the field title should indicate what is required (mmyyyy).

  Example 10-1

  Openwave Browser



Date
(mmyyyy):

```
<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 no longer than 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.

- **Prefill known data.**

  For example, for the Openwave browser and the 6210/6250 only the first two digits of the year (20xx) can be automatically filled for the user. Since the user can never change prefilled information, be careful not to prefill any information that is subject to change.

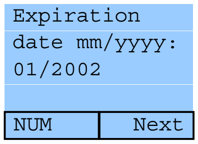
Openwave Formatted Input

- **When appropriate, delay displaying the softkey label until the user supplies data for a required entry field.**

  Use `emptyok="false"` to prohibit users from continuing without entering data.

Example 10-2

Openwave Browser



Expiration
date mm/yyyy:
01/2002

```
<card id="exp" title="order cd">
  <do type="accept" label="OK">
    <go href="#confirm"/>
  </do>
  <p>
   Expiration date mm/yyyy:
   <input name="exp" title="exp date mmyyyy" format="NN\/\2\0NN"/>
  </p>
```

Example 10-2 shows how to add a slash and the 20 automatically after the user enters the first two digits. The `<do type="accept">` softkey label appears only after the user has entered the complete date.

- **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 is also true for 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 / : ; -) in the formatting string to indicate the expected input. Example 10-3 shows how to embed special characters in the input field. This provides better visual information to the user.

- **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 10-4

  ```
  State (2 Letter Code)
    <input name="state" format="MM"/>
  ```

  Example 10-4 shows how to set the appropriate case when the user first accesses the input field.

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

  Example 10-5

  ```
  Last Name:
    <input name="lname" format="M*m"/>
  ```

  Example 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.

Forms

11

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.

- **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 35 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“Openwave Backward Navigation” on page 38 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.**

  Beware that this may cause unnecessary extra key strokes to navigate forward.

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

  Large amounts of text above an `<input>` or `<select>` element will be scrolled off in order to display the input field or first option.

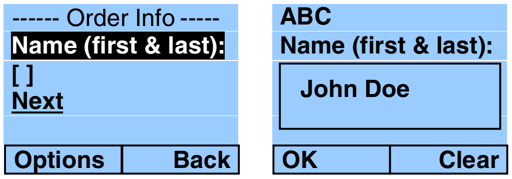
Nokia 7110 Wizard Forms

- **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



  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.**

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">` task 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 user data for future access, provide a card displaying the data entered and the elements on the form. For example, in an address book application, show the user all the data entered for that contact.

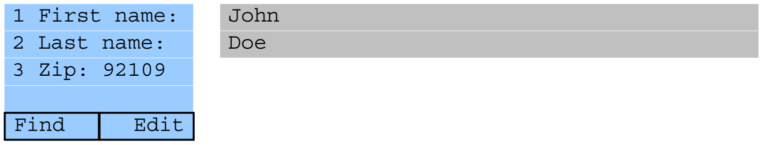
Openwave Elective Data Forms

- **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 11-2

Openwave Browser



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

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 11-2 shows how to add a variable string to the end of a menu item to display the entered data to the user.

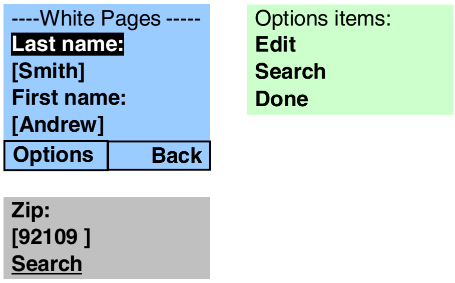
Nokia 7110 Elective Data Forms

- **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 11-3

Nokia 7110 Browser



----White Pages -----
Last name:
[Smith]
First name:
[Andrew]
Options **Back**

Zip:
[92109]
Search

Options items:
Edit
Search
Done

The Search link in Example 11-3 allows the user to search on the entered fields without having to select the Options softkey and then the Search item.

Alerts 12

Alerts are currently a Openwave feature supported on the Openwave gateway. They add value to an application because they can notify users of new information in their areas of interest. The Nokia 7110 currently does not support the alert functionality.

- **Alerts support alert type (priority), time to live (TTL), alert removal, delivery status, and security.**

  Refer to the Openwave *UP.SDK Developer's Guide* for details on how to implement alert notifications.

- **Allow the user to turn off or change the alert setting for the application.**

  This may not be supported on all phones, but when possible allow users to customize settings according to their preferences.

- **Use only one alert inbox slot, and ensure that the same URL is used for all the application's alerts.**

Example 12-1

```
// Set Values of Variables
m_TTL = 3600;
m_AlertType = "D---";
m_AlertTitle = "News!";
m_Url = "http://www.newssite.com/storys/importantstories/latestnews.wml";
m_SubscriberID = argv[1];
m_HostName = "devgate2.uplanet.com";

// Convert the standard ansi strings to Unicode (WCS)
// strings. The COM interface expects WCS BSTR’s.
mbstowcs(host, m_HostName, sizeof(host));
mbstowcs(subs, m_SubscriberID, sizeof(subs));
mbstowcs(url, m_Url, sizeof(url));
mbstowcs(alertTitle, m_AlertTitle, sizeof(alertTitle));
mbstowcs(alertType, m_AlertType, sizeof(alertType));

long upnotifypPort = 8501;
the_result = m_Ntfn->NtfnSetNonSecurePort(upnotifypPort);
// Set the UP.Link to which you will send the notification
the_result = m_Ntfn->NtfnSetHost (host);

if (S_OK != m_Ntfn->NtfnPostAlert (subs, url, m_TTL, alertType, alertTitle,
0)){
    cout << "Alert Failed! \n";
  }
  else {
    cout << "Alert Success! \n";
  }
```

Example 12-1 shows how to set the URL for an alert. See the *UP.SDK Developer’s Guide* and the *UP.SDK Tools and API Guide* for more details.

- **Give alerts a short title of 15 characters or fewer.**

  The title should fit on one line.

- **Do not include specific data in a title.**

  The title is a way to group messages under one alert heading. The same alert title should be sent whenever messages are delivered from an application. This is the title that appears in the inbox, not the message title. That is, only one alert title appears in the inbox even if more than one message is sent to that box.

- **Make sure that the URL remains active for at least 24 hours and that the user can access the alert once it is in the inbox.**

- **Alerts are delivered to a device by specifying the subscriber ID in the UP.Link address.**

  The subscriber ID is delivered as `HTTP_X_UP_SUBNO` in every HTTP request from a device. If `HTTP_X_UP_SUBNO` header is not present in HTTP request then either the user is not provisioned (in which case alert cannot be sent) or the gateway is not an Openwave gateway.

- **The alert consists of at least the title string, URL, priority, UP.Link address, and subno (subscriber ID).**

  Example 12-1 illustrates how to set the title string and define the subscriber ID.

13 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 the `localsrc` images.

- **Images are 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.**

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

  Example 13-1

```
<option onpick="my_url"><img localsrc="envelope" src="" alt=""/>Email</option>
```

  Example 13-1 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, use 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.

Nokia 7110 Image Support

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

  Images wider than 96 pixels are left aligned and cropped on the right. Images longer than 96 pixels are scrollable.

Cache 14

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.**

  For example, 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 14-1

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

  In Example 14-1, 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.**
- **Force the reloading of the deck for dynamic data.**

  Use HTTP header content-location information to change the location of the deck in the cache. For example, this can be used in applications that have an “Update information” link to dynamic data.

  Example 14-2

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

  Example 14-2 shows how to force a reloading of the deck each time the card is accessed in the forward or backward direction.

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, use the prefetch API from the UP.SDK.

  Example 14-3

  ```
  <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 14-3 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.

- **Use the notification APIs from UP.SDK to invalidate an item in the cache.**

  For applications that can access the same data from multiple devices (such as a PC and phone), the UP.Browser™ can be instructed that any cached information related to the application is outdated when the data changes on the data-storage location. This forces the browser to request new data from the server.

Example 14-4

```
sub PostCacheOp
{
    local($subid, $server, $demo) = @_;

    $url = $CONTENT_URL;
    $ttl = 3600;
    $cacheOpCode = "invalidateurl";

    use OLE;

    $ntfn = CreateObject OLE ’Ntfn3Client.Ntfn3Client.1’;

    if (!$ntfn) {
        &AppUtils::ErrorExit(1, "Ntfn class error");

    } else {

        # Set the UP.Link server name
        $ntfn->NtfnSetHost($server);

        # Post the CacheOp
        $ntfn->NtfnPostCacheOp($subid, $url, $ttl, $cacheOpCode);

        # Get the notification return status
        $result = $ntfn->NtfnGetLastResult();

        # StatusNoContent indicates success
        if ($result != $CHTTPStatusNoContent) {

            $err = $ntfn->NtfnGetErrorDetail();
            if ($err eq "") {

                &AppUtils::ErrorExit($result, "No error detail");

            } else {

                &AppUtils::ErrorExit($result, $err);
            }

        } else {

            $deck = sprintf($CACHE_DECK);
            &AppUtils::OutputDeck($deck);
        }
    }
}
```

Example 14-4 shows how to use notification APIs to invalidate an item (such as information for a person in the contact list) in the cache.

Nokia 7110 Caching

- **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.**

Cookies 15

Use cookies to store data, thereby reducing the amount of information the user must enter.

Shared Feature Set: Cookies

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

  However, the phone can access cookies if it is connected to a 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 that 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.

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

- **Always provide an alternative to using cookies.**

  Make sure the code supports cases when the gateway/browser combination does not support cookies.

Labels and Links 16

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 capitalized unless the word is always capitalized, such as OK.

Since softkey labels are displayed in the main window on the Openwave browser, the following labels apply to Openwave browsers. 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 summarized 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. Both the Nokia 7110 browser and Openwave browser have a dedicated Back key. Avoid defining a Back softkey on for the Openwave 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 wizard forms for the Nokia 7110.
- **Home: May imply the browser’s home card.**
- **Bookmark: May conflict with the browser menu.**

A Identifying the Browser

Several classes of web clients could potentially access your site, but for the sake of simplicity, this appendix addresses three 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

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 parse the `HTTP_ACCEPT` header 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")
```

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. The following code first looks for an Openwave browser and if that is not found, delivers content formatted for the Nokia browser.

- **Perl**

```
if($ua =~"UP.B" || $ua =~"UP/"){
    print "Location: /opwv/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");
}
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")
Else
    Response.Redirect("/nokia/index.wml")
```

Sample Applications

B

This appendix discusses two sample applications that illustrate the guidelines already described and show how to modify applications for ease of use.

The sample applications have been designed to work well for the Openwave and Nokia 7110/6210/6250 browsers. Two types of sample applications are reviewed: a general-search application (accessing a stock quote) and an application for making a purchase (online CD shopping). The design of the applications was based on the following methodology:

- Determining who the user is
- Determining the user’s goal
- Making it easy to accomplish the goal
- Making the application easy to navigate
- Scaling the application to perform only the necessary function
- Creating consistency throughout the application
- Avoiding or reducing the amount of required text entry
- Avoiding situations that cause unnecessary errors

The WML code for these applications can be accessed online at:

```
http://demo.openwave.com/styleguide/gsm/index.wml
```

The entire directory structure is downloadable from:

```
http://demo.openwave.com/styleguide/gsm/code.zip
```

Stock Quote

This example applies the steps for creating usable applications to the building of a search application that uses wizards, entry fields, selection lists, and screens displaying long text sequences.

- **Define the user.**

  The expected user of this application is one who participates, or is interested, in the stock market, seriously or as a hobby. The user already knows how to look up a quote and read a quote. The range of expected users is from first time/novice users to advanced users of browser applications.

- **Determine the goal.**

  Goals include tracking owned stocks, following stocks of interest, and viewing stocks the user may have heard about. Users who own stocks may want to quickly view the price of certain stocks many times a day, some may want to know the final quote for the day, and others may just want to be informed when vast changes occur. Other users may want to know the news for a given company that might affect the price of the stock.

- **Make the goal easily attainable.**

  Users who own a number of stocks may want to view their stock portfolio to track changes throughout the day or once in a while. Additionally, a user may have heard about a stock and may be considering adding it to a portfolio. This user may want to view that stock once a day to see what it is doing. In either scenario, the user wants quick access to the information. Providing methods for setting and storing stock portfolio information or the last viewed quote gives users this quick access.

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

  Providing a choice between the ticker symbol and the company gives the user access to quotes even when the symbol is not known. Advanced functions, such as searching multiple symbols at once, may also help the user quickly access the desired information. Placing important and user-relevant information at the top of the menu makes it visible and easily retrievable.

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

  Since the primary use for the application is to retrieve a stock quote, the main focus is to provide quick and easy access to this information. Secondary functions, such as viewing news information or changing a portfolio, may require additional steps. Additional functions, such as setting alerts, may also be accessible through secondary paths.

- **Make the application consistent throughout.**

  Using a consistent set of labels for elements (or softkeys for Openwave browser phones) reduces the time the user needs to access option lists or review softkey labels. Additionally, using the same links from the stock quote screen gives users a consistent way to find information relevant to their needs.

- **Avoid text entry.**

  Two ways of reducing text entry are to display the last search or a list of the most recent searches and to allow the user to create portfolios. Also, providing partial match searches allows the user to enter a minimal amount of characters and select from a list of matches.

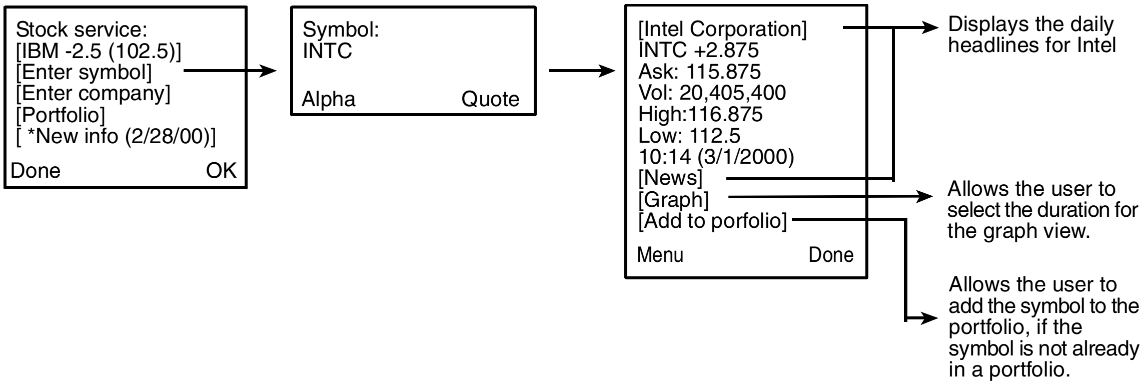
- **Avoid unnecessary user errors.**

  Limit the length of the ticker symbol query to prevent the user from trying to enter the entire company name. The application can also provide a list of potential matches if the user enters a symbol that cannot be found. Further, the application can allow the user to enter the company name (or partial match) if the desired company is not listed. Another way to avoid errors is to provide dynamic link labels, for example, allowing users to add a stock to a portfolio (or remove it from a portfolio) directly from the stock quote screen.

Application Overview

Figure B-1 and Figure B-2 show the differences when the application has been designed to take advantage of the user interfaces and features of both the Openwave browser (Figure B-1) and the Nokia 7110 browser (Figure B-2). Note that the display size is not based on any particular phone; rather, the figures are meant to illustrate content and forward navigation. Backward navigation is discussed in the design summary section. In these examples, the `<do type="accept">` label on the Openwave browser is displayed on the right softkey, and `<do type="options">` labels are on the left softkey. The labels displayed on the softkeys of the Openwave browser can be accessed via the Options softkey on the Nokia 7110 browser. Additionally, this application is only an example and has been designed for a given cultural environment. The actual fields and text should be modified for the appropriate market or culture.

Figure B-1. Stock quote application on the Openwave Browser

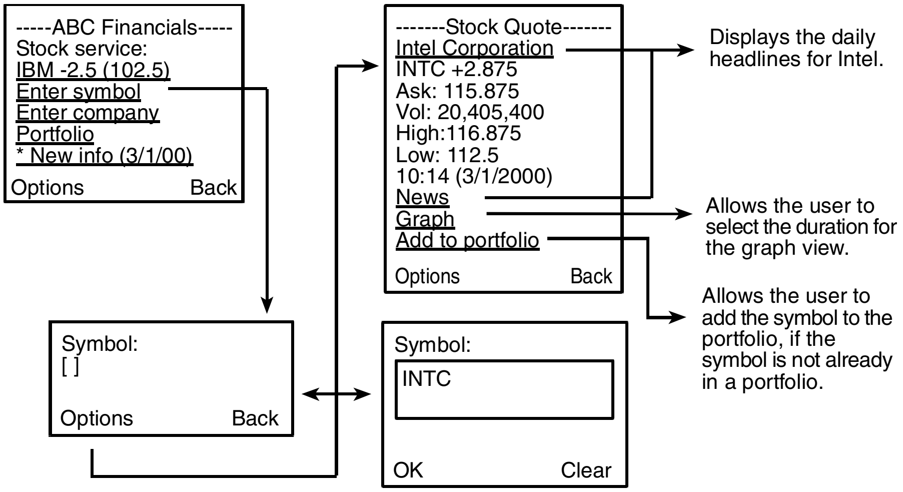


Displays the daily headlines for Intel

Allows the user to select the duration for the graph view.

Allows the user to add the symbol to the portfolio, if the symbol is not already in a portfolio.

Figure B-2. Stock quote application on the Nokia 7110 Browser



Displays the daily headlines for Intel.

Allows the user to select the duration for the graph view.

Allows the user to add the symbol to the portfolio, if the symbol is not already in a portfolio.

Design Summary

The design limits the number of steps it takes to access the desired information while retaining as much intuitiveness and usability as possible. Although only one path is shown, this summary also discusses additional capabilities to meet the design goals.

General Navigation

For both browsers, the application allows the user to search by a company's ticker symbol, or company name, and to access a quote from a created portfolio. The application also allows users to define and quickly access news or other information, such as major gains or losses, for a selected company.

The application design minimizes the number of steps it takes for users to access information relevant to their interests. For example, when the user first enters the stock service screen it shows the current value and change of the most recently viewed stock. The portfolio menu item has a dynamic location. When the user adds a stock symbol to a portfolio, access to the portfolio moves to the first item under the "Stock service" header. While there is a symbol in the portfolio, the portfolio item remains at the top of the menu. Access to the portfolio must be via a login, which should be stored as a cookie. In this way, the user needs to log in only once, and the login is remembered during all future access. This reduces the amount of typing needed each time to view a portfolio. However, if the browser is not connected to an UP.Link Server, the information must be stored on the server along with the application. Likewise, the first time the user enters the application, the first item allows the user to enter the stock symbol. If the user establishes a portfolio, the item "Recent searches" can be listed displaying the last few (5 or 10) searches. Finally, the item "New info (3/1/00)" displays the date of the most recently added information about selected companies.

When the user selects the "Enter symbol" item, the edit field allows the user to enter the ticker symbol for a given company. If the user enters a symbol that is not found, the application can display alternative similar symbols. The last item allows the user to search by company name if the ticker symbol is not in the alternative list.

The stock quote screen displays the company name to ensure the user has entered the correct company. The quote information shows the ticker symbol along with the most recent change and other information. The date and time indicate when the quote was last updated. Other links allow the user to access the company news, display (or fax) a graph for a given period of time (last month, last 3 months, last year, or last 5 years), and either add a stock to the portfolio or remove it. Additional items under the Menu key can allow the user to add to/remove from the portfolio without having to scroll to the bottom of the quote screen. Users can set the conditions for when they would like to receive alerts, such as what information is of value and how often they want to be notified. Users can also set information to be added to the item "New info (3/1/00)" or select the link "New search."

Backward Navigation

A Done label allows the user to go back to the main menu for the stock services. If the user presses Done after viewing the stock information, the most recently searched stock is added to the top of the searched list or to a Recent searches menu item, and the variable is cleared. This way, when the user selects the item "Enter symbol," the browser displays an empty query. If the user presses the Back key from the detailed quote screen, the Symbol query is displayed with an empty buffer so that the user can search for another quote.

Developing for the Individual Browsers

- **Openwave Browser.** The application as designed for the Openwave browser displays the menu items as a selection list (which allows the user to select a menu item by pressing a number). Inline graphics such as ↑ and ↓ are added to make the daily change more easily visible. Finally, users can set the conditions for when they would like to receive alerts, such as what information is of value and how often they want to be notified. In this case, the user can determine whether to store in the item "New info (3/1/00)" and whether to be notified when new information is stored.
- **Nokia 7110 Browser.** Creating an `<input>` element on the initial card of the application allows the user to directly select and enter data without first selecting a link. In this case, the first search query, "Enter symbol," is listed as an `<input>` element so that the user can search for a ticker symbol by choosing the Find item under the Options softkey. Technical information is described in the next section, "Technical Summary." Note that for environments where the option of searching by a stock ticker symbol is not available, the item "Enter company" should be designed as an `<input>` element.

Technical Summary

This application can be accessed in its two forms at:

```
http://demo.openwave.com/styleguide/gsm/stockapp/index.wml
```

This application is designed to be driven by a database (or live access to stock information), and these WML decks have been designed to show how this type of application would function and display data. The initial screen of the application should generate and display up-to-date data about the most recently viewed stock. If the application is running through the UP.Link WAP Gateway from Openwave, cookies can be used according to RFC2109 to store the last requested stock without requiring the user to log in.

Use variables to allow the main card in the application to display the most recently requested stock information:

```
<a href="#details" title="view">$stock $price</a>
```

The static version of this application uses an `onevent` task to set these variables when the card is first visited:

```
<onevent type="onenterforward">
  <refresh>
    <setvar name="stock" value="IBM"/>
    <setvar name="price" value="-2.5"/>
  </refresh>
</onevent>
```

When the user requests a new quote, the data is updated by the following code:

```
<onevent type="onenterforward">
  <refresh>
    <setvar name="stock" value="INTC"/>
    <setvar name="price" value="+2.875"/>
  </refresh>
</onevent>
```

In a real application, the values for the variables are supplied by the database or live stock information service.

For the Openwave version of this application, an additional graphic along with the information about the stock price is included. This icon is a `localsrc` image defined in a variable. The code below shows how to set the name of the `localsrc` icon:

```
<setvar name="icon" value="downarrow2"/>
```

This code fragment shows how the icon is used for display:

```
$stock <img src="" localsrc="$icon" alt=""/> $price
```

The first card can be displayed with a `<select>` element to simplify presentation. The `<spawn>` action is used to ensure that all activities that take place after the user requests information about a stock are dropped from the history stack when the user is done with it. This not only simplifies backward navigation but also ensures that a user does not inadvertently access stale data.

For the Nokia 7110 browser, you can simplify the stock quote request by placing an `<input>` element on the main card for the application. This eliminates one navigation card from the process of requesting a quote.

```
<card>
  <do type="accept" label="Quote">
    <go href="#quote"/>
  </do>
  <do type="options" label="Done">
    <prev/>
  </do>
  <do type="prev">
    <prev/>
  </do>
  <p mode="nowrap">
    Stock App<br/>
    <a href="#details" title="view">$stock $price</a>
    Enter symbol:
    <input name="sym" title="stock symbol" emptyok="false"
      maxlength="4"/>
    <a href="#entercom" title="quote">Enter company</a>
    <a href="#portfolio">Portfolios</a>
    <a href="#alerts" title="view">New info 3/1/2000</a>
  </p>
</card>
```

This code lets the user request a quote by selecting the input stock symbol event and invoking the `<do type="accept">` action.

CD Online Shopping

This example applies the usability guidelines to an online shopping application that includes a wizard form with selection lists, formatted entry fields, and screens displaying long text sequences. The search function in the online shopping application is different from the stock quotation application search because the information stems from the selection of the first link or choice.

- **Define the user.**

  The expected user of this application enjoys listening to music, has experience purchasing CDs from stores, may have limited experience purchasing CDs online, and may go to clubs or concerts. As a consumer in a music store, this user may expect to view at least a limited amount of information to make a decision about the CD before making the purchase.

- **Determine the goal.**

  The user would access the application for one or more of the following reasons:

  - to purchase a CD
  - to learn what new CDs are available
  - to learn if a favorite artists have any new CDs available
  - to find out what CDs or songs are on the top 40 (for a given genre)
  - to search the concert schedule for favorite artists
  - to search a concert schedule for a given city
  - to learn what songs are available on a given CD

  Secondary features can also be made available, such as providing information about clubs and similar artists. In addition to using the site for purchasing a CD, many users may use the site to find information.

- **Make the goal easily attainable.**

  Because there are many possible goals, the main menu should provide options corresponding to each goal. Accordingly, the first card on the application should provide many different search options for accessing the desired information. To facilitate the search process, the application can store a cookie of the last searched item in many of the categories. For example, if the user searches for concerts in San Francisco, the user may see San Francisco as the first item on the next visit to the concert search site. Within the Concerts item, a Schedules link can lead to information about concerts in searched cities or concerts by searched bands. This link allows the user to receive updates on searches. This storing of information for quick retrieval can be applied in other cases throughout the application. Cookies can be used to facilitate these updates on continuously searched information.

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

  From the list with all primary searchable items, the user can choose the path to achieve the goal. Links to other primary and secondary goals prevent redundant searches and provide easy access to other information.

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

  The user may have a goal that is not among those listed above. For example, a user may also want news reports or short biographies on selected artists or bands. Getting this additional information is probably secondary to the initial goal, and such information will not interest many users. For this reason, this information should not be available from the primary list, which is potentially long. It should become available only after the user selects the artist.

- **Make the application consistent throughout.**

  Ordering the menu list consistently helps the user anticipate the shortcut keys while the cards are loading. There are exceptions, however. For example, suppose the user is searching for new CDs. In this case, it makes sense to list genres according to which are most frequently searched rather than alphabetically because classical and opera CDs may not be released as frequently as others. However, alphabetical ordering of genres is useful when the user searches, in general, by genre.

- **Avoid text entry.**

  Two ways to reduce text entry are to allow the user to search from a selection list when possible and to use partial match searches. Additionally, the application can present selection lists to narrow search results after the user has entered or selected some information. Cookies also reduce text entry.

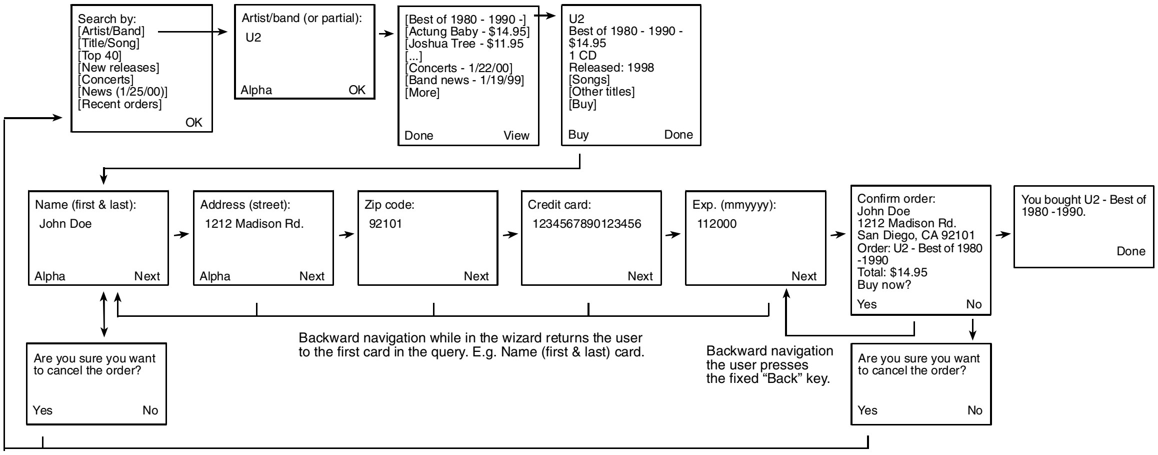
- **Avoid unnecessary user errors.**

  In this application, the user first selects a broad category and then narrows the search when possible. The search can begin after the user chooses the first link. Use a wizard form and formatted text fields to reduce the number of user errors. Also pay attention to backward navigation: the user should not be able to backtrack after placing and confirming an order. Delete shields prevent users from accidentally exiting a purchase order.

Application Overview

Figure B-3 and Figure B-4 show the differences when the application has been designed to take advantage of the user interfaces and features for both the Openwave browser (Figure B-3) and the Nokia 7110 browser (Figure B-4). Note that the display size is not based on any particular phone; rather, the diagrams are meant to illustrate content and navigation. In these examples, the `<do type="accept">` label on the Openwave browser appears on the right softkey; the `<do type="options">` label, on the left softkey. The labels displayed on the softkeys of the Openwave browser can be accessed via the Options softkey on the Nokia 7110 browser.

Figure B-3. CD online shopping on the Openwave Browser

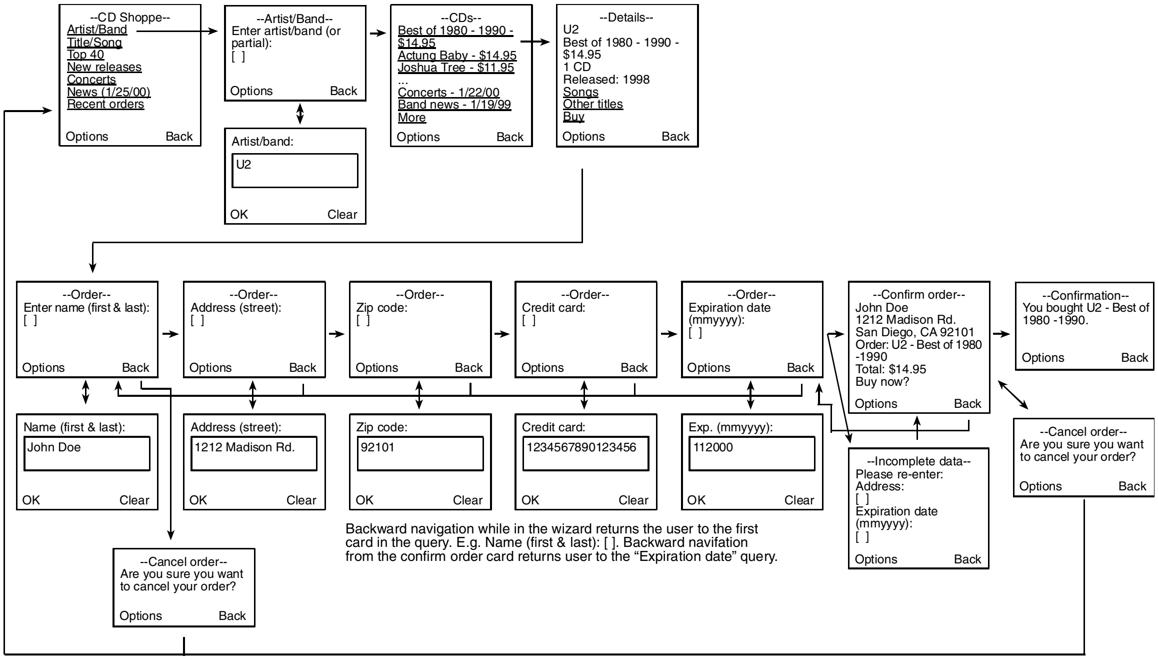


Backward navigation while in the wizard returns the user to the first card in the query. E.g. Name (first & last) card.

Backward navigation the user presses the fixed “Back” key.

Backward navigation when the order is canceled returns the user to the first card in the application.

Figure B-4. CD online shopping on the Nokia 7110 Browser



Backward navigation while in the wizard returns the user to the first card in the query. E.g. Name (first & last): [ ]. Backward navifation from the confirm order card returns user to the “Expiration date” query.

Backward navigation when the order is canceled returns the user to the first card in the application.

Design Summary

The application is designed to provide alternative methods for finding a set of information as quickly and easily as possible. Redundancy reduces the number of steps required. The navigational structure of this application gives the user an idea of the sequential steps required to complete an order and prevents loss of data when the user exits the order form. This structure allows for easier selection of items, more intuitive navigation, and a cleaner looking display. As with the Stock Quote application, only the main path is shown. The analysis describes other paths in order to meet the design goals.

General Navigation

The initial card of the application allows the user to search for information in various forms. When the user enters the item "Artist/Band name," a list of partial matches appears. When the desired artist or band is selected, the application displays a list of CDs along with links to other information, such as concerts, recent news, and bands with similar sounds. Because the primary goal is to find CD titles for a given artist, these are listed first, followed by the secondary information. Some of the secondary goals are listed and are also accessible through other menu items as primary search paths. Based on the main menu choice, the initial search determines the primary versus secondary goal. In either case, redundancy has been built into the application to allow access to the desired information while reducing the number of navigation steps to that information. In this case, the user does not always have to return to the main menu to find a particular set of information and is not overburdened by other information needed to complete the task. The last link on the menu, "Recent orders," gives easy access the most recent searches (if any exist) and lets the user view the results at a later time, if necessary. This menu item is available on Openwave browsers and Nokia 7110 browsers connected to an UP.Link Server.

Because all fields in the order form are required, a wizard is used. Formatted entry fields help the user avoid mistakes. Descriptive text gives feedback on the length or order of the entry. On the Nokia 7110 browser, an additional error card ensures that all fields are filled properly, because the user can use the Options softkey to skip a field. This card should not appear for the Openwave browser: there, the user does not see the primary softkey until satisfying the conditions of the formatted text field. For example, the OK softkey is not displayed on the query "Credit card number" until the user enters 16 digits. A final card allows the user to view and accept the order before committing to the purchase. The order of the softkey display (for the Openwave browser) or list of options under the Options softkey (for the Nokia 7110 browser) forces the user to make an active choice, preventing inadvertent orders.

Backward Navigation

The user can go back one step at a time when searching for desired information. Although the user can backtrack by pressing the Done key from these search results screens, it is probable that the user will press the Back softkey (on the Nokia 7110) or fixed Back key (on the Openwave browser) to go back one step. This provides quick backward navigation to the main menu for a new search or new type of search as well as an immediate backstep to change the course of the navigation (for instance, to view another CD by the same band).

Once the user has committed to the purchase, the application displays a final card, “Confirm order,” confirming it. If at this point the user decides not to make the purchase after all, the application displays a delete shield reminding the user that information will be lost. If the user exits the order, the application’s home card is displayed. If the user does not exit the card, “Confirm order” is displayed again. In either case, the variables already entered are retained (with the exception of the secure variables, such as the credit card number), in case the user decides to make any other purchases while information is stored in the cache. This caching of information reduces the number of fields that the user must fill in again.

Developing for the Individual Browsers

- **Openwave Browser.** The menus display as a selection list so that the user can quickly access the desired item with a single keypress. Backward navigation is optimized to allow the user to navigate backward intuitively, returning to the first card in the wizard without deleting entered information. While entering the purchase information, the user can revisit the forms to make changes. The Back key takes the user to the first card in the wizard. In this way, the user can backtrack without having to delete previously entered information: the user can quickly navigate forward through cards while editing previously entered information. If the user tries to go back from the first card in the wizard, a delete shield verifies that the user is aware that all data will be lost. Exiting back from the delete shield returns the user to the first card in the application. Similarly, if the user tries to go back from the card “Confirm order,” the application displays the first card in the wizard. Refer to the “Technical Summary” below for more information. When the user wants to exit the purchase order form, the cache must be cleared as soon as the user returns to the first card in the application (refer to the “Technical Summary” below for more information).

  Because the Openwave browser supports alerts, the application can allow the user to set notifications for information of interest. An alert can also be sent informing the user that new information has been added to the Favorites menu.

- **Nokia 7110 Browser.** Links are added following the `<input>` element to make the next data field easily accessible. For example, a link to the card “Street information” can be displayed on the card to edit the name. In this way, the user does not have to choose the Next item from the Options softkey. These links tell the user what data is required and help guide the user through the form.

  When the user wants to backtrack to the previous card in the wizard form, it is not necessary to go all the way back to the first card in the wizard, that is, the Name query card. Because the user must explicitly select the editor to enter the text, going back does not delete what has already been entered.

Technical Summary

The two versions of this application can be accessed here:

http://demo.openwave.com/styleguide/gsm/cdapp/index.wml

The designer should take into account that this application is driven by the interaction between scripts and a database. The code that is presented here is simply static WML to demonstrate how to present the results to the end user. There is only one active path currently through the code: the Artist/Band search.

The code is broken into two decks simply to limit the deck size. In a live, database-driven version of this application, the links would result in a query to a real database, and the `#results` card would come back as a deck of its own.

When the Album card is loaded, the album name and price are set as variables so that they can be used throughout the following cards and decks. This data can also be stored on the server to build the Favorites list for the user.

From the order confirmation screen, the Web server must be accessed to verify that all data fields have been filled. If data is missing, the application should display the card prompting the user to enter the missing information.

Once a user has purchased an item, the user's name and address should be stored. If an UP.Link WAP Gateway is being used, this information can be stored in a cookie. If it is unknown whether an UP.Link Server will be used, the data should be stored on the application server. In this case, the only reliable way to identify the user is to ask the user to log in to the site.

In the Openwave version of the application, notice that the DTD is from Openwave and *not* the WAP Forum. This is critical, because the application uses Openwave extensions to WML (sometimes called WML+).

In the `cdbuy.wml file`, when the user chooses the Buy link or softkey, a `<spawn>` action is used to create a new context. As a result, the cards used to accomplish the purchase are dropped from the history stack completely as soon as the user completes the purchase. At this point, the `<catch>` statement takes the user back to the starting point of the application. If the user cancels the order from the confirmation screen, the user returns to the album details screen, and any information already entered is cleared out of the variable space. The `<send value=>` element can be used to pass this information back to the calling activity. This simplifies backward navigation through the form wizard. Because on many devices the Clear key also performs the `prev` action, the user cannot navigate back through the cards in the wizard without deleting previously entered information. To prevent this situation, each `<input>` element is defined on a card of its own, and the `prev` action is bound to a `<throw>` task with a name value of `"bail"`.

```
<do type="prev">
  <throw name="bail">
    <send value="$street"/>
    <send value="$zip"/>
    <send value="$cc"/>
  </throw>
```

In this code, a card asks the user to enter the expiration date of a credit card. The `<throw>` returns the values that have already been entered back to the Name query card where the user first began to enter data.

This <throw> is caught by <catch name="bail"> on the card with id="buy" (the card where the user entered their name).

```
<catch name="bail">
  <receive name="street"/>
  <receive name="zip"/>
  <receive name="cc"/>
  <receive name="exp"/>
</catch>
```

Values passed from a <throw> are received in the order in which they appear in the deck with <throw>. The above code receives four values. If fewer than four elements were defined in the corresponding <throw> element, the values would be assigned to the variables in top to bottom order. For example, if a card returns only three values to the <catch> statement above, the variable "exp" would not receive a value.

If users navigate backward from the card with id=buy, they see the delete screen (cnclcrd) asking them to confirm their choice to exit the application. This card contains an <exit> element that passes back only the name and street address that the user entered. This data can be stored in a cookie so that the user never has to enter it again.

```
<do type="options" label="Yes">
  <throw name="reset">
    <send value="$name"/>
    <send value="$street"/>
  </throw>
</do>
```

This <throw> action is handled in the Album card with the following block of code:

```
<catch name="reset" onthrow="cdapp.wml">
  <receive name="name"/>
  <receive name="street"/>
</catch>
```

This code receives the name and street but discards the credit card information. The onthrow attribute causes the browser to navigate to the cdapp.wml deck immediately (similar to an onenterbackward event).

When the user confirms the purchase, this task is executed:

```
<exit>
  <send value="$name"/>
  <send value="$street"/>
</exit>
```

This code is also handled in the album card, but from within the calling <spawn> element, not from a <catch> statement.

```
<spawn href="#buy" onexit="cdapp.wml">
  <setvar name="album" value="U2 Best of 1980-1990"/>
  <setvar name="price" value="$$16.95"/>
  <setvar name="name" value="$name"/>
  <setvar name="street" value="$street"/>
  <receive name="name"/>
  <receive name="street"/>
```

Notice the <receive> elements, which retain the name and street data, and the <onexit> attribute, which redirects the browser to the main page of the application upon exiting.

For the Nokia 7110 browser, the text-entry mode is not automatically re-entered upon backward navigation, and the user can simply use the Back key to move through the history, retaining any data already entered. In this case the newcontext="true" attribute is needed, and the <do type="prev"> task can simply be defined as follows:

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

Also, some browsers may not display the softkey label over the <do type="accept"> key. As a result, it is a good idea to include a link bound to the same action as <do type="accept">, to make it clear to the user what should be the primary action.

```
<do type="accept" label="Next">
  <go href="#add1"/>
</do>
<do type="prev">
  <prev/>
</do>
<p>
  Name<br/> First &amp; Last
  <input name="name" title="Full Name"/>
  <a href="#add1">next</a>
 </p>
```

If it is known that the device will be accessing the application through an UP.Link Server, then cookies can be used to store not only the name and address of the user but also such information as a list of bands featured in recently purchased albums. This information can then be accessed from the links “Recent orders” and “Favorites,” on the main card of the application.