



Mobile Entertainment Platform 1.1 and 2.0
User Interface Guidelines

NOKIA
CONNECTING PEOPLE



**Copyright © Nokia Corporation 2001.
All rights reserved**

**Mobile Entertainment Platform
User Interface Guidelines
Release 2.0
Doc Rev. June 28, 2001**

The Nokia Group Finland reserves the right to make changes to this document and the software described herein at any time and without notice.

Nokia Group Finland
P.O. Box 102
FIN-00045 NOKIA GROUP
Tel +358 71800 8000
www.forum.nokia.com

This document is provided “as is” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. Nokia will not, in any event, be responsible for errors in the document or for any damages, incidental or consequential (including, but not limited to monetary losses), that might arise from the use of or inability to use the document or the information in it, even if Nokia has been advised of the possibility of such damages.

The document could include technical inaccuracies or typographical errors. Nokia welcomes customer comments as part of the process of continuous development and improvement of the document. Any such comments shall become the property of Nokia without any duty of compensation or obligation to used.

Please send comments, suggestions and corrections to the address given above.

Changes are periodically added to the information herein; these changes will be incorporated in new editions of the document. Nokia may at any time make improvements and/or changes in the product(s) and/or the program(s) described in the document.

Reproduction, distribution or transmission of part or all of this documentation in any form without the prior written permission of Nokia is prohibited.

Nokia and Nokia Connecting People are registered trademarks of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners, and are mentioned for identification purposes only.

GLOSSARY OF TERMS

NMES	Nokia Mobile Entertainment Service – The overall service including software, hardware and support. This encompasses people, process and technology.
MEP	Mobile Entertainment Platform – The platform including WAS, Portal, Cartridges and Database. This encompasses the technology.
WAS	Wireless Application Server – The application server that runs the game services (provides the business logic back end to the games)
Portal	The entry point for accessing the games running on the MEP. The Portal also provides login and registration functionality as well as access to community features.
GSF	Game Service Framework – The re-usable code that 3 rd party vendors use to create their games (allows their games to run on the MEP Portal)
WBMP	WAP Bitmap (graphics format)
Game Tools	A series of game features that can be used by 3 rd party games (e.g. Invitations, Open Save etc.)
Just-in-time	A design approach used to create an interface which the user is able to use to achieve a specified goal, with no prior experience or training.
Make-over	This is designer terminology for a complete redesign.
Face	Short for typeface, known by computer types as a font
Copy	Short for body copy, which is a designers name for a section, paragraph or column of text, as opposed to a heading text or the style of that text.
Widget	Any user interface element, control, button, link or even a text label within the screen content.
Leading	The vertical spacing between lines of text.
Browser	Any WML presentation device e.g. WAP phones, PDA WAP browsers, PC WAP browsers.

Intended Audience

This document describes the user interface guidelines for the Mobile Entertainment Platform. The guidelines are primarily for third party developers creating WAP applications for the platform, and also for internal Nokia developers creating internal MEP applications and extensions to the service which require a user interface. Whilst the guidelines are aimed at MEP applications, many of the principles detailed here are relevant for all WAP development and even software design generally.

TABLE OF CONTENTS

1.	INTRODUCTION	1
1.1	Nokia MEP version 1.0 versus versions 1.1 and 2.0	2
2.	USER INTERFACE GUIDELINES.....	3
2.1	Game name	3
2.2	Screen names (card titles)	3
2.2.1	Give every screen an informative and unique title	3
2.2.2	Why is the title so important?	3
2.3	Text.....	4
2.3.1	Text is the most important element of the WAP user interface.....	4
2.3.2	Avoid Ambiguities and Jargon	4
2.3.3	Avoid abbreviations.....	4
2.4	Text Formatting	5
2.4.1	Text formatting is also an important interface design tool.....	5
2.4.2	Left justify sentences and paragraphs of text.....	5
2.4.3	Center justify navigational links	5
2.4.4	Center justify and italicize headings.....	5
2.4.5	Define all text as ‘small’	6
2.4.6	Condense sentences used for links and labels.....	6
2.4.7	Capitalize nouns and verbs in links	7
2.4.8	US English Spelling.....	8
2.4.9	Avoid blank horizontal lines.....	8
2.5	Navigational Menus and Links	8
2.5.1	Center navigational links	8
2.5.2	Links navigating in a reverse direction are listed last.....	9
2.6	Menus within an application or game.....	9
2.6.1	The main menu of the game is structured to follow a consistent standard	9
2.6.2	Other menus display menu items in the order of usage (and frequency of usage)	10
2.6.3	Lists of links representing objects or items, are left justified	11
2.6.4	Long lists of items.....	12
2.6.5	Empty lists	12

2.7	The navigational model	13
2.7.1	Left to right spatiality	13
2.7.2	Top to bottom spatiality	13
2.7.3	Limit the use of 'back' terminology	13
2.7.4	The breadth and depth of the navigational model (and the number of items in a list).....	16
2.8	Metaphors and themes	17
2.8.1	Using metaphors to the real world	17
2.8.2	Themes (also known as skins)	18
2.9	Help.....	19
2.10	Avoid mapping interface options to the hardware phone buttons	20
2.11	Game 'Playability'	20
2.11.1	Rewards	20
2.11.2	Randomness	20
2.12	User testing	21
2.12.1	Start the 'software test' with a well thought out design on paper.....	21
2.12.2	Testing should be frequent with the end users being the experts.....	21
2.12.3	The most important feedback is non-verbal.....	21
2.12.4	Interpret the feedback and redesign accordingly	22
3.	THE TYPICAL STRUCTURE OF A SAMPLE GAME	23
3.1	The Splash Screen.....	24
3.2	The main menu of the game.....	24
3.2.1	Continue Game	25
3.2.2	Start New Game	26
3.2.3	Open Saved Game	26
3.2.4	Save Game	27
3.2.5	Preferences.....	28
3.2.6	High Scores.....	29
3.2.7	Help.....	29
3.2.8	About	30
3.2.9	Exit <gamename>	30
3.2.10	Exit <portalname>	30
3.3	Game Turn Menu.....	31
3.4	Game Over.....	32

4.	STANDARD SCREENS	33
4.1	Alert screens	33
4.1.1	Notification alert.....	33
4.1.2	Note alert	33
4.1.3	Caution alert.....	34
4.1.4	Warning alert	34
4.1.5	Naming links used in alerts.....	34
4.1.6	Language style for errors	35
4.2	Data entry screens	36
4.3	Text entry	38
4.4	Numeric Entry.....	38
4.5	Radio Buttons	39
4.6	Check Boxes	39
4.7	Tables.....	40
5.	MISCELLANEOUS USER INTERFACE RECOMMENDATIONS.....	42
5.1	Item delimiters	42
5.2	'Alt' Tags.....	42
5.3	Axis on Grids and boards.....	43
6.	OTHER USER INTERFACE DEVELOPMENT ISSUES	45
6.1	Image formats	45
6.2	Large Image Sizes.....	45
6.3	Extra spaces in long strings of text	46
6.4	Extra line breaks when displaying on the 7110	46
6.5	Unwanted paragraph indent	47
6.6	Interoperability and cross browser compatibility	48

7. USER INTERFACE RESOURCES	49
7.1 Other reading material relating to more generic WAP applications.....	49
7.2 Other User Interface reading material.....	49
7.3 Web Sites for further User Interface Information.....	49

1. INTRODUCTION

There is no right or wrong way to design an application user interface. However there are some design strategies which yield applications which are more intuitive and ergonomic to the user. Often the difference can be dramatic and may even determine the success or failure of the software, despite the fact that it may function perfectly. Given different design alternatives it is rare to find one approach which is a 100% perfect solution. Each alternative will have different pros and cons, and the most appropriate design is the one that is most easily understood and used by the target user group. An indication of a good interface is where the user is focused on the steps and strategies of the task or game at hand, rather than the interface itself. Another test of a successful interface is whether the user enjoys the experience of using the software, rather than trying to figure out how to use it – even if the figuring out process is deemed trivial by the developers.

The following guidelines, wherever possible, build upon the existing Nokia phone software interface. They also incorporate many of the successful methodologies for using computer-based interfaces and successful web navigation strategies. There are many subtle yet profound mechanisms embodied in this implementation structure which are touched on in section two. Rather than go into great detail about them here, there is a wealth of information covered by other publications as listed in section seven. These Human Interface publications are usually very good at communicating cognitive concepts through broadly applicable philosophies. However, it is common for an experienced developer to think to themselves ‘of course’ on every point, but then go and create products that do not put these points into practice. These concepts are often understood when browsing such publications, yet there is a wide chasm between intellectually understanding the issue, and actually implementing them in a design. This document covers many of the foundations of designing interfaces for the MEP, and presents an interface template which puts them into practice. This allows any developer to be consistent with this optimized usability and interoperability structure simply by following the guidelines and copying the typical game structure in section three.

Where ever possible the guidelines have developed to avoid any unique interface implementations by certain manufacturers, including Nokia. With such a varied range of WAP browsers these guidelines embody designs a common denominator of tested implementations. Another goal has been to follow the ‘just in time’ design methodology, where the goal is for the user to get meaningful results from a single initial use, rather than having to take multiple sessions to learn the interface. Consider a public kiosk interface as opposed to a complex computer application. In the former case, explicitly obvious design and labeling is used, rather than condensed symbols more suitable to complex tools learned by skilled operators over long periods of time. The result is a hybrid graphical user interface and data navigation interface, which is designed for highest possible usability with consistency across the hosting site, applications and WAP browsers.

1.1 Nokia MEP version 1.0 versus versions 1.1 and 2.0

Developers for the MEP will notice differences between user interfaces in version 1.0 of the Mobile Entertainment Platform and those in this document. This document is based on the user interface developed for version 1.1 of the platform. It is recommended that in all cases developers follow and copy the examples detailed in this document, as version 1.0 of the platform will be quickly updated to version 1.1. Version 1.1 is a more comprehensive and detailed user interface standard, which will remain for many revisions to come. Version 2.0 is based on the design foundation set in version 1.1 and post 2.0 enhancements such as graphical interface elements will scale up congruently on this version.

2. USER INTERFACE GUIDELINES

2.1 Game name

The name of your game should only be comprised of English and European alphanumeric characters. Symbols such as !@#\$\$%^&<> should NOT be used, and the length of the name should be limited to a maximum of 16 characters, to avoid line wrapping on some browsers.

2.2 Screen names (card titles)

2.2.1 Give every screen an informative and unique title

To a user the screen title is the most important cue in a navigable system. Each screen is a static location to the user. Note that a screen is usually a single card in a deck with a unique title, or in the case of a single animated screen, it is actually multiple cards in a deck all with the same title (and 'paged' with a timer).

2.2.2 Why is the title so important?

It is what defines the screen as a tangible location to the user.

When people navigate around in the real world without signposts, it takes many repetitions for people to create an accurate mental map from just visual cues alone. Even when they do, senses of distance become dilated, and communicating a journey to someone else becomes difficult because people use different cues to mark the territory to themselves.

As soon as road signs or milestones are introduced, people create more accurate mental maps which bind together their sensory navigational experiences. They create these maps far more quickly when places, destinations and distances are marked with text labels. In addition, an individual's internal mental map can be communicated to another person far more effectively.

This is as true in software design as it is in other forms of human navigation and communication.

2.3 Text

2.3.1 Text is the most important element of the WAP user interface.

This is especially the case with WAP development but is also true with richer graphical interfaces. A text label immediately tells the user what a particular control is for. There is far less interpretation than with some graphical icon. The number of universally recognized graphical metaphors is very small and generally can't be 'squeezed' to fit usage in a specific application. The importance of text is compounded with applications that are used 'just in time' without prior training or familiarization. Graphical interface elements usually have to be learned over a period of time and are more suited to palettes in applications which are used day in and day out by experienced users. When graphical icons are used it should be in conjunction with a text label. Using both together creates a powerful interface representation, many times more useable than just iconic graphics.

2.3.2 Avoid Ambiguities and Jargon

Text as with graphics, should use the most generally understood terminology, and avoid ambiguity, jargon and technical phrases (especially computer terms) wherever possible. The potential range of WAP users is a much larger userbase than that of computers and the web. It is a different marketplace which includes people who are technology phobic, and who consciously shun computers and the Internet. Market research indicate that such concepts as 'Internet on the phone' immediately excludes a considerable number of potential users who just want games and services on their phones.

As software developers it is very easy to lose touch with what is computer jargon and what isn't. Such words as 'refresh', 'login', 'default' have little meaning and can be misleading, even confusing to non computer users. A quick test is to look the word up in a non-technical dictionary. If a general meaning is not listed, then use an alternative metaphor over the more technically accurate term. For the words listed above, more generally meaningful terms would be: 'update' instead of 'refresh', 'sign-in' instead of 'login' and 'standard' instead of 'default'.

2.3.3 Avoid abbreviations

Many years of user testing has shown that abbreviations, no matter how obvious, are often misinterpreted. Even universal examples such as "Max." (maximum) "Prev." (previous) are frequently misunderstood when presented in a new interface to a user. Avoid abbreviations.

2.4 Text Formatting

2.4.1 Text formatting is also an important interface design tool.

Different typefaces (fonts), positioning of headings, the number of characters that can be displayed on a line, the ‘chunking’ of concepts in sentences and paragraphs, all dramatically effect readability and retention of communicated concepts. Note also that the formatting of text also communicates some universally understood meanings on a subconscious level.

We will ignore the fact that early versions of the Nokia 7110 don’t support centered text, as more recent Nokia and 3rd party devices do support it. Use of WML formatted with styles unsupported by the 7110 will not ‘break’, it will just be displayed in plain, left justified text. On other supporting devices, it will be displayed correctly.

2.4.2 Left justify sentences and paragraphs of text

Left justified text is conventionally used for sections of readable text (also known by designers as ‘body copy’), and vertical columns of text items. Sections of text formatted in this fashion have been found to be many times more readable than other text alignments.

2.4.3 Center justify navigational links

Centered text is conventionally used in specific instances: headings, restaurant menus, invitations and prize or winner lists. When used in a menu it implies a list of alternatives. This stylistic mechanism is consciously used in the MEP navigational model, with all navigational links being centered.

2.4.4 Center justify and italicize headings

The name of a screen (defined as the card title) is presented by the browser as centered text. Likewise any headings or prompt strings within the screen area are centered and formatted as italic.



Note the centered and italicized prompt string – “Enter a meeting room.”

2.4.5 Define all text as 'small'

Many studies have been undertaken on the best number of characters displayed in a column width. It depends on the subject matter and presentation mechanism, ranging from books, to magazines, newspapers, computer screens, PDAs and now mobile phones. A good compromise for shorter sections of text is 40-80 characters per line, as found in Newspaper columns. This allows related words to be linked together on one line to form meaningful strings of information which can be read and interpreted in a glance. This of course can not be achieved on most WAP browsers which are limited to 14-18 characters in the width of a screen. The situation is made worse with the default font size in WAP browsers being quite large, limiting the characters per line and also spacing each line vertically so that there are fewer lines per screen. Yes, the screen can be scrolled to get more text, but it is bad enough to break meaningful strings of information over more than one line, and even worse to break them over more than one screen. Readability is reduced dramatically.

Looking at devices which support multiple font sizes such as the Nokia WAP Toolkit and the Ericsson, the smaller character size still has good character formation and legibility. Using the smaller font size actually enhances readability, by increasing the number of characters displayed per line and screen. In turn it allows greater flexibility in language use and interface design. Coincidentally text formatted with the small tag in the Nokia emulator, is actually closer to the single font size found in the 7110. Based on all current devices the use of small text is recommended in almost all instances of WAP based MEP interface design. Even if the device you are testing with only supports one font size, make sure to define the text as small, for those devices that do support multiple sizes.

2.4.6 Condense sentences used for links and labels

When presenting blocks of readable text, the text should use conventional language rules. However, links should not be considered as sentences and don't have to contain a noun and a verb. Think of them more as labels to a navigable place or interface action. As such they should also not have a 'full stop' or period character terminating the link text.

In order to further maximize the text space for labeling interface constructs, it is recommended that a condensed form of language is used to label interface elements and actions. An example would be to take "Exit from Chess" and condense it to "Exit Chess"

Whilst this is not a correct use of language, it is readily accepted by users and actually enhances usability, especially on small restricted devices. The condensed strings communicate the essence of the concept and subconsciously hint at an interface element to be interacted with and not simply a sentence of text to be read. Try to keep interface links as short as possible, and definitely avoid having them wrap over one line in length.

2.4.7 Capitalize nouns and verbs in links

Specific menus and screens within the MEP and games are to be treated as named places, and as such the first letter of each noun and verb in the name should be capitalized. This style is known as ‘title caps’ in the world of print design. When these names appear at the top of a screen, or appear in links, they should have this capitalization. E.g. “Select from Player List”

Depending on the language, there are also some additional capitalization styles for links which are based on web and computer interface conventions.

2.4.7.1 English link capitalization

Capitalize all nouns and verbs, but not connecting words such as ‘of’, ‘the’, ‘and’. This style further strengthens that the text of a link is not a sentence of text, but a title of a single interface element to be interacted with. An example would be “Edit Player Groups”. Note that this stylistic convention has been used for some time, for exactly the same reason, to label menu items in graphical interfaces used on Macintosh and Windows. There are also many instances of this in web design.

Note that if menus and lists of links do not use these last two styles (language truncation and capitalization), the text looks like a singular slab of text filling the menu or screen, rather than discreet select-able items.

2.4.7.2 Scandinavian link capitalization

For Scandinavian interfaces, only capitalize the first letter of the first word in links or titles and proper nouns. If the proper noun contains more than one words then capitalize the first letter of each word. Keep the leading letters of verbs and other words in lower case. Note again that in the context of the MEP, locations such as “Games Menu” and “Player List” are considered to be proper nouns and should be capitalized.

2.4.7.3 German link capitalization

For German interfaces, capitalize the first letter of the first word in a link or title, and all nouns, but not verbs or ‘connecting’ words.

2.4.8 US English Spelling

English language versions 1.0 and 1.1 of the MEP use US English spelling throughout. Support for International English, and any other language translation will be provided in version 2.0.

2.4.9 Avoid blank horizontal lines

Where there is less content in a card than will fill the visible screen, the use of a blank horizontal line for formatting content is acceptable. However when there is substantial card content, which takes up more room than the viewable area of a single screen, then avoid using blank horizontal lines. User tests revealed that if one of these blank lines aligned to the bottom of the visible screen region, it created the illusion that there was no more information to be displayed. Some users didn't think to scroll any further and waited for something to happen, remained 'stuck', until guided by someone who knew. For users unfamiliar with the scroll bar graphic on the right of the screen, the abstract scroll bar symbol was not an obvious enough to imply that there was more content. The blank line appeared to be a stronger cue.

If separating various vertical regions of the user interface is critical, then try using a centered row of period ('full-stop') characters, rather than a blank line.

2.5 Navigational Menus and Links

2.5.1 Center navigational links

If the main objective is to present a group of navigational links to different destinations, then this a special type of screen – a navigational menu. It is special because the screen is an interactive 'milestone' and forms an important junction in the navigational model.

As mentioned previously, these screens are being presented as locations or objects with descriptive names. The style used to present the listed destination links, is one link per line, centered and in an appropriate order.



A typical Navigation Menu (Showing links to different destinations)

Depending on the use of the navigation menu, your decision on the order of navigational links may be as simple as an alphabetical list (if appropriate), or preferably in the order of importance to the user. If there is a sequential order of use, then list the links in this order. Additionally order of links from top to bottom can be influenced by the users expected frequency of use.

Note that in the MEP, that it is a conscious design that selecting the top link in all navigational menus is the quickest path, with least decisions, into playing any selected game.

i.e. From the entry this is the path of using the topmost navigation menu links, all the way into a game:

Enter : Games Menu : <Game> : New Game : Queue to Play Anyone : <Game Turn Menu>

2.5.2 Links navigating in a reverse direction are listed last

At the bottom of all screens, the last link is the link which navigates to the previous screen in the navigational hierarchy. This repeating spatial cue is consistent throughout the MEP.

Another general style for navigational links is to not tag “Menu” onto links navigating forward, but only on links navigating backwards. The single exception to this is the core “Games Menu”.

e.g. The link navigating forward to the screen called “Beginners Room Menu” is titled “Beginners Room”. The link navigating back from the message board back into the “Beginners Room Menu” screen is titled “Beginners Room Menu”

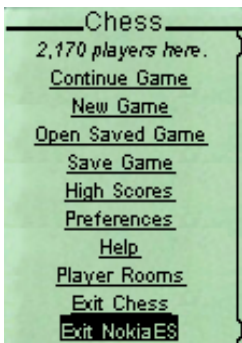
2.6 Menus within an application or game

2.6.1 The main menu of the game is structured to follow a consistent standard

The main structural menus within an application or game usually list different operations or actions for the user as well as navigable locations. These menus will have a style very similar to Navigational menus, with links and prompt strings being centered. The important distinction for these core game screens is that the order of elements on the screens as closely as possible reflect the typical order of usage in the game, as opposed to an alphabetical or other order.

For example the main menu of a game is the first level of the game from a users point of view. It lists the most global operations of the game. The order of the menu lists ‘New Game’ at the top, followed by Save Game, followed by less important Preferences, then a less frequently used Help link followed by an Exit link. This structure implies which operations should be undertaken first, and which last. Please copy this

main menu structure in your application, and understand the methodology used to derive this structure as it should be applied to all other menus in your application.



The typical layout of a game's 'main' menu.

2.6.2 Other menus display menu items in the order of usage (and frequency of usage)

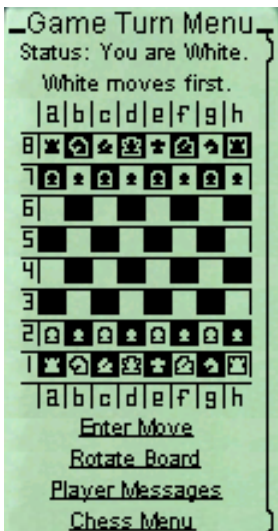
Past the main menu of the game, is the 'core' of the game, which may have any appropriate navigational structure that the developer requires. Regardless of the navigational road map of screens used in a game, the content of each screen should follow a simple structure, ordered with the first sequential step the user would take, through to the last functional step. This creates a consistent structure, where the user knows to start at the top of any screen, work through the options, and exit using the link at the bottom. This structure is used throughout the MEP and should follow through in all games.

Note that if there is no strict sequence of use, then order the items in a rough sequence of use, and weight the order towards the more frequently used to the least frequently used.

A typical example for simple turn based games:

The next level into the game from the main menu, is the Game Turn Menu. This is the game itself - the core menu of game play. Once the game has started, the user will use this menu to play the game, with specific links taking the user to screens with relevant decisions, then back to the Game Turn Menu for the next step or turn of the game. Exiting the Game Turn Menu will of course take the user out of the context of playing the game, and return the user to the games main menu, to more global functions one step back from actually playing.

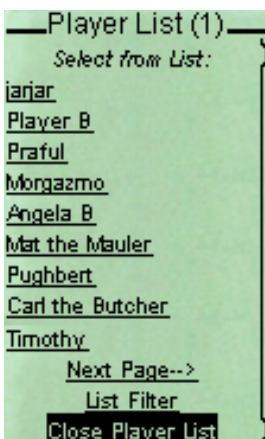
The Game Turn Menu, as with all menus of game operations, should also have all game operations listed in the typical order that the user would need them. E.g. In Chess the most important information is whose turn it is, followed by the game board, followed by a 'make a move' mechanism, followed by other optional actions, followed by a Chess Menu link which takes the user out of the turn-by-turn play, to the games main menu.



An example 'Game Turn Menu' from Chess

2.6.3 Lists of links representing objects or items, are left justified

Left justify lists of links representing listings of items or objects (nouns) of the same type. But links of navigable destinations, game controls or actions (verbs) – should remain centered.



E.g. A screen with centered navigation and control links, and left justified links for listed objects and items. Each of these player names link to a Player Details screen expanding on the details of that player, and navigation goes no further. These links of objects or nouns encapsulate the properties of the objects,

in contrast to game functions or navigation selections. This is why in the MEP, lists of players are left justified, while the Games Menu is centered.

E.g. Another example would be a phone directory application. Links to each section would be centered, but the list of individual names which would be left justified

2.6.4 Long lists of items

In the space of the average phone screen, more than ten items in a list is a long list. There are some ergonomic reasons for this breakpoint of ten listed items which is covered in the next section. When there are lists of more than ten items it is recommended to break the list into multiple pages of items. The name of the links used to navigate these multiple pages is “Next Page- ->” and “<- -Previous Page”. The right pointing arrow is made from two dash symbols “-“ and a greater than symbol “>”. The left pointing arrow is made from a less than symbol “<” and two dash symbols “-“. The order in which the two links appear is also important – Next first and Previous last. E.g.

Next Page- ->

In WML: Next Page-->

<- -Previous Page

In WML: <--Previous Page

This design conforms with the navigational spatiality in the MEP and has been implemented throughout for long lists. It is recommended that 3rd party applications are also consistent with this convention.

2.6.5 Empty lists

In the situation where a screen that normally lists items, but has no items to list, there is an interface convention used in the NMES which is recommended for applications.

The most generic message to display instead of an empty list is:

- none -

i.e. dash space none space dash

Note that whilst the list is left justified, this message should be centered.

More specific messages are also a good idea, especially when it is helpful to indicate what should have been listed and why it isn't being listed. An example from the Player List:

- no players listed -

i.e. dash space no players listed space dash

(try different filter settings)

2.7 The navigational model

2.7.1 Left to right spatiality

There is a spatial representation of time and progress which the majority of people subconsciously understand. That is, the past or origin is to the left, and the future or the destination is to the right.

Past - Future
Previous - Next

This subconscious symbolism seems to be learned early and crosses racial boundaries even for those who use text which ‘travels’ the other way – Asian text reading from right to left.

This spatiality is embodied in design of the MEP and should also be reflected in applications developed for it.

2.7.2 Top to bottom spatiality

There is another subconsciously understood spatial mapping – top is first, bottom is last.

Top
|
Bottom

First
|
Last

Shallow
|
Deep

As mentioned previously, this top-to-bottom spatial layout is used to dictate the order of steps in using a screen, from first down to last. An applications usability will be strengthened and made consistent with the MEP by following this structure.

2.7.3 Limit the use of 'back' terminology

The use of ‘Back’ interface terminology arrived with the first world wide web browsers and is geared towards the web page navigation metaphor. User testing of generic ‘Back’ terminology in custom multimedia interfaces prior to the world wide web, did not yield ideal results. However because web browsers are a learned application type, with the sole function of navigating through mostly static web content, the consistent use of ‘Back’ as a generic control works well.

If your design follows this metaphor, with typically static WML pages delivering web page like content, to an audience familiar with the web, then using 'Back' is definitely worth considering. (Noting that in these cases the use of 'Back' should be consistent with the other interface requirements in this document.) The MEP provides frameworks and services to create applications which are dynamic and can contain customized internal navigation interfaces, will be perceived by the user as an application rather than a web page style presentation, and that will eventually scale up to be more discreet Java based 'applets'. It is for these reasons that the design of the MEP interface consciously moves away from specific web based metaphor, and the generic use of back on each and every screen is not recommended. Instead there are alternatives as listed below. Where 'Back' is recommended, it is with certain conditions.

2.7.3.1 Don't use 'Back' when the destination is the NokiaES Menu (the <portalname> Menu)

The main menu of the MEP is labeled in this document as the NokiaES Menu or the <portalname> Menu. NokiaES Menu is the default name, which will be different on most installations of the MEP, as this name is customized by the operator hosting the service. As such the name of this menu varies greatly and is often up to 13 characters. This leaves no room for a "Back to" to prefix the link, without the link wrapping over one line, a situation which should be avoided at all costs.

2.7.3.2 Use 'Exit' when leaving a metaphor of a physical space

In the MEP there are some interface metaphors to physical 'spaces'. These are: the MEP itself, ('Enter' and 'Exit NokiaES'), each game (hence 'Exit Chess') and the Player Rooms within games ('Exit to Rooms Menu').

To keep this metaphor consistent, the use of the 'Exit' terminology is suggested when relating to metaphors of a physical space in your application e.g. 'Exit Maze', 'Exit Cantina' Another variation on this, where it is important to communicate the destination, is to state where the user is exiting to e.g. 'Exit to Rooms Menu'

2.7.3.3 Don't use 'Back' when the screen opened is an error or Alert

Use the standard Alert screen terminology as outlined later in Section 4 of this document.

2.7.3.4 Don't use 'Back' when the screen contains settings/state which is canceled by going 'back'

There are many examples of this such as Preference screens, configuration screens, the Accepted Invitation waiting screens etc. In these instances follow the 'Cancel' and 'OK'/'<Action verb>' style which is used for Alerts, and other settings style dialogs.

2.7.3.5 Don't use 'Back' when the screen opened is a notification in the normal flow of use

Use "Continue" instead of 'Back' even if the screen navigated to is the same either way. This difference is subtle yet important, as it creates a more rewarding, forward-moving feeling to the user, and avoids perceived 'end nodes'.

2.7.3.6 Use 'Back' to conceptually separate the link from forward navigating links

If the link which goes to the previous screen does not fall into any of the above categories, and is at the end of a number of forward navigating links, and is hard to differentiate from the forward navigating links, then it is a candidate to be prefixed with "Back to"



This is a good example of where prefixing the link going to the previous screen with 'Back' is a good idea for navigational clarity.



Here is the amended screen, which is a good example of the use of 'back'. In these instances always make sure to keep the Back link as the last item in a menu.

2.7.3.7 Use 'Back' in large complex navigational structures

Where the navigation structure of the game application is large, complex or lacks a structure using key milestone menus, then use 'back' to mark reverse navigation links. Again always make sure to keep the Back link as the last item in a menu.

2.7.3.8 When using 'Back,' make sure to specify the destination

If using 'Back', always specify the destination e.g. 'Back to Utilities Menu' – never use 'Back' by itself. If the link name wraps over a line in length, abbreviate the menu link if possible, in this case to "Back to Utilities" If it is impossible to abbreviate in this manner, revert to just listing the name of the menu "Utilities Menu" dropping the "Back to" altogether.

2.7.4 The breadth and depth of the navigational model (and the number of items in a list)

There are two distinct axes in our navigational hierarchy:

- *The number of links listed down in a screen
(Navigational Menus which take the user to different screens)
- *The number of lateral screens across the hierarchy
(that a user must navigate through to reach a specific location)

Ideally it should take the user as few navigational decisions as possible to get where they are going. Superficially it would be easy to create very long menus to achieve this. However there are some psychological and ergonomic factors to consider.

On some phones like the Nokia 7110, the user has a scrollable wheel to position the users vertical view in a card. It is very easy to navigate the screen view up and down. In this case there is more effort clicking links to get to a new location, than scrolling through a list of links. On other phones like the Nokia 6210, there is no scroll wheel and the screen has a smaller vertical height. To scroll up or down a line at a time on this type of device, takes a user click for every line. Scrolling through a list of ten items would take ten clicks.

So the best possible design would of course have very few total screens and very few navigational links per screen – two requirements which conflict with one another. So where is a good compromise?

Let us diverge for a bit.

It is a common first year psychology test to find out how many different items in a list the average person can remember. Without distractions the average result is 7 plus or minus 2.

So going by this rule, a good “rule-of-thumb” number of different navigable locations we can hope the average person to remember would be seven. This minimizes the number of links per screen, and button clicks for those without scroll wheels. It also maximizes the potential for people to remember all of the navigable branches at any Navigational menu, strengthening its usefulness.

However if we have a list of links which are not different navigable locations, but a list of similar items, which may change all of the time and don’t really need to be remembered by the user, then conceptually the list can be of any length. (E.g. The Player List) Our internal tests with current phone based WAP devices has favored these type of lists to be around 10-12 items in length. When using a phone without a scroll wheel, a list of up to 20 items becomes unwieldy.

2.8 Metaphors and themes

2.8.1 Using metaphors to the real world

The majority of WAP interface metaphors are currently limited to what can be presented with text. With the growing graphical capabilities of WAP, combined with increased support for tables, sophisticated graphical interfaces are emerging. Regardless of whether it is just a textual interface or a graphical one, the importance of understanding when to use metaphors is becoming increasingly important to WAP development.

Many design publications and interface experts communicate the importance of metaphors. But if the importance of metaphors is taken at face value, it can lead to the design of bad interfaces. One must understand the purpose of metaphors, when to use them, how to use them and when to avoid them.

At the simplest level an interface button may be designed which looks and behaves like a real world button. It is not really a button, it is a pixel representation of a button which can be interacted with to present different graphical states analogous to states that a real world button would have, i.e. depressed or not depressed. Since it is a metaphor to a button in the real world, the users experience with buttons in the real world creates a strong concept of how this software metaphor will operate before the user even interacts with it. This is a good case for a metaphor.

In a different example a developer has chosen to create a slide show program. They are using a cassette deck metaphor as an interface to control the slides, with fast forward buttons, rewind buttons, a play button to trigger the audio and a graphical design, similar to a tape deck. It doesn’t quite work like a tape player, as there are discreet slides, and lots of other small differences. But similarly it progresses forward and reverse, and the plays audio to go with each slide. The majority of users will be familiar with the cassette deck forward, reverse and play buttons – what a great metaphor – what a cool interface? Right?

Absolutely not.

Because it is designed as a metaphor to a real world cassette deck, the user will expect it to work exactly like a real world cassette deck. Each and every difference will be a stumbling block, where the user will take many iterations to unlearn the expected functions. Using this inappropriate metaphor will create a bad interface.

As an interface is designed; interactive elements are labeled with short descriptive text, the order of presentation mapped to the order of use, it becomes operational (at least in prototype), and has been tested on the target users. Just when it starts to come together, but doesn't yet have final graphics or layout, then it is probably a good time to think about a global metaphor to a real world design. If the functionality of your interface EXACTLY maps to a real world design, then it MIGHT be a good idea to modify your design to look and behave like the real construct. However if there are any operational differences, then avoid the temptation of using an inappropriate metaphor.

Keep in mind also that there are geographic design differences. In some parts of the world switches turn on by clicking them down, in other parts of the world the same style of switch will turn on by clicking them up. An interface element copying this style of switch will cause continual grief to users in different countries. Make sure to be sensitive to these sorts of issues.

In summary, if a small scale interface element can be mapped exactly to a real world control like a button or switch, that acts in a universal way. then it is usually a good idea to use it. On a larger scale, don't use a real world metaphor for major parts, or your complete interface design, unless the interface precisely matches the real world object.

2.8.2 Themes (also known as skins)

Rather than using interface metaphors, try and use themes in the interface text and graphics. Graphical themes are often known as 'skins' and these can be completely replaced with another set of graphics, without changing the operational interface.

The same principle applies to text based interfaces. Let us look at a list of actions in both a space game and a fantasy game.

Alien Bomber Menu

- New Mission
- Show Radar
- Use Medikit
- Enter HyperSpace
- Radio for Support

Battle Lord Menu

- New Campaign

Vision of the Oracle
Touch the Healing Stone
Create and Use Portal
Summon a Helper

Each item on the two menus has exactly the same function, but the theme for each is different. If done well, following an appropriate theme throughout the application can create a richer experience. When creating such a theme using linguistic metaphors (not to be confused with interface metaphors as discussed above), and you cannot find the right words for the theme of the interface, then don't force it, and use neutral terminology instead. Remember that the goal is to create an interface that is useable, so don't go overboard in creating a theme that makes it worse.

2.9 Help

There are many lessons to be learned from over a century of industrial design. One of the most often quoted phrases is 'form follows function' Design your interfaces so that the design implies how they are to be used. Structure the application so that the major steps in using it are discreet ordered sections of the software. If a step is important, dedicate space and layout to it. If something is less important then present it as less important. If it acts like a button, make it look like a button. You shouldn't need an instruction manual to open a door. But one can easily design a door so badly, that it requires a manual.

The same goes for software.

A well designed interface should not require help. In fact if users require help then treat this as a large flashing warning sign that your interface is designed badly, and needs to be fixed.

That said, help is not a bad thing, but it should merely be supporting your design, giving detail for which there is no room in the software, providing an overview of usage, or explaining presupposed rules or goals (like the rules of chess).

Usability research has shown that the most effective way of writing help is to use a goal driven language style. This style puts the users goal at the beginning of the description, followed by the steps to achieve it.

Don't do this

e.g. Enter the coordinates, and click the 'Fire' link to attack the ship.

e.g. Going to the main menu and selecting the 'Save Game' link, will save your game.

Do this instead

e.g. Attack the ship, by entering the coordinates and selecting the 'Fire' link.

e.g. Save your progress, by going to the main menu and selecting the 'Save Game' link

2.10 Avoid mapping interface options to the hardware phone buttons

Apart from the scrolling and select link buttons there are up to three other buttons supported in the WAP specification. These usually map to text labels at the bottom right and left of the screen, using the 'options' or 'do' WAP features, and are activated by pressing physical buttons on the phone. However our interoperability and interface testing has yielded the recommendation to avoid relying on, or even using these in your interface design.

Firstly these hardware buttons are not available on some phones (e.g. early Ericssons), which is very important if backwards compatibility is important to the application. In addition it has been found that on phones that do support these, only very small percentage of users find features placed here. If a design has to use actions mapped to the optional hardware buttons, make sure the functions are global to the application, and do not change in context with each screen. Otherwise the chance of users missing the functionality is compounded many times.

2.11 Game 'Playability'

The audience for this document will include many experienced game developers who need no help in designing their games. However for newer game developers there are many books to help you get started. Whilst it is beyond the scope of this document, a couple of issues will be touched on.

2.11.1 Rewards

Avoid using unique or constantly changing graphics or interface during normal game play. Opt instead for sets of repeating graphics and subtle variations. When a user achieves an important goal or reaches a milestone, then reward them with a new graphic, or sound (when this is technically possible) or even a simple animation - especially if the player is moving to a new context of the game. This type of animated transition is known as a cut-scene to gamers. This simple dynamic can be very compelling.

2.11.2 Randomness

Another psychological device to use is random behaviors. Rather than have items, objects or characters act in exactly the same way every time, give them some variance. Doing this gives the game a natural unpredictability. Random reinforcement also has another profound effect.

When rats are administered a dose of cocaine solution *each time* a certain lever is pressed, and the cocaine is eventually is taken away, they will press it a few dozen more times, then stop pressing altogether. If

another set of rats are given the solution *randomly*, these rats will press the lever up to ten thousand times before they unlearn the behavior.

This is called intermittent reinforcement and is the most powerful form of conditioning. It is a strong psychological element in all forms of gambling. If you give the user a random rewards for some action, then you can influence the user to perform that action many, many times. For example if the user discovers that say, pressing on a wall with a crack in it randomly opens a secret door, then the user will probably end up pressing cracked walls over and over. Use this technique sparingly and wisely.

2.12 User testing

The importance of user testing through all stages of development cannot be understated. Interpreting feedback from your target market can steer your product from failure to success.

2.12.1 Start the 'software test' with a well thought out design on paper

Start with a well thought out design and test this design with users before any interface coding starts. Presenting paper mockups following a navigational model may seem clumsy, but will yield valuable feedback while you have ample time to rethink architectural issues. Present the user test very carefully. Let the users know that they are the experts and have been enlisted to test the software. Treat it as a software test not a user test.

2.12.2 Testing should be frequent with the end users being the experts

As the game progresses make sure to test each reasonable milestone with end users and allocate time for changes. Remember also that the end users are the usability experts, not you. If enough of them have usability issues, then don't dismiss this, or justify their feedback as inapplicable. Instead, understand that their feedback is critical, and may flag fundamental issues that need to be worked back into a redesign of the software.

2.12.3 The most important feedback is non-verbal

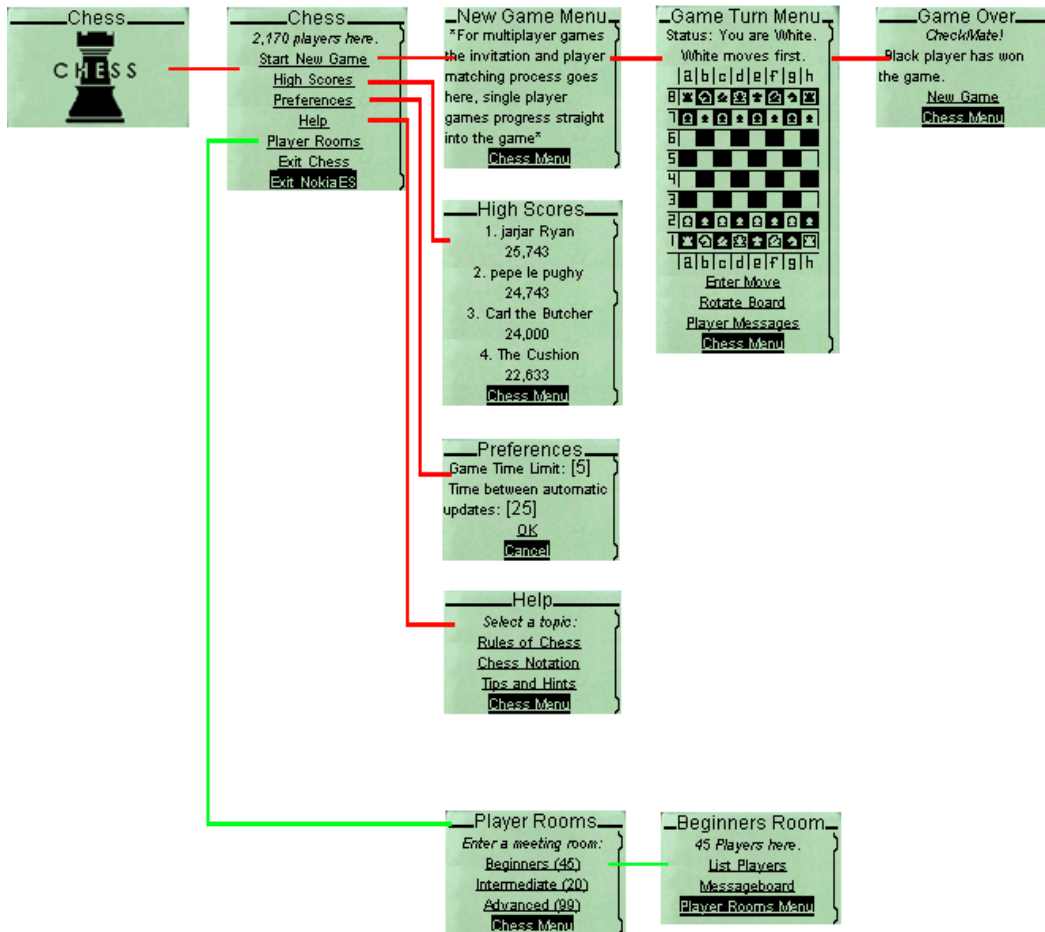
End users can consciously identify many changes and improvements and communicate these verbally. However the inverse is often true and people generally blame themselves for things that they can't understand and issues that they may not consciously identify about the interface. These 'silhouettes' of more fundamental problems are usually missed in written or verbal feedback, but can be identified by video taping the screen and user simultaneously. On reviewing such tapes a wealth of information can be

identified and it becomes obvious when users get stuck on menus or interfaces. These situations are often not communicated later, but are usually critical issues. When users are quizzed about a certain issue a common response is “No that was OK. I was stumped for a while but I figured it out” or “I am just a bit of dummy, that was just me.” - In hindsight the user didn’t see this as a problem, but the very fact that they were stumped indicates that this is a problem and is definitely a deficiency in the interface.

2.12.4 Interpret the feedback and redesign accordingly

After collecting all the user feedback, resist the temptation to make immediate changes that the user explicitly asks for. One quick fix in one spot may unravel the rest of the interface. Instead interpret all of the feedback, especially the non-verbal issues that you have videotaped. Look for patterns and get to the root of these problems. Once the issues are understood, and the impact over other parts of the interface is determined, work these back into a redesign, so that the interface incorporates the all of the changes throughout as a complete holistic expression.

3. THE TYPICAL STRUCTURE OF A SAMPLE GAME



A simplified sample game structure.

Many simple turn based games will probably look very much like this sample game. Other more complex games will have different, more divergent structures, whilst keeping the layout within each screen conforming to the guidelines in this document. The games main menu and certain other aspects of the interface are required to be consistent with the example outlined here, and this consistency will be evaluated during certification prior to being shipped with the MEP. If the game is developed utilizing the services in the default Game Service Framework, then the main menu and associated screens structure is provided by default. Games not using the default framework will need to create these screens and associated logic.

The structure of a simple demonstration game follows.

3.1 The Splash Screen

This is the first screen the user is presented with after selecting a game from the Games Menu. The function of the splash screen is to present a logo or corporate branding for the game. It may also be used to present any copyright, trademark, and publisher information. To remain compatible with the largest number of WAP devices, this splash screen should be a centered WBMP no larger than 96 pixels wide by 42 pixels high. If all the relevant information cannot be presented in one screen, a second splash screen may be used (but no more than two) and will appear sequentially using a card timer.



A typical splash screen.

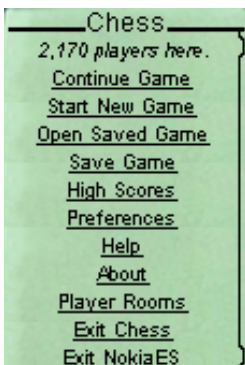
The optimal time for a splash screen is about 2-3 seconds. However different WAP browsers start the timer before the graphic is loaded, and some after. If a 2-3 second delay is used on say an Ericsson, then as soon as the card is loaded the timer starts, and the splash screen graphic starts loading. It will take about 2 seconds to load, with only a second in which to display the graphic. On a slow connection it will only appear momentarily. A 5 second delay clocks in a little long on many browsers, but will work on the lowest common denominator of WAP devices. A better solution involves identifying different browsers from header information, and using a custom timer value to compensate for these discrepancies.

3.2 The main menu of the game

After the game displays its splash screen the user is presented with the 'front' or main menu for that game. The name of this menu should be the name of the application plus 'Menu' (e.g. "Chess Menu"). To be consistent with other games in the Entertainment Service it is required that all games have a main menu which is more or less identical to what is being presented here.

For single or multi-player games:

Continue Game	(if user has a current game session open)
Start New Game	
Open Saved Game	(if the application supports saved sessions user has a saved game to open)
Save Game	(if the application supports saved sessions and there is an unsaved session)
Preferences	
High Scores	(if the game logs high scores)
Help	
About	(If an additional credits or copyright screen is required)
Player Rooms	(if the game supports meeting rooms by using the room game service)
Exit <gamename> (e.g. Chess)	
Exit <portalname> (e.g. NokiaES)	



The main menu of a sample game with a game currently open and a saved game.

3.2.1 Continue Game

*This link will only appear in the menu if the user has a game which is currently open. E.g. if the user was in the middle of a game and has navigated to this menu, the 'Continue Game' link will be visible.

*Selecting this link will return the user to the last appropriate position/screen in the game that they were playing.

3.2.2 Start New Game

*If there is a game in progress, the user is presented with a Decision alert stating “Starting a new game will discard the current game in progress.” <OK> <Cancel>



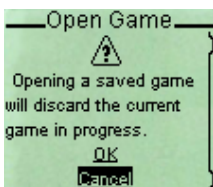
*In a single user game, when the user selects the ‘Start New Game’ link they are taken straight into the game, optionally via a player configuration screen.

*In a multi-player game, the user is taken to a New Game Menu where various player options are selected.

3.2.3 Open Saved Game

*This will only appear in the menu if there is a saved game to open.

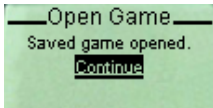
*If there is an unsaved game in progress, the user is presented with a Decision alert stating “Opening a saved game will discard the current game in progress.” <OK> <Cancel>



Depending on the game, different methods of saving are appropriate. For example, some adventure games only support one save so that the user cannot cheat by using a different saves. Some games may be optimized to support a specific number of saved position ‘slots’, some games may support any number of named slots. Some games may also automatically save and will require no user interface. All of these methods are available to developers using the game services. The user interfaces for saving will need to be created following these guidelines, (unless provided for ‘free’ with the default game services.)

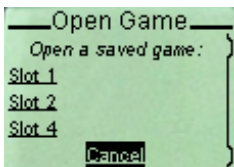
Save Method 1 – Single Save

*If the Game service is configured so that the game only has one ‘save’ then a simple Notification alert is presented.

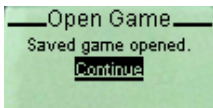


Save Method 2 – Multiple Saves in Slots

If the Game service is configured so that the game supports more than one ‘save’ then the user is presented a list of slots which are not empty and contain saved games.



When the saved game is selected from a list of multiple saved games, then the user is presented with a Notification alert with a text string stating “Saved game opened.” followed by a “Continue” link which goes straight to the last saved position in the game.



There are also other valid styles of saving that may be created for appropriate games e.g. auto loading on return to the game.

3.2.4 Save Game

*This link will be visible in the menu, if the application supports saved sessions and there is an unsaved session in progress.

Save Method 1 – Single Save

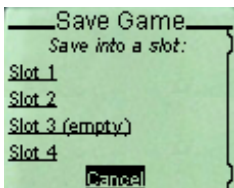
*If the Game service is configured that the game only has one ‘save’ then the users games is saved immediately and a simple notification is presented.



* Selecting this ‘OK’ link takes the user back to the game menu.

Save Method 2 – Multiple Saves in Slots

*If the Game service is configured that the game supports more than one ‘save’ then the user is presented a list of both empty slots and those already containing saved games (which can be saved over)

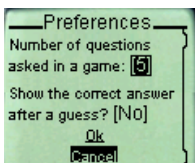


After a slot is selected, the current state of the game is saved into this slot. The user is presented with a notification screen with a text string stating “Game saved” followed by a “OK” link which goes back to the games main menu.



3.2.5 Preferences

*Most applications will have general user preferred options, and this link takes the user to the Preferences screen, where they are accessed.



*Selecting OK should save the preferences and take the user back to the game’s main menu.

*Selecting Cancel should discard all changes to the preferences since the screen was opened and take the user back to the game’s main menu.

*Unless there is some special game-specific reason, preference settings should be persistent and remain between uses of the game, and entertainment service. The persistence framework can be used to do this.

*For more complex games with many preferences it is recommended to make this a menu of links leading to screens containing sub groups of preferences.

3.2.6 High Scores

*This will only appear in the menu, if the application supports the logging of high scores. This link takes the user to a 'High Scores' screen with the top scores followed by a game menu link. The scoring of each game will probably be different, depending on the game. Only the top ten high scores are listed.



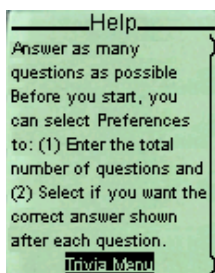
*Another variation on the content of the high scores screen is to list high scorers. If a player has have multiple scores in the top ten they are only listed once, perhaps with their highest score. This is also known as a 'hall of fame' where the emphasis is on the players rank and not the scores.

3.2.7 Help

*Many applications should be simple enough not to require help.

*It is a good idea to provide help even just to give a simple overview of the application and provide any pre-supposed rules (e.g. as with Chess).

*The last link on the screen is an <game name> menu, taking the user back to the games main menu.



*If there are multiple help topics, the Help screen will become a Help menu, listing links to each topics. Typical categories for help topics would be: How to Play, Strategies, Tips and Hints & Frequent Questions (but not its web-centric abbreviation FAQ)

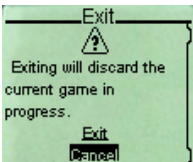
3.2.8 About

- *This optional link opens a screen which details any additional version, copyright, publisher and credits information.
- *The screen is titled “About” following by the gamename which is styled in boldface.
- *The dialog should also contain an “OK” or “<gamename> Menu” link to close the dialog and return the user to the games front menu.



3.2.9 Exit <gamename>

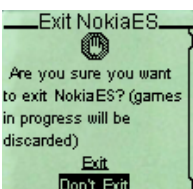
- *This exits the environment of the game application and takes the user back one level in the navigational hierarchy. <gamename> - is the name of the game, for example “Exit Chess”
- *If this link is selected and there is a game in progress, which has changed from any saved game, the user is presented with the following alert.



3.2.10 Exit <portalname>

This provides a shortcut to exit the service at the game level, avoiding the need to navigate to the main menu of the service.

- *If this link is selected the following Warning alert is presented. (see the Alerts section later for style guidelines for these alerts)



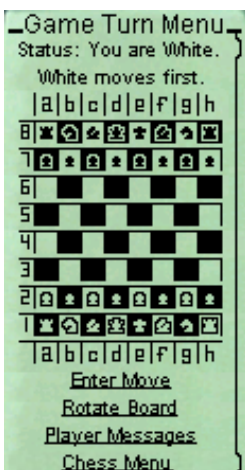
- *Selecting Exit leaves the Entertainment Service completely,

*Note that the “NokiaES” is a variable which will be different for every installation. If the game is created using the default game service, this is done for developers automatically. Otherwise refer to the developer documentation to find out how to retrieve the “shortPortalName”, then present it in the interface as outlined.

3.3 Game Turn Menu

As discussed throughout this document, the navigational structure beyond the games main menu is dependent on the game being developed. The individual content of the game takes priority and will most probably deviate from what is presented here. However, even though the ‘road map’ of screens may be very different, the content of each screen will follow the style presented through these guidelines.

In our sample game, the user starts from the games main menu and is taken to the “Game Turn Menu” This is a consistent name recommended for similar turn based games, but not necessarily for more complex or different applications.



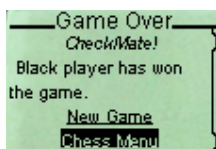
The Game Turn Menu is where the core of the game should reside. For more complex games using a Game Turn Menu try and use this screen as the main list of decisions for a user to branch out from and come back to.

Note that for our example that this is a good place for the ‘Player Messages’ link, which takes the user to a ‘in-line’ message board to chat with members of the same or opposing team. Of course the relevance of in-game messaging depends on the game, but since it is provided as part of the Game Services, game developers can use it in their games as they see fit.

*Whilst this screen contains the core of the unique content of the game, there should be a link at the bottom to take the user back to the main menu of the game. The link should be titled “<Gamename> Menu” where <Gamename> is the name of the game.

3.4 Game Over

It is interesting how people subconsciously recognize and rely on certain symbols and signs. Creating a game without the cliched 'Game Over' screen results in confused and dissatisfied user responses. The game has no sense of completion, even though it may be obvious through other cues that the game is over. The same goes for "The End" before the credits of a movie. It may seem obvious, but having the game over screen is far more emotionally satisfying signifier of the game conclusion. Hence, in almost all types of games a Game Over screen is recommended.



It is a good idea to clearly report who the winner is, with any appropriate scores or times. For simpler games, provide a link at the bottom of the screen to start a "New Game". With all games, provide a link to return to the games main menu. The link should be titled "<Gamename> Menu" where <Gamename> is the name of the game.

4. STANDARD SCREENS

4.1 Alert screens

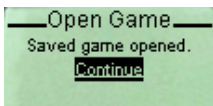
These screens are a generic presentation for notes, information, errors, special conditions, decision situations and where confirmation is required. These screens have been developed and tested to provide a standard appearance and compatibility on a wide range of browsers.

The format of the error screens is as follows:

- *Give the screen a title, which as closely as possible matches the link or action which calls the screen.
- *Apart from the plain notification alert, use one of the three alert icons, and center justify it.
These alert icons are available from the MEP 1.1 installed directories.
- *If the text string is shorter than one line, then center it, otherwise left justify it.
- *Center justify the links which acknowledge, cancel or respond to the alert condition.
- *List the links in the following order: accepting or forward moving links first (e.g. "OK"), followed by backward moving, reverting style links last (e.g. "Cancel").

There are four types of alerts:

4.1.1 Notification alert



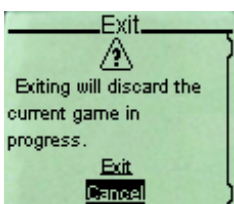
The simplest form of alert, which is presented in response to a regular operation. It simply notifies the user that a normal operation has taken place, rather than something out of the ordinary, or requiring caution, decision or warning. No alert icon is used.

4.1.2 Note alert



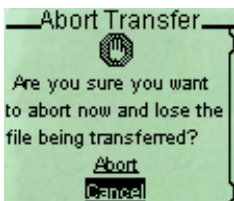
This is used to present information or conditions which are not critical, or may not be obvious from the initiating action. The alt tag to use with the small icon graphic at the top of the dialog is the text string “Note”.

4.1.3 Caution alert



This is used where a decision needs to be made and/or proceeding may lose work, progress or data. These type of alerts commonly have three buttons e.g. Save, Don't Save, Cancel. The alt tag to use with the small icon graphic at the top of the dialog is the text string “Caution”.

4.1.4 Warning alert



The stop sign, where proceeding may lose important work or data, or where a critical error or condition has been encountered. It is also to be used to indicate that no further progress can take made due to important conditions. The alt tag to use with the small icon graphic at the top of the dialog is the text string “Warning”.

4.1.5 Naming links used in alerts

*For simple alerts where there is no decision required by the user, or the alert stops a normal flow of use then name the link using the standard “OK”. If the alert interrupts a normal flow of use, which will continue after the alert is closed, then name the link “Continue”

*For alerts which present a decision, name the links as clear concise verbs.

The alert presents a description and user-actions based on the description. If the links represent a distillation of the description, then it greatly increases the chance that the user will make an informed decision. Often the user will not interpret the description properly, or at all, so verb-links spell out the goal state of the decision clearly.

*Avoid links which are neutral confirmations of decisions – ‘Yes’, ‘No’, ‘OK’

*And again avoid jargon – ‘Refresh’, ‘Purge’ etc.

Hypothetical examples of exiting a game:

Bad Example (avoid doing this)

Do you wish to save the game before exiting? <Yes> <No> <Cancel>

Bad Example (avoid doing this)

Do you wish to save the game before exiting? <OK> <No> <Cancel>

Better

Do you wish to save the game before exiting? <Save> <Discard> <Cancel>

Best

Do you wish to save the game before exiting? <Save> <Don't Save> <Cancel>

Having emphasized the importance of naming link as verbs, if there is not a widely understood verb which is appropriate don't use an obscure verb, and use an ‘OK’ instead. Again the importance of user testing can never be understated. Note the difference between the Better and Best examples. Whilst the meaning of ‘Discard’ is commonly understood as an antonym (opposite) to ‘Save’ or ‘Keep’, users tests disclosed that using ‘Don't Save’ in conjunction with ‘Save’ was clearer and needed less interpretation.

One final detail, ‘OK’ is generally written as two capital letters and not as ‘Ok’.

4.1.6 Language style for errors

The user interface should be as self explanatory as possible, but even with the best interfaces, users are bound to make mistakes. In addition the software itself will meet error conditions which are beyond the control of the user. With all errors, use one of the above alerts and make sure the language used in the error message is as informative as possible, avoiding jargon and leading the user to an appropriate solution or understanding. As with most user interfaces the user will blame themselves for misunderstandings or mistakes. An interface which ‘takes the blame’ for errors provides a far more empowering and helpful

user experience. Simply prefixing an error alert with ‘Sorry’ creates a supportive tone which removes the blame from the user.

It is recommended that all errors beyond the control of the user, and errors out of the implied scope of the interface be prefixed with “Sorry,”.

E.g. “Sorry, but your tournament application could not be processed immediately due to an error in the service.”

If error codes or strings are presented to assist with support or bug fixing, make sure to start with a soft and friendly message followed by the error code.

Don't do this:

I/O Error #48

Do this instead:

Sorry, but there has been a problem saving your details. Try again later. (I/O Error #48)”

4.2 Data entry screens

There are four standard data entry screens (a.k.a. select links or input widgets) available in every WAP device - text, numeric, radio buttons and check boxes. Whilst the WML used to call them is consistent and uniform across browsers the interface the user interacts with can be very different depending on the implementation created by that browser manufacturer.

Regardless of the variation in browsers, there are still many ways of making these more usable to the end user, which will work for all implementations.

These screens are opened using a link. To let the user know what the link will do, it should be labeled with a text string preceding the link on the card that contains them.



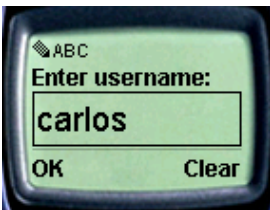
The text label is to the left, the link is the “[]” symbol on the right. The label preceding it is required to let the user know what the link is for. Note that the card based label is always followed by a colon “:” to imply its attachment to the link.

When this link is selected it opens a text entry screen that requires user input. If this input widget link is not explicitly given a title string the user is presented an empty screen like so:

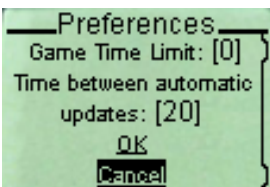


It is easy to assume that the user will immediately know what is expected of them and start entering a username. Well, user testing has shown that often they don't. When users open the input screen they often lose context to what they are supposed to do – what are they supposed to enter? Inevitably they close the screen, look at the label string, and open the link again.

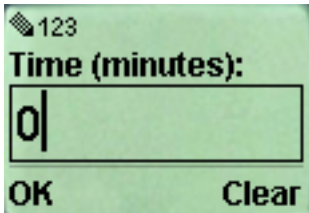
Therefore, it is always recommended that a title string be provided for all input and select widget so that the user is reminded exactly what they are expected to enter here. E.g. "Enter Username:"



Furthermore, one browser visibly displays the widget title string back on the card itself. This causes a cosmetic oddity on these browsers when both a label is created on the card and a title is provided for the text widget. Subsequent phone browsers from this company do not exhibit this behavior, so this cosmetic problem is contained. In summary provide both a card based label prefixing the widget link and a widget based title, and if possible make the title a helpful variant of the label.



E.g. Card based widget label "Game Time Limit:"



E.g. Text widget title string “Time (minutes):” Note that title is a helpful variant of card based label, in that the increment of time is specified so that the user does not confuse, say minutes with seconds.

4.3 Text entry

In WML there are many ‘input masks’ to guide the user into entering valid information e.g. Lowercase characters, range limiting etc. There is also a variant which allows for the entry of Passwords, protecting preceding characters from view by others. Make sure to use as many of these input masks as possible to assist the user to enter valid data. This can help dramatically. For example if a numeral is required then masking the input to accept numbers means that the user only has to click the number buttons once per numeral, and not multiple times to page through the different characters that the key can represent.

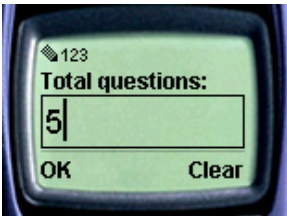
Interoperability testing found that different browsers handle the transmission of accented characters differently. Future versions of MEP will provide tested developer tools for taking entered text encoding it and transferring it to your application for correct display. As of MEP 2.0 this is not provided and it is recommended that you create your own solution for the transmission of encoded characters.

Another issue to note is that if you take user entered text and display it on a card then you will also need to ‘escape’ certain characters that are used by WML to symbolize WML structures. Not dealing with user entered characters such as “<” and “\$” may bring your WML or Java to a grinding halt.

A solution to both these issues can be created in the application itself.

4.4 Numeric Entry

This brings up just the numeric entry, avoiding text input. Input masks should be used and can be quite sophisticated, e.g. conforming data entry to say a credit card format, the number of digits etc.



4.5 Radio Buttons

This select element brings up a modal screen which allows for the selection of multiple choice data. Radio buttons are 'exclusive' meaning that only one button can be selected at a time. Selecting another radio button will deselect the existing selected button before selecting the new button.

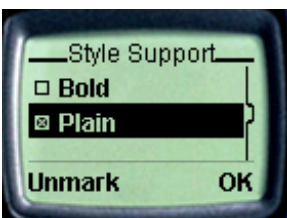
Always use a widget title and note the explicit type and value of the selections - "3wide x 4high" as opposed to say "3x4". This leaves no ambiguity to the user as to which axis is which. Verbose yes, unambiguous – definitely.



Another issue with radio buttons is that one button must always be selected. Having no buttons selected is an invalid state. A solution to the issue of not wanting any button to be selected, is to add an additional radio button named "No Setting" or "None" which caters for this requirement.

4.6 Check Boxes

This select element brings up a modal screen which allows for the selection of multiple choice data. Check boxes allow for multiple selection.

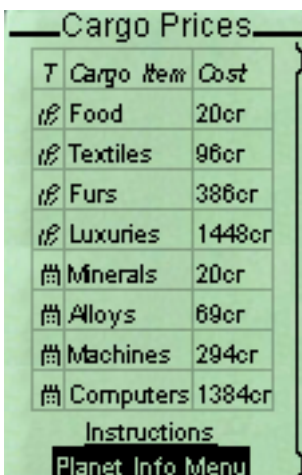


An important aspect to the use of radio buttons and check boxes is that they should only be used to select different aspects of the same collective property and not completely unrelated options. These widgets should definitely NOT be used for navigational controls, in the place of typical navigation links. This is a subtle but important distinction. Having said that, radio buttons may be used to set the aspect of navigation (e.g. North, South, East, West) followed by a link (e.g. Navigate) which actually performs the navigation.

Another way to think about it is that radio buttons and checkboxes are usually adjectives or adverbs (properties of nouns or verbs), while links are verbs (actions) or nouns (places or objects) navigated to or invoked.

4.7 Tables

More complex lists of items will require item attributes to be listed in columns. The use of WML tables is ideal for this. However the support for tables varies considerably amongst different phone browsers, so care must be taken in relying on tables for crucial parts of the interface. Also remember to test widely on different browsers as many have different limits of the dimensions of text and images which can be displayed.



The screenshot shows a WML table with the title "Cargo Prices". The table has two columns: "Cargo Item" and "Cost". The items listed are Food (20cr), Textiles (96cr), Furs (386cr), Luxuries (1448cr), Minerals (20cr), Alloys (69cr), Machines (294cr), and Computers (1384cr). Below the table, there are two buttons: "Instructions" and "Planet Info Menu".

Cargo Item	Cost
Food	20cr
Textiles	96cr
Furs	386cr
Luxuries	1448cr
Minerals	20cr
Alloys	69cr
Machines	294cr
Computers	1384cr

Here is an example of a table used purely for display purposes.

Tables can also be used as an interactive interface elements by adding links into the table. Note that most newer browsers support adding links to tables and even multiple links in a table.



Here is an example of a table used as an interactive interface element, where the table contains links.

Some browsers support images as links and even images as links in tables. This combination can give rise to sophisticated user interfaces, and is a path to the future in terms of extending what is achievable using WAP and the type of user interfaces to expect from upcoming releases of MEP. The main restrictions to such implementations are interoperability issues. Different browsers do not support all of these table variations and those that do, have limitations in the way they are supported (limits on image size etc.) So proceed carefully and test on a wide range of browsers when using tables as they may not work correctly, or at all depending on the browser. Current implementations of MEP are based on the lowest common denominator of browser functionality, which excludes the use of tables.

5. MISCELLANEOUS USER INTERFACE RECOMMENDATIONS

5.1 Item delimiters

When entering in lists of words or items, the user can delimit each item by specific delimiter characters. The MEP supports the following characters for item delimiters:

“ “ – the space
“ , ” – the comma

This is supported in some games at the moment (Wordhunter) and with each successive release of the MEP there will be increased support for these delimiters. We recommend support for these delimiters in specific games.

As the MEP evolves the requirement to support phrase delimiters will emerge. It is proposed that the MEP support the following characters for phrase delimiters:

“ ” – the comma
“ ; ” – the semicolon

5.2 'Alt' Tags

Most modern WAP browsers support alt tags. An alt tag is a text string which is displayed in place of the graphic while it being loaded by the browser. A cards text content is displayed first and the user can scroll up and down on the card before the graphics appear (which are being loaded in the background). Due to the slow speed of networks it may take some time for these graphics to appear, creating a sense of instability to the user. In some games which rely on time limits, it can give some users time advantages when playing the game.

*In general provide 'alt' tags for all graphics. These textual tags are displayed in place of the graphic while it loads, and can provide a reassurance to the user and may even allow for game play before the graphic appears.

*For generic graphics we recommend using the alt tag text "Loading..." which notifies the user that something is happening but it has not yet finished.

*Where possible provide even more informative alt tags which give a textual representation of the image. E.g. In Wordhunter, there is a table of letters displayed. These letters are graphics. Each graphic has an alt tag which is the same letter as that which the graphic depicts. This enables the user to view the table of letters and play the game even though the graphic for that letter has not yet loaded.

*For the small icon graphics used in alerts use the text “Note”, “Caution” and “Warning”

The creative and informative use of alt tags can greatly increase the user experience of an application.

Note that older versions of the Nokia 7110 wait until all of the elements in a screen have loaded before displaying the screen. No content or alt tags are displayed just a “Connecting to service” barber-pole progress bar. As such there is a longer wait for meaningful content which should be considered and tested for when developing time based applications.

5.3 Axis on Grids and boards

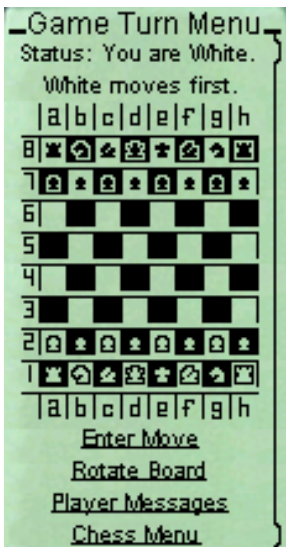
A couple of details and recommendations have emerged from user testing when developing games which use grids and boards.

*Label each axis with a different character type.

E.g. The vertical axis with alphabetical characters, and the horizontal axis with numerals. This avoids requiring the characters precedence to define the axis. When using one character type to label both axis, the coordinates (2,4) and (4,2) are completely different, and require another level of consideration by the user to define and enter coordinates. Using different character types makes things more usable as the coordinates (A,7) and (7,A) are precisely the same board position and can be entered in any order by the user.

*Label the axis on the left and bottom of the grid.

With a board which is larger than the viewable screen area and whose axis is labeled at the top, scrolling to the top of a screen to read the axis labeling, scrolling down to check the destination coordinate, then scrolling down further to enter these coordinates becomes a multi-stage feat of memory rather than a single action. User testing revealed this to be a major stumbling block in prototype board games. The solution was to label the grid axis at the left and the bottom, so that order in which a user determines the destination square, its coordinate, and enters it, becomes a more linear process. Having the grid coordinates at the bottom of the screen positions the labeling closer to the input link, so that there is less navigation between the two.



The Chess board example displays how these interface recommendations are put into practice. Note that for overkill the horizontal axis is duplicated at the top of the screen.

6. OTHER USER INTERFACE DEVELOPMENT ISSUES

6.1 Image formats

In the WAP 1.1 spec only WBMPs are supported. These special black and white forms of a BMP image file can be created or sourced via the following methods:

1.A WBMP export plug-in is available for Photoshop 5.0 and above. It is available from RCP at:
<http://www.rcp.co.uk/distributed/Downloads>

2.A BMP to WBMP converter is available as a DOS application free with the Nokia MEP SDK and is named: bmp2wbmp.exe

Note that we have found that some forms of BMP cannot be properly converted by this utility. Saving a copy as a monochrome BMP from PaintshopPro 4.0 (and above) in RGB format seems to create a format compatible with this utility.

3.Public domain WBMP site at Hicon (Note that the website is in Dutch)
<http://www.hicon.nl/>

6.2 Large Image Sizes

Images wider than 96 pixels will display, but will be cropped at the right hand side on narrow browsers like the 7110. When creating images taller than 95 pixels wide x 42 pixels in height, they will need to be constructed from rows of smaller images. An example of this is Wordhunter. In order to remain visually compatible with all types of browsers, use the following format for your images that are required to be displayed on separate lines:

Incorrect Method

(this will NOT cause line-breaks between your images when displayed on a very wide screen browser)

```
<wml>
    <card>
        
        
    </card>
</wml>
```

Incorrect Method

(this will cause an EXTRA break on versions of the Nokia 7110 prior to 4.92)

```
<wml>
    <card>
         <br/>
        
    </card>
</wml>
```

Correct Method

```
<wml>
    <card>
        <p align="center">
            
        </p>
        <p align="center">
            
        </p>
    </card>
</wml>
```

6.3 Extra spaces in long strings of text

Often when multiple screens of text are being presented, three spaces in a row are being displayed on the phone instead of one space. However when checked in a text editor the source looks fine. The problem was found to be introduced by some text editors which use different line break characters in the source files. If you are encountering this then remove the line breaks in that string in your editor, or use a different editor which has control over which line break characters are used e.g. Metroworks

6.4 Extra line breaks when displaying on the 7110

In order to avoid excessive line-feeds in your decks when displayed on older version of the Nokia phone browser software (such as the 7110 with software prior to version 5.0) the following strategy is recommended.

The <small> tag seems to be the main culprit.

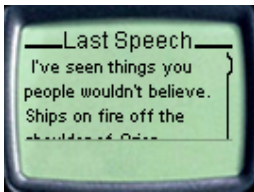
* Remove excess <small></small> tags. These produce extra line-feeds. If possible, encapsulate your whole card content in one <p><small>... ..</small></p>. If you require different formatting, e.g. centered and left-justified, you will require extra <p> tags for the alignment of each item.

*Sometimes screens have a `
` as their 1st item. Remove these! They do not show up in the Nokia WAP Toolkit, but they do show up on the actual 7110 handset.

*There is a display deficiency with widgets in many browser implementations that will not show a link next to a text string on the same line. When there is text, it pushes the link to the next line and adds an additional line feed after the link. What should have taken one line now takes three. This is a browser issue, so if this formatting can be tolerated in the short term, the problem should disappear with later browsers. A current workaround is to put the string on one line and the link on the next. The extra line-feed will not be generated.

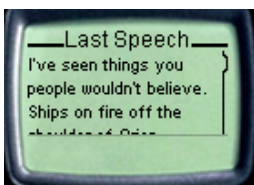
6.5 Unwanted paragraph indent

Some browsers like the Nokia WAP Toolkit automatically add paragraph indents to the text display. It is a little difficult to see the indent before “I’ve” in the example below, but in scrolling lists of text these are very noticeable and give an irregular format when these indents are not wanted.



This is an example of the WML which creates the above screen.

```
<card id="page1" title="Last Speech">
  <p align="left">
    <small>
      I've seen things you people wouldn't believe. Ships on fire off the
shoulder of Orion.
    </small>
  </p>
</card>
```



Here is the screen without the indent. The solution to getting rid of this indent is quite simple – remove any white space or source formatting between the end tag symbol “>” and the text itself – “I’ve seen...” The WML is shown below.

```
<card id="page1" title="Last Speech">
  <p align="left">
    <small>I've seen things you people wouldn't believe. Ships on fire off the
shoulder of Orion.
    </small>
  </p>
</card>
```

6.6 Interoperability and cross browser compatibility

There are many variations in the presentations of different WAP browsers. Not only are the screen dimensions, screen resolution, fonts, and leading (line spacing) different, but each browser varies in how much of the WAP feature set is implemented. Of the WAP features that are supported, each of these can be implemented in completely different and almost unrecognizable ways. Not only that, the user input mechanisms vary from handy keyboards, to touch screens, scroll wheels to simple single-press buttons.

All of the interface design and guidelines presented in this document have been developed and tested on a wide range of browsers, with the aim to give as consistent a visual presentation as possible, and provide ergonomic consideration to the various input types.

For both MEP 1.1 and 2.0 applications it is strongly recommended that these guidelines be followed to leverage off the interoperability solutions which have driven the formation of these guidelines. For 1.1 developers the guidelines provide a lowest common denominator of user presentation without having to do customized version for each browser.

With MEP 2.0 we have introduced some simple on-the-fly transformations of the WML to provide a customized presentation to certain browsers. It is assumed that the developed application adheres to these guidelines, which are used as a base point to start the transformation. For example on some Ericsson browsers, if there isn't a card title it will put the pathname of that dialog into the title region resulting in confusing and ugly titles. MEP 2.0 interoperability transforms add a dash to these dialog titles to create a clean neutral title (no a space doesn't work). At the other end of the spectrum the UP browser never displays card titles, destroying the navigational foundation of the interface. In this case the transforms take the card titles and insert these in the top of the viewable area of the card itself effectively creating titles. Another UP issue is that a card containing multiple input elements is segmented into multiple sequential screens, the MEP 2.0 transformations put these on the same screen.

All the MEP developer has to do is follow these guidelines and dialog layouts and they will have customized presentations created by the MEP, reducing the need for multiple versions. Note that this feature can be turned off by the developer is desired. More information about the interoperability transforms can be found in Developer's Guide.

7. USER INTERFACE RESOURCES

7.1 Other reading material relating to more generic WAP applications

“WAP Service Development Guidelines” Anne Kaikkonen NRC Copyright 1999 Nokia Inc.
“Developing User Friendly Applications” Copyright 1999 Phone.com
“Design Guidelines for WAP Services” Copyright 1999 Ericsson
“Nokia 7110 Style Guide for Service Developers” Copyright 1999 Nokia Inc.
“WML Reference Version 1.1” Copyright 1999 Nokia Inc.

7.2 Other User Interface reading material

“The Design of Everyday Things” – Don Norman
“Turn signals are the facial expressions of automobiles” – Don Norman
“Things that make us smart” – Don Norman
“Tog on Interface” - Bruce Tognazzini
“About Face: The Essentials of User Interface Design” – Alan Cooper
“The art of human-computer interface design” – Brenda Laurel
“Designing Web Usability” – Jakob Nielson

7.3 Web Sites for further User Interface Information

Ask Tog – Bruce Tognazzini’s home page
<http://www.asktog.com/>

UseIt.com – Jakob Nielson’s home page
<http://www.useit.com/>

Don Norman’s home page
<http://www.jnd.org/>

Human Computer Interaction Bibliography
<http://www.hcibib.org/>

Human Computer Interaction Bookshelf
<http://www.ida.liu.se/~jlo/hci.bookshelf.html>

Human-Computer Interaction Resources on the Net
<http://www.ida.liu.se/labs/aslab/groups/um/hci/>

Interface Hall of Shame

<http://www.iarchitect.com/mshame.htm>

Usable Mobile - usability forum for the 'mobile internet'

<http://www.usablemobile.com/>

Understanding USA – Information Architecture

<http://www.understandingusa.com/>