# Applications for Mobile Information Devices

*Helpful Hints for Application Developers and User Interface Designers using the Mobile Information Device Profile*

*A White Paper*


Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
1 (800) 786.7638
1.512.434.1511

Please
Recycle

# Contents

CHAPTER 1

# Introduction

Developing applications for new Internet-enabled, wireless consumer products, such as cellular phones and two-way pagers, poses an entirely new set of challenges for user interface (UI) designers and software developers. If you fall into either of these categories, and have a background in developing software for desktop computers and workstations, you may be called upon to use your skills to develop new applications and wireless services for these consumer devices. Or, your company may ask you to develop software for use on devices issued to its sales force, field service representatives, or other corporate road warriors.

To some extent, knowledge gained by working with larger, more powerful systems is transferable. But consumer devices also bring with them special requirements that you quite possibly may never have encountered in the world of workstations and desktop computers. There are a number of key issues that center on the technology as well as the users of the technology.

Because of limited screen size, memory, and processing power, wireless technology sets limits on what can be accomplished. Developing new applications for these devices will require ingenuity. Along with technology constraints, you must also respond to the expectations of your end users — consumers. Unlike workstation or desktop users who are accustomed to working with big screens, large amounts of memory, and relatively complex technology, consumers want devices that are simple, intuitive, and reliable. For them, rebooting a cell phone in the middle of a conversation or conducting an online stock transaction just isn't acceptable. Even corporate users who are highly computer literate have the same expectations of simplicity, reliability, and ease of use when it comes to their wireless devices.

# A Helping Hand

This white paper describes the new Mobile Information Device environment and some of the differences you are likely to encounter. It provides suggestions that will help you create innovative, compelling consumer products despite the technological challenges you have to overcome. You'll find a brief overview of consumer expectations for these devices; some thoughts on strategies for designing for this marketplace; and an overview of the basic concepts in the Mobile Information Device (MID) Profile including screens and abstract commands.

Designing for these consumer devices requires more than just squeezing information into a tight, little GUI. The entire device must be considered. If you incorporate some procedures and interfaces that desktop users take for granted, the technology may cause consumers to either not buy the device in the first place, or toss it into a kitchen drawer or glove compartment, never to be seen again. The goal of this paper is to help you navigate these potentially dangerous waters.

This white paper assume familiarity with the basic human interface lifecycle and associated methodologies, including user testing. If you need to brush up any of these topics, there is a short bibliography at the end of this paper that can help you get started.

CHAPTER 2

# The Consumer Market — A Whole New Ball Game

When a typical user sits down at the keyboard of a desktop computer or a workstation, he or she knows that despite the efforts of the manufacturers to make their systems user friendly, a certain level of technical competence will be necessary. Applications must be installed, Internet and other networking connectivity navigated, and inevitable crashes dealt with. Within an enterprise, PC users have access to nearly unlimited storage, GUIs that can accommodate multiple windows simultaneously, and keyboards and mice that make handling data and applications fast and easy. If all else fails, technical support is only a phone call away. A person who uses a PC at work is effectively being paid to put up with the system's idiosyncrasies — it comes with the territory.

At home it's a different story. A consumer’s level of expertise is more than likely much less than that of a power user in the corporate world. Even though over 60 percent of U.S. households now have a personal computer (and that percentage is growing), compared to computers in the workplace, these home PCs are used far less often, have fewer applications, and perform far less complex activities. Outside of North America, the penetration rate of PCs in households is considerably lower — for example, 20 percent in Japan — while a higher percentage of people own wireless devices.

When it comes to consumer devices, it's a whole new ball game. The level of the consumer's PC expertise does not really come into play. Metaphors and models borrowed from desktop environments usually don't work. Many consumers may not be familiar with such desktop staples as pop-up menus, scroll bars, drag and drop functionality, and a rich desktop GUI.

Even computer-savvy workers who know their way around a PC have different expectations when they interact with a consumer product. Psychologically, the fact that consumers spend their own money to purchase a TV, cell phone, or new game box makes them far less tolerant of products that demand that the user adapt to its mode of operation, rather than the other way around.

Whether computer literate or computer challenged, consumers expect these products to be powerful yet uncomplicated, and available for use right out of the box. The learning curve has to be instantaneous — no manuals, training sessions, or instructional videotapes should be required. And the devices must be completely predictable and reliable — no surprises or unfortunate crashes.

On the other hand, although many people may reject a device they consider too high-tech because of complex interaction models, the fact is that most consumers have already mastered a set of impressive skills associated with today's appliances. They routinely handle TV remotes and VCRs, push-button phones loaded with features, and microwave oven control panels. But the failure rate of new consumer products — more than 80 percent never make it to market — is an object lesson for UI designers and software developers venturing into this demanding market. If it's not easy to use, it's not going to make it.

## Consumer Products — A Taxonomy


**FIGURE 1** A typical cellular phone displaying a computer game screen

Unlike personal computers, consumer products such as TVs, smart pagers, or advanced cellular phones are specialized rather than general-purpose devices.

A TV is an entertainment device; a pager or a cell phone a communications device. They may have other functions, but compared to the wide-ranging capabilities of a PC, they are finely honed specialists. Success in the marketplace often depends on finding just the right mix of applications and features that fit within the boundaries of the particular device's specialized niche.

Compared to PCs, consumer products typically have less memory, smaller and lower resolution displays, and different input/output mechanisms. Because it would be impossible to cover all the various consumer devices coming to the market, this paper focuses on Mobile Information Devices (MIDs) The MID Profile includes a set of user interface APIs that are designed for wireless devices.

Following are just a few considerations that software developers or UI designers working with these devices must take into account when designing applications using the MID Profile:

- The minimum MID supported by the MID Profile must have a display that is 96 pixels wide by 54 pixels high and 1-bit in depth (black and white).
- The device must support either a “one-handed” or “two-handed” input mechanism.
  - A “one-handed keyboard” is a term commonly used to describe an ITU-T phone keypad. An ITU-T phone keypad typically has the numbers 1-9, an asterisk (*), and a pound sign (#).
  - A “two-handed keyboard” is a term commonly used to describe a QWERTY keyboard (as might be used by a pager).

Among the MIDs currently in the marketplace are cellular phones and two-way pagers. Many of the human interface concerns impacting these devices apply to other consumer devices as well.

## Some Fundamental MID Profile UI Concepts

The MID Profile UI APIs are logically composed of high-level and low-level APIs. The APIs are designed for applications or services where the MID functions as the client device. The user gains access to applications and services that run on the MID through a network service provider.


**FIGURE 2** A concert ticket application

The high-level APIs are designed for applications where software portability across a range of different MIDs is desired. This is important if you are writing an application or service that a network provider plans to deploy to a selected set of MIDs. To achieve this portability, the APIs use a high level of abstraction. The trade-off is that the high-level APIs limit the amount of control the developer has over the human interface's look and feel. This enables the underlying implementation of the UI APIs, which is done by the device manufacturer, to be adapted appropriately to the human interface on the device's hardware and native user interface style, ensuring a consistent user experience between the native applications and the MID Profile applications.

The actual drawing to the MIDs display is performed by the device implementation. The implementation also controls the visual appearance (e.g., shape, color, font, etc.) of the components and encapsulates navigation, scrolling, and other primitive interactions. The application is not aware of these interactions and cannot access concrete input devices such as individual keys. Typical applications using a high-level API are stock transactions, travel reservations, online purchase of concert tickets, news headlines, weather updates, and traffic information.


**FIGURE 3** Space Invaders game

On the other hand, the low-level API provides very little abstraction and requires a bit more design work to remain portable. It is designed for applications that need precise placement and control of graphic elements, as well as access to low-level input events. The low-level API allows the application to access special, device-specific features. Games are a typical application using this API.

CHAPTER 3

# A Matter of Screens

As mentioned earlier, the MID Profile UI API has few parallels with the desktop systems that most people are familiar with. Because of the small displays, a major difference that designers and developers must take into consideration is the absence of windows or a windowing system. Instead, the central abstraction of the MID Profile UI API is the “screen”.

A screen is an object that encapsulates device-specific graphics rendering user input. Only one screen is visible at a time, and the user can only interact with the items on that screen. The screen handles all events that occur as the user navigates on the screen. Only high-level events are passed on to the application. Screens scroll vertically only; there is no horizontal scrolling.

Screens should be simple in design, basically one user task per screen, and contain as few UI components as possible to complete the task. An application is composed of a set of screens, which the user steps through as they complete tasks. The set of screens that compose an application do not need to be linear — branching and jumping between screens to enable proper usability is expected. The application developer should design the application in such a way that navigation is clear and obvious to users. Running usability testing with actual users can help tune this area of the application human interface.

The MID Profile makes use of four types of screens: List, Alert, Text Box and Form.

- The **List** screen provides three variations: an implicit (menu) list; a single-choice list; and a multiple-choice list. The implicit list can be used to display a list of menu commands inside of the application.
- The **Alert** and **Text Box** screens should seem familiar to developers acquainted with the desktop world — with the caveat that content must adapt to the MIDs' limited resources and screen size. Alerts are much simpler and use the entire screen. Text boxes generally will not support fancy text formatting or font styles and allow only the basic text editing capabilities.

- A **Form** is a screen that contains an arbitrary mix of items including images, read-only text, editable text, date and time fields, gauges and choice groups. The form screen is where developers will be most able to use their design and development skills to create a winning application within the MID Profile constraints.

  Although there is theoretically no limit as to what you can include in a form, because of the small screen size, it is best to limit the form to a single user task. Once a form screen is designed, it may be determined that the screen is longer than can be displayed at one time. Form screens should be not longer than three to four displays in length. Forms that require extensive scrolling can overtax the performance of the device and can become unwieldy for the user to interact with. In general, any item or its subclass may be contained within a form. The implementation, not the application, handles layout, traversal, and scrolling. None of the components contained on the form screen are able to scroll independently — the entire contents scroll together vertically. When a form is present on the display, the user can interact with it and its items indefinitely, for example, traversing from item to item or scrolling.

It is the form screen capabilities, combined with the MID Profile abstract commands, that provide the most flexibility in developing compelling, competitive, portable applications. Conversely, they are also likely to cause the most headaches.

CHAPTER 4

# Abstract Commands

To ensure application portability across devices, a method is needed to insulate the developer from the actual set of buttons that may be on a device (button sets will vary significantly among devices). As mentioned earlier, MIDs will either have a one-handed ITU-T phone keypad or a two-handed QWERTY-style keyboard. Beyond that, devices may have any number of programmable (soft) buttons or no programmable buttons at all. There may be dedicated hardware buttons for Back, Help, Clear, etc., or there may no dedicated buttons at all. The Abstract command API is the layer that is used to map between the buttons on a device and the set of commands that an application developer attaches to a screen.

This ultimate mapping of commands onto the device is a prerogative of the manufacturer. Developers gain the freedom of creating portable applications, but lose control over where each specific command will map on the device. However, the policy for how buttons are mapped on a particular device is the same for all applications that run on that device. In other words, there is one policy per phone.

What is possible is to define the set of actions the user can select from each screen in the particular application. Each action is defined as a command with an associated command type, priority, and label. The type, priority, and label provide hints to the underlying device implementation that can be used to map the commands appropriately. For example, a command of type HELP might get mapped to a dedicated Help button on a phone.

When specifying the commands associated with a screen, based on the command type, priority, and label, the underlying implementation uses the abstract command APIs to determine where on the physical device those commands will be mapped. On some devices the commands may be mapped to programmable buttons that may or may not have a label associated with them. Other devices may use roller buttons, joysticks, or instead of buttons, a menu-style user interface.

Working at this high-level of abstraction enables developers to state their preferences, but offers have no guarantee that the commands will appear in a specific order or on a specific button. All that can be expected is that all the commands will show up somewhere on the device. Commands are mapped by type first, and then priority within that type. Developers should plan on testing their application on as many different target devices as possible to understand how the abstract command mapping policies of different devices will effect their application.

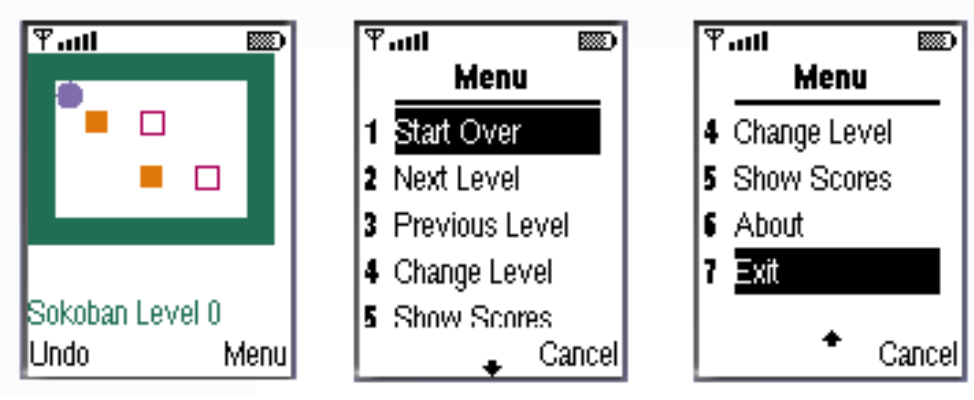


**FIGURE 1** Typical menu screens from the Sokoban game

If an application asks for more abstract commands than there are available buttons, the MID Profile reference implementation, for example, uses another UI mechanism, the menu, to make the commands accessible to the user. The overflow of abstract commands is placed in a menu and the label Menu[1] is mapped on to one of the programmable buttons. Other device implementations may use a menu mechanism or other method to handle the overflow of abstract commands.

1. This Menu is a system menu handled by the underlying implementation. This system Menu is not the same as the Implicit List, which can be used as an application menu.

CHAPTER 5

# A New World

The interaction of lists, alerts, text boxes, and form screens with high-level abstract commands provides a workable way for developers to write portable applications in the Java language that can run across a variety of MIDs. This opens the door to a variety of new wireless services and applications. For example, the device can automatically track a user's stocks and let them know when a specified stock reaches a certain price. A similar application could track auctions, alerting the user when bidding reaches a particular point. Or, a user may wish to play a game that can be downloaded and played locally; scores can then be uploaded to compare against other players.

The MID Profile has pushed cell phone functionality in wider and sometimes unexpected directions. It provides a foundation that enables developers to unleash their ingenuity and creativity in the development of applications that will not only be compelling and competitive, but will help shape the wireless world to come.

CHAPTER 6

# Bibliography

Bergman, Eric (ed.), *Information Appliances and Beyond: Interaction Design for Consumer Products;* Morgan Kaufman Publishers (2000). isbn# 1-55860-600-9.

Norman, Don, *The Invisible Computer: Why Good Products Can Fail: the Personal Computer Is So Complex and Information Appliances* Are *the Solution*, MIT Press (1998). isbn# 0262140659

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303

1 (800) 786.7638
1.512.434.1511

http://www.sun.com/consumer-embedded/

November 2000