

(19) World Intellectual Property Organization
International Bureau(43) International Publication Date
2 February 2006 (02.02.2006)

PCT

(10) International Publication Number
WO 2006/010255 A2(51) International Patent Classification: **Not classified**(21) International Application Number:
PCT/CA2005/001162

(22) International Filing Date: 26 July 2005 (26.07.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/592,129 30 July 2004 (30.07.2004) US(71) Applicant (for all designated States except US): **RE-SEARCH IN MOTION LIMITED** [CA/CA]; 295 Phillip Street, Waterloo, Ontario N2L 3W8 (CA).

(72) Inventors; and

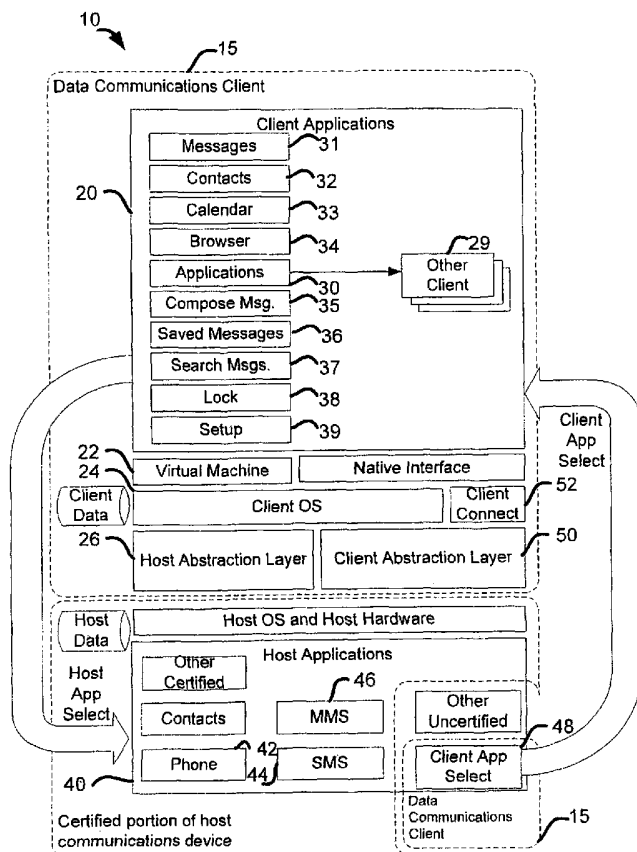
(75) Inventors/Applicants (for US only): **HODGSON, John, S.** [CA/CA]; c/o Research in Motion Limited, 450 March Road, Kanata, Ontario K2K 3K2 (CA). **PLESTID, Thomas, Leonard, Trevor** [CA/CA]; c/o Research in Motion Limited, 450 March Road, Kanata, Ontario K2K3K2 (CA). **EMERY, Jeffrey, K.** [CA/CA]; c/o Research in Motion Limited, 450 March Road, Kanata, Ontario K2K 3K2 (CA).(74) Agent: **MOFFAT & CO.**; 1200-427 Laurier Avenue West, Ottawa, Ontario K1R 7Y2 (CA).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT,

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR PROVISIONING A COMMUNICATIONS CLIENT ON A HOST DEVICE



(57) Abstract: An apparatus for provisioning a data communications client on a host communications device, the host communications device adapted to operate on a communications network, the apparatus comprising: a first data store adapted to store variant configuration information; a second data store adapted to store provisioning information; a provisioning module adapted to select the provisioning information stored in said second data store as a function of the variant configuration information stored in said first data store and apply the selected provisioning information to provision the data communications client; and a user interface interacting with said provisioning module to enable a user of the host communications device to provision the data communications client.



RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Published:

— *without international search report and to be republished upon receipt of that report*

METHOD AND APPARATUS FOR PROVISIONING A COMMUNICATIONS CLIENT ON A HOST DEVICE

FIELD OF THE APPLICATION

[0001] The present application deals with a method and apparatus for provisioning a communications client on a host device and, in particular, to a method and apparatus in which a user can provision various service types for a device without requiring a software download.

BACKGROUND

[0002] In a host wireless device, it is sometimes desirable to add a client onto the host to perform functionality that the host normally would not include. The host is typically certified with its software and hardware to communicate over a wireless network, whereas a client typically would not be. Further, certification could occur prior to the client being added, especially in the case that the client is integrated after-market onto the wireless device.

[0003] It is further desirable that the client is able to communicate with the native applications on the host and that the host applications are able to communicate with client applications. This communication preferably includes controlling a user interface on the host device from a client application, including registering inputs to the host device for the client application and displaying or outputting from the client application.

[0004] In some cases it is also desirable to be able to use device settings from the host environment in a client setting. Examples of this could include locale information, time zones, display themes or backgrounds. The automatic propagation of a change in host device setting would be preferable in some situations.

[0005] In one embodiment it is also desirable to have symbol inputs to a client correspond with symbol inputs to a host. It is further desirable that the input of symbols be simplified.

[0006] It is further desirable to be able to change the provisioning of a client directly from a host device without having to load new software onto the host device. In

particular, it is desirable to be able to select a service type from a list of service types to suit a user without having to change the device the user has, or without having to perform software changes on the user's device

SUMMARY

[0007] The present apparatus and method provide a divided architecture for integrating a client into a host wireless device. One key to the present system is that the host is recognized as the dominant determinant in a divided architecture due to the fact that device type certification efforts (Global Certification Forum (CGF)/ PCS Type Certification Review Board (PTCRB)) may happen prior to the client being integrated onto the host. This necessitates that the host and tightly-tied applications to the host remain unfettered.

[0008] The present apparatus and method provide a virtual machine that is started upon start-up of the host device and is used to run client applications. The virtual machine communicates through a client OS that would normally send client application commands and functions to host dependent features, such as hardware, software, firmware or communications networks. However, since the host dependent features are certified and controlled by the host device, the operating system instead communicates with abstraction layers. The abstraction layers have a native interface for communicating with host applications, allowing client applications to use the host dependent features by utilizing host applications.

[0009] Device setting such as locale, time zone, display themes and backgrounds can be set using a binary variable. In one mode, the client settings are adapted to automatically adjust when host device settings are changed. This change is propagated by either having a listener at the host to signal a change in device settings, or polling when a graphical interface of a client is brought to the foreground. In the other mode the client settings can be fixed at the client and changes at the host device are ignored.

[0010] A client application accesses the user interface of a host device using a host native application, a platform abstraction layer and a host independent engine communicating between the user interface and a client application. The host independent engine is platform independent and relies on the platform abstraction

layer to translate and/or map function calls. The host native application depends on the user interface and host device, and is used to control actions and updates to the user interface.

[0011] One example of an input for the host native application is the input of symbols. In a system for inputting symbols to a client where the host has a native system for inputting symbols from a host symbol table by navigating a host cursor to move between adjacent symbols displayed within a host grid and the host further has a keyboard, the keyboard can be taken advantage of to map symbols to one keystroke. A client symbol table is created conforming to the host symbol table, and a grid is made where the indicia of at least one keyboard key is associated with a symbol such that when a user actuates a key in the keyboard, the cursor jumps to the corresponding symbol.

[0012] Provisioning of the device can be accomplished from software that is already loaded onto the device. By following steps from a client application on a host device provisioning of the client can be changed. A host device user is thereby enabled to upgrade or downgrade client service, i.e. to provision the data client.

[0013] The present application therefore provides an apparatus for provisioning a data communications client on a host communications device, the host communications device adapted to operate on a communications network, the apparatus comprising: a first data store adapted to store variant configuration information; a second data store adapted to store provisioning information; a provisioning module adapted to select the provisioning information stored in said second data store as a function of the variant configuration information stored in said first data store and apply the selected provisioning information to provision the data communications client; and a user interface interacting with said provisioning module to enable a user of the host communications device to provision the data communications client.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] The present apparatus and method will be better understood with reference to the drawings in which:

FIGURE 1 is a schematic diagram of the components and dataflow according to the present apparatus and method;

FIGURE 2 is a screen-capture of a host application showing various applications that can be selected in the host environment;

FIGURE 3 is a screen-capture of a client application started from a host environment;

FIGURE 4 is a screen-capture of a client application started from the host environment in a host application;

FIGURE 5 is a schematic diagram of the components and dataflow for a user interface according to one aspect of the present system and method;

FIGURE 6 is a view of a host symbol table;

FIGURE 7 is a view of various input options on a host device;

FIGURE 8 shows a 9*6 grid in a generally QWERTY keyboard layout with symbols mapped to certain letters;

FIGURE 9 shows a 9*6 grid in a generally AZERTY keyboard layout with symbols mapped to certain letters;

FIGURE 10 shows a 9*6 grid in a generally QWERTZ keyboard layout with symbols mapped to certain letters;

FIGURE 11 is a block diagram illustrating a host mobile station;

FIGURE 12 is a flowchart showing a change in service type with more functionality in the provisioning of a client;

FIGURE 13 is a flowchart showing a change in service type with less functionality in the provisioning of a client; and

FIGURE 14 is a block diagram of the components and dataflow of **FIGURE 1** specifically showing the components for provisioning the client according to the present method and apparatus.

DETAILED DESCRIPTION OF THE DRAWINGS

[0015] The present system and method is directed to a divided architecture for a client on a host device. One example of such an arrangement would be a data-enabled cellular telephone with a data device client running on top of the host telephone environment. Other examples of clients running on host environments would, however, be known to those skilled the art and the above is not meant to limit the scope of the present method and system. The examples below will use a host that is a cellular telephone and a client that is a data device client merely for illustration purposes.

[0016] A host device will require certification prior to being released for sale and use in a given market. Examples of certification include GSF- and PCS-type certification review board (PCTRB) certifications. These certifications are for the hardware and tightly-tied applications to this hardware.

[0017] In order to include a client that has communications capabilities without having to certify the client, the integration of the client requires a divided architecture in which the phone and the tightly-tied applications to the phone remain unaltered.

[0018] One example of an architecture to accomplish this is illustrated in **Figure 1**. **Figure 1** shows a method and system for a divided architecture **10** which includes client applications **20** running on top of a virtual machine **22**.

[0019] Client applications **20** can be any application that is designed to run on a virtual machine **22**. In the example of **Figure 1**, these could include a messages application **31** for viewing messages that have been received, a contacts application **32**, which presents an address book including phone numbers, e-mail addresses or other contact information for individuals or companies, calendar application **33** for scheduling appointments and managing time, a browser application **34** for browsing the internet or other network, a compose-message application **35** to compose messages for SMS or e-mail, a save-messages application **36** to view messages that

have been saved, a search-messages application **37** to search for a particular message, a lock application **38** to lock the keyboard and screen of the mobile device, and a set-up application **39** to change the set-up configuration for client **15**. Other applications **30** could also exist as part of client applications **20** and the above-listed applications are not meant to be limiting. Further, other clients besides client **15** could exist on the host device and these other clients could have applications **29** which could be invoked from application **30**.

[0020] Virtual machine **22** is preferably started at power-up of the host device and stays running no matter what. In one preferred embodiment, the virtual machine is a JAVA virtual machine and client applications **20** are JAVA applications.

[0021] All client applications **20** use virtual machine **22** to invoke instances of objects created by client applications **20**.

[0022] A feature call such as a hardware call on system **10** from client applications **20** would normally go through client OS **24**. Client OS **24** includes a number of primitives for interacting with hardware. However, in the case that client applications **20** are built onto a host device and because the host device has acquired certification for its host dependent features such as hardware, software and firmware, it is preferable that instead of interacting with the features directly, client OS **24** interacts with a host abstraction layer **26**. Host abstraction layer **26** converts calls from client applications **20** to host calls through a native interface **28**. Native interface **28** invokes host applications **40** in order to use the host dependent features on the host device.

[0023] Because host applications invoke the features of the host device rather than client applications directly utilizing the features, the above architecture provides that client applications **20** can run on a host environment and use the features of the host device without having to re-certify. This enables the client to be added to the host device after certification, including an after-market addition to the host device.

[0024] One example of a client application using the above includes the making of a telephone call when the host device is a cellular telephone. When in the host environment this simply involves using host applications to create the telephone call where these host applications use certified hardware, firmware and software to connect through the wireless system. However, when in a client application **20**, the

above architecture requires the invoking of a host application in order to make the phone call. A client application could be an address book or contact application **32** that includes phone numbers for individuals. A user may wish to select a phone number from the address book and have the wireless device phone that person. In order to accomplish this, a user may select the phone number and select an option to phone that phone number. In this case, contact application **32** indicates through virtual machine **22** to OS **24** that it needs to make a phone call. Instead of using the host dependent feature directly from OS **24**, a notification is sent to host abstraction layer **26** which, through native interface **28** invokes the correct host application **40** to make the phone call. In the example of **Figure 1**, this would be phone application **42**. Phone application **42** then starts the phone call and the user proceeds as if the phone call was started from client application **20**.

[0025] Similarly, client application **20** could give a user the option (instead of phoning the phone number) to use a short-message service or a multi-media message service to contact the individual. In each of these cases, a different host application **40** is invoked, but this is done similarly through the host abstraction layer **26** and native interface **28**. In the example of **Figure 1**, these host applications include SMS application **44** or MMS application **46**.

[0026] An alternative example of a client application **20** could be an e-mail message that includes a phone number within it, for example, in messages application **31**. Messages application **31** could give a user the option to contact the phone number with the message. A phone-related application **42**, short-message service (SMS) application **44**, or multi-media message service (MMS) application **46** is started within host applications **40**. This is done through the client OS **24** to the host abstraction layer **26** where the request is converted with a native interface **28** for a host application **40**.

[0027] As one skilled in the art will realize, data is supplied between the applications **20** and host applications **40**. In the example above, the phone number would be supplied to host application **40** including phone application **42**, SMS application **44**, and MMS application **46**.

[0028] It is further desirable that a client application can be activated from a host application **40**. Reference is now made to **Figure 2**. **Figure 2** shows a screen capture of a host application. The host application lists a series of client applications

that can be activated. As used herein, activated can mean to both start a client application **20** or to bring an already started client application **20** to the foreground. In order to activate a client application **20**, the user scrolls to the client application that s/he desires and selects the client application. Reference is made again to **Figure 1**. When an application is selected in the host environment, a client application selection application **48** uses a set of application programming interfaces (APIs) by which the host operating system can request a client application **20** to activate.

[0029] Client application selection application **48** uses a client abstraction layer **50** to activate an application within client applications **20**. Client application selection application **48** calls a function that is translated in client abstraction layer **50**. Client abstraction layer **50** then uses virtual machine **22** to activate a client application **20**.

[0030] Client abstraction layer **50** in alternative embodiments can either inject the client OS **24** event into virtual machine **22** which causes the selected client application **20** to become active or, alternatively, performs a "reverse native call", either through client OS **24** or via client connect **52** to manipulate the native representation of some client object which causes the selected client application **20** to become active.

[0031] Client connect **52** can be used for network features for client applications. This enables, for example, client **15** to communicate using a specific protocol that was not originally supported on the host device. Client connect **52** involves a protocol stack to perform this messaging, and thereby increase and improve client functionality.

[0032] An example of the above is a client calendar application as illustrated in the screen-capture **Figure 3**, or a client e-mail application as illustrated in the screen-capture **Figure 4**. Requests for a client application to be activated are converted into function calls through native interface **28**, which, in turn, makes calls on client applications **20**. These applications **20** are then brought into the display foreground.

[0033] **Figure 3** represents calendar application **33** and **Figure 4** represents a screen-capture of the display of messages application **31**. As will be appreciated by one skilled in the art, a screen bar or other marker on the screen capture could be used to indicate that the client is in the host environment.

[0034] Alternatively, client application selection application **48** may communicate directly with virtual machine **22** in order to activate a client application **20**. This may occur, for example, in the case where client application selection application **48** knows the code or a hook to start client application **20**.

[0035] Once virtual machine **22** receives a message to activate an application, either from the client application selection application **48** directly or through client abstraction layer **50**, a client application is activated and needs to assume control of the host user interface. In order to do this, client application **20** makes a call back to client application selection application **48** indicating that client application **20** needs the user interface. Client application selection application **48** then uses host code to take over the UI and thus becomes a portal between client application **20** and the host. Client application selection application **48** adapts all of the host inputs to events for the client and takes over control of the user interface. Client application selection application **48** is an uncertified embodiment of a host native application **60** described in more detail below. As will be appreciated by one skilled in the art, other embodiments could be certified.

[0036] If the host requires control of the user interface back, client **15** is notified through client application selection application **48** of this.

[0037] It is further desirable when using a host and a client that the device settings be synchronized in certain situations. Device settings could include locale settings, time zone settings or theme settings. Locale, as described herein, includes various settings such as the language of the interface, for example, English or French or Spanish. It could also include the keyboard configuration, e.g., QWERTY, AZERTY, QWERTZ or DVORAK. Theme settings could color patterns and background images.

[0038] In setup application **39** referred in **Figure 1**, a user can choose between a mode that allows the user to use host settings for the device settings or custom settings. As one skilled in the art will appreciate, a different mode setting could be used for theme, locale and time zone, or these could be all included in one mode setting.

[0039] If the mode is set to the host settings, the device settings for the client are synchronized with the host's device settings. Any changes in the host's device settings are propagated to the client and the client's device settings are, therefore, also changed. For example, if a user changes the language from French to English in a host application **40**, this is propagated to client **15** and client applications **20** will use English.

[0040] In the case where the client display is set to mimic the host display, propagation of changes in the host display is accomplished by having a listener application monitoring the host device settings. Upon a change in the host device settings, the host listener will notify setup application **39** that a change has been made to the host device settings and this change will be reflected in the client device settings.

[0041] Alternatively, propagation of a change in the device settings could include polling every time a graphical user interface from the client takes over. This polling involves comparing the host device settings with the client device settings and thereby determining if a change has been made. If a change has been made, the client device settings are updated.

[0042] Thus if the mode is 'automatic' or 'host settings', changes in the host device settings are pushed to the client, either through a listener or by polling, as described above.

[0043] If, conversely, the mode is set to 'manual' or 'client settings', the user can update the device settings in the client and client applications will use these display setting instead of the host device settings. If the mode is set to 'client settings', changes to the host's device settings will be ignored by client applications **20**.

[0044] Reference is now made to **Figure 5**. In order to interact with a user of a host device, client applications **20** need to provide a user interface **90**. On a host device having one or more input devices such as a keypad, keyboard, roller wheel, scrollstrip, touch-pad, d-pad or other navigation device, and a screen, a user must be able to input actions for client applications **20**, and the results of the input and client applications **20** operations need to be displayed on the screen. In order to accomplish this, three components are provided within the I/O architecture of the

present system. These are a host-native application (HNA) **60**, platform abstraction layer (PAL) **80**, and a host-independent engine (HIE) **70**.

[0045] HIE **70** is a platform independent component. Since PAL **80** contains the host abstraction layer (HAL) **26** and the client abstraction layer (CAL) **50**, translation between the client and the particular host is performed in it. In a preferred embodiment PAL **80** is a C function interface.

[0046] HIE **70** includes both virtual machine **22** and client OS **24**. These are used to activate, start, or call instances of objects in client applications **20** when client **15** is object oriented, or call functions in client **15** when it is not.

[0047] HNA **60** resides beyond PAL **80**, and thus can adapt user interface **90** to conform and adapt to a particular host on behalf of client applications **20**. HNA **60** can take radically different forms depending on the design of the host application infrastructure and the user interface requirements of the host operating system. For example, it is envisaged that in alternate embodiments, a keyboard and display user interface are required, or alternatively a radically different voice-only interface can be provided. HNA **60** is responsible for creating the framework necessary to receive input and update output when a user brings a client application **20** to the foreground. In a keyboard/display embodiment, this may involve creating windows, buttons or graphics widgets of any kind. In a voice-only host embodiment, this may involve speech recognition and voice synthesis.

[0048] For input **92**, HNA **60** is responsible for passing user actions to HIE **70** through platform abstraction layer **80**. Inputs **92** can include button presses, keystrokes, stylus inputs, roller wheel motions, scrollstrip motions, touch-pad motions, d-pad motions, voice commands, accelerometer motions or other inputs that would be known to those skilled in the art. These inputs are translated and/or mapped as received from the host operating system and fed through the input function of the platform abstraction layer **80**.

[0049] For output **94**, HNA **60** may receive screen updates from HIE **70** at any time, including when client application **20** is not in the foreground. These updates must be stored and memory is used by HNA **60** to maintain a complete frame buffer copy separate from the application display area. If HNA **60** is in the foreground when receiving an update from HIE **70**, then the application display area must be updated

as well as the frame buffer so that the display on the host device reflects the screen change immediately. Whenever HNA **60** transitions into the foreground, it must update the application display area with the complete contents of the frame buffer.

[0050] Other output **94** types envisaged include audio tones, voice, and signals to actuate host-specific features, such as an offset motor or led for discrete notification or indication.

[0051] In a preferred embodiment, HNA **60** uses a framework that updates the user by simply displaying a graphic image provided by PAL **80**, and processes user actions by adapting them to be sent down as events to PAL **80**. This greatly reduces the complexity of HNA **60** thus enhancing the portability of the client to other host devices.

[0052] Reference is now made to **Figures 6-10** which illustrate a specific example of how HNA **60** adapts user actions and provides a user interface on behalf of client applications adapted to the semantics of a particular host.

[0053] First, a system for symbolic input on a particular host is shown in **Figure 6**. The system employs a particular semantic for symbolic input that the users of traditional application on the host will expect to be valid on all applications utilized on the host.

[0054] Operationally, host graphical user interface element **100** offers a 9 x 6 symbol table to a user. Cursor **102** is moved along a 9 x 4 grid in order to select a symbol. Since the number of rows of the grid is smaller than that of the table, the symbols are offered on two pages,

[0055] Referring now to **Figure 7**, the user may manipulate any number of input devices on a particular host, including a keypad **104**, a 4-directional D-pad **106**, rocker switches **108**, as well as a QWERTY keyboard provided in two portions, a left keyboard portion **110L** and a right keyboard portion **110R**. Most notably, the host semantics for symbolic input on this particular host require that cursor **102** be moved on the grid by manipulating D-pad **106** to select a particular symbol.

[0056] In this particular example, HNA 60 preserves the semantics of symbolic input on the host while adapting actions a client application user is likely to desire for symbolic input given the data-centric features of client applications 20.

[0057] Referring now to **Figure 8**, operationally, client graphical user interface element 114 offers a slightly different 9 x 6 symbol table on two pages, wherein symbols can still be selected by operation of d-pad 106, thus preserving the host symbolic input semantics. Cursor 116 is now moved however on a 9 x 3 grid instead of a 9 x 6 grid. This grid height is preferable in order to be able to map one row of each of the letter rows of the keyboard of **Figure 7** to one of the symbol rows of the grid. The width of the grid is maintained the same as in the host graphical user element 114 to allow a traditional host application user to quickly learn the layout of the symbol table while utilizing client applications, and to continue to contemporaneously support the use of the d-pad 106 for selecting a symbol.

[0058] Thus, a user is enabled to directly input one of 27 symbols using one keystroke instead of having to resort to using the d-pad 106-D, while still accepting input using d-pad 106-D.

[0059] Note that the embodiments of **Figures 8-10** deliberately do not use the right-most key of the top row of a standard keyboard, i.e. in the case of the embodiment of **Figure 8**, the key marked by the 'P' indicia in right keyboard portion 110R, since the topmost rows of the three standard keyboards shown herein contains 10 keys. This has been shown above to provide advantages and thus should not be considered a limitation. Nonetheless, it is envisaged that the techniques taught herein could be adapted to other keyboard layouts and grid sizes on a per host basis by those of skill in the art, and thus those adaptations are also within the scope of this application.

[0060] In alternate embodiments, the unused keys can remain unutilized, or they can be assigned a function to further enhance symbolic input, such as toggling between the various symbol pages. It is also envisaged that toggling between symbol pages can be accomplished by use of any one of the many other keys available on the particular keyboard available on the host keyboard.

[0061] **Figures 9 and 10** illustrate how the HNA 60 can further adapt and conform to the semantics of variants of a host. In particular, **Figure 9** shows adaptation and conformity to a host variant having an AZERTY keyboard, and **Figure 10** shows the

same in the case of a QWERTZ keyboard. Note that in each case, the indicia used in user interface element **114** conforms to the layout of the particular host keyboard, and that the input actions taken by the user are adapted to select the corresponding symbol.

[0062] To summarize the example, for input, HNA **60** adapts keystrokes by mapping each grid location on the 9 x 3 grid onto a key on the keyboard graphical user interface element, shown as QWERTY, AZERTY and QWERTZ variants in **Figures 8,9 and 10** respectively. For output, HNA **60** enhances the display by showing the indicia of a corresponding alphabetic key directly below each symbol.

[0063] Referring to the drawings, **Figure 11** is a block diagram illustrating a host mobile station including preferred embodiments of the techniques of the present application. Mobile station **1100** is preferably a two-way wireless communication device having at least voice and data communication capabilities. Mobile station **1100** preferably has the capability to communicate with other computer systems on the Internet. Depending on the exact functionality provided, the wireless device may be referred to as a data messaging device, a two-way pager, a wireless e-mail device, a cellular telephone with data messaging capabilities, a wireless Internet appliance, or a data communication device, as examples.

[0064] Where mobile station **1100** is enabled for two-way communication, it will incorporate a communication subsystem **1111**, including both a receiver **1112** and a transmitter **1114**, as well as associated components such as one or more, preferably embedded or internal, antenna elements **1116** and **1118**, local oscillators (LOs) **1113**, and a processing module such as a digital signal processor (DSP) **1120**. As will be apparent to those skilled in the field of communications, the particular design of the communication subsystem **1111** will be dependent upon the communication network in which the device is intended to operate. For example, mobile station **1100** may include a communication subsystem **1111** designed to operate within the Mobitex™ mobile communication system, the DataTAC™ mobile communication system, GPRS network, UMTS network, EDGE network or CDMA network.

[0065] Network access requirements will also vary depending upon the type of network **1119**. For example, in the Mobitex and DataTAC networks, mobile station **1100** is registered on the network using a unique identification number associated with each mobile station. In UMTS and GPRS networks, and in some CDMA

networks, however, network access is associated with a subscriber or user of mobile station **1100**. A GPRS mobile station therefore requires a subscriber identity module (SIM) card in order to operate on a GPRS network, and a RUIM in order to operate on some CDMA networks. Without a valid SIM/RUIM card, a GPRS/UMTS/CDMA mobile station may not be fully functional. Local or non-network communication functions, as well as legally required functions (if any) such as "911" emergency calling, may be available, but mobile station **1100** will be unable to carry out any other functions involving communications over the network **1100**. The SIM/RUIM interface **1144** is normally similar to a card-slot into which a SIM/RUIM card can be inserted and ejected like a diskette or PCMCIA card. The SIM/RUIM card can have approximately 64K of memory and hold many key configuration **1151**, and other information **1153** such as identification, and subscriber related information.

[0066] When required network registration or activation procedures have been completed, mobile station **1100** may send and receive communication signals over the network **1119**. Signals received by antenna **1116** through communication network **1119** are input to receiver **1112**, which may perform such common receiver functions as signal amplification, frequency down conversion, filtering, channel selection and the like, and in the example system shown in **Figure 11**, analog to digital (A/D) conversion. A/D conversion of a received signal allows more complex communication functions such as demodulation and decoding to be performed in the DSP **1120**. In a similar manner, signals to be transmitted are processed, including modulation and encoding for example, by DSP **1120** and input to transmitter **1114** for digital to analog conversion, frequency up conversion, filtering, amplification and transmission over the communication network **1119** via antenna **1118**. DSP **1120** not only processes communication signals, but also provides for receiver and transmitter control. For example, the gains applied to communication signals in receiver **1112** and transmitter **1114** may be adaptively controlled through automatic gain control algorithms implemented in DSP **1120**.

[0067] Network **1119** may further communicate with multiple systems (not shown). For example, network **1119** may communicate with both an enterprise system and a web client system in order to accommodate various clients with various service levels.

[0068] Mobile station **1100** preferably includes a microprocessor **1138** which controls the overall operation of the device. Communication functions, including at least data

and voice communications, are performed through communication subsystem **1111**. Microprocessor **1138** also interacts with further device subsystems such as the display **1122**, flash memory **1124**, random access memory (RAM) **1126**, auxiliary input/output (I/O) subsystems **1128**, serial port **1130**, keyboard **1132**, speaker **1134**, microphone **1136**, a short-range communications subsystem **1140** and any other device subsystems generally designated as **1142**.

[0069] Some of the subsystems shown in **Figure 11** perform communication-related functions, whereas other subsystems may provide "resident" or on-device functions. Notably, some subsystems, such as keyboard **1132** and display **1122**, for example, may be used for both communication-related functions, such as entering a text message for transmission over a communication network, and device-resident functions such as a calculator or task list.

[0070] Operating system software used by the microprocessor **1138** is preferably stored in a persistent store such as flash memory **1124**, which may instead be a read-only memory (ROM) or similar storage element (not shown). Those skilled in the art will appreciate that the operating system, specific device applications, or parts thereof, may be temporarily loaded into a volatile memory such as RAM **1126**. Received communication signals may also be stored in RAM **1126**.

[0071] As shown, flash memory **1124** can be segregated into different areas for both computer programs **1158** and program data storage **1150**, **1152**, **1154** and **1156**. These different storage types indicate that each program can allocate a portion of flash memory **1124** for their own data storage requirements. Microprocessor **1138**, in addition to its operating system functions, preferably enables execution of software applications on the mobile station. A predetermined set of applications that control basic operations, including at least data and voice communication applications for example, will normally be installed on mobile station **1100** during manufacturing. A preferred software application may be a personal information manager (PIM) application having the ability to organize and manage data items relating to the user of the mobile station such as, but not limited to, e-mail, calendar events, voice mails, appointments, and task items. Naturally, one or more memory stores would be available on the mobile station to facilitate storage of PIM data items. Such PIM application would preferably have the ability to send and receive data items, via the wireless network **1119**. In a preferred embodiment, the PIM data items are

seamlessly integrated, synchronized and updated, via the wireless network **1119**, with the mobile station user's corresponding data items stored or associated with a host computer system. Further applications may also be loaded onto the mobile station **1100** through the network **1119**, an auxiliary I/O subsystem **1128**, serial port **1130**, short-range communications subsystem **1140** or any other suitable subsystem **1142**, and installed by a user in the RAM **1126** or preferably a non-volatile store (not shown) for execution by the microprocessor **1138**. Such flexibility in application installation increases the functionality of the device and may provide enhanced on-device functions, communication-related functions, or both. For example, secure communication applications may enable electronic commerce functions and other such financial transactions to be performed using the mobile station **1100**.

[0072] In a data communication mode, a received signal such as a text message or web page download will be processed by the communication subsystem **1111** and input to the microprocessor **1138**, which preferably further processes the received signal for output to the display **1122**, or alternatively to an auxiliary I/O device **1128**. A user of mobile station **1100** may also compose data items such as email messages for example, using the keyboard **1132**, which is preferably a complete alphanumeric keyboard or telephone-type keypad, in conjunction with the display **1122** and possibly an auxiliary I/O device **1128**. Such composed items may then be transmitted over a communication network through the communication subsystem **1111**.

[0073] For voice communications, overall operation of mobile station **1100** is similar, except that received signals would preferably be output to a speaker **1134** and signals for transmission would be generated by a microphone **1136**. Alternative voice or audio I/O subsystems, such as a voice message recording subsystem, may also be implemented on mobile station **1100**. Although voice or audio signal output is preferably accomplished primarily through the speaker **1134**, display **1122** may also be used to provide an indication of the identity of a calling party, the duration of a voice call, or other voice call related information for example.

[0074] Serial port **1130** in **Figure 11** would normally be implemented in a personal digital assistant (PDA)-type mobile station for which synchronization with a user's desktop computer (not shown) may be desirable, but is an optional device component. Such a port **1130** would enable a user to set preferences through an external device or software application and would extend the capabilities of mobile

station **1100** by providing for information or software downloads to mobile station **1100** other than through a wireless communication network. The alternate download path may for example be used to load an encryption key onto the device through a direct and thus reliable and trusted connection to thereby enable secure device communication.

[0075] Other communications subsystems **1140**, such as a short-range communications subsystem, is a further optional component which may provide for communication between mobile station **1100** and different systems or devices, which need not necessarily be similar devices. For example, the subsystem **1140** may include an infrared device and associated circuits and components or a Bluetooth™ communication module to provide for communication with similarly enabled systems and devices.

[0076] A mobile communications device, such as a phone, is typically formed of software, firmware, and hardware adapted to provide communications services over a wireless communications network. This process of forming the relationship between the mobile communications device and the service is known in the art as provisioning. Typically a network operator provisions the mobile via a subscription to a service contract. Thus, once the mobile has been provisioned, the user of the mobile is often referred to as a subscriber.

[0077] In a voice and data network such as GSM (Global System for Mobile Communication) and GPRS (General Packet Radio System), CDMA (Code Division Multiple Access), or various other third generation networks such as EDGE (Enhanced Data rates for GSM Evolution) or UMTS (Universal Mobile Telecommunications Systems), both voice and data services may be available to mobile communications devices. Example voice services include voice calling and Short Messaging Service (SMS). Example data services include Internet browsing, email, and Multimedia Messaging Service (MMS).

[0078] Although many services may be available on a given network, only those subscribers that use mobile communications devices that have been provisioned for those services will be able to benefit from them. This may present problems for the subscriber and the network operator alike. On one hand, the subscriber may desire an existing service he does not have, i.e. an upgrade, or desire disabling a service,

i.e. a downgrade. On the other hand the operator may want to offer a new service, but may hesitate if subscribers cannot benefit from them.

[0079] One known solution is to provide an out of band communications link, such as a Universal Serial Bus, on the mobile communications device, and enable the subscriber to load new software onto the mobile via the out of band communication link using a personal computer, thus re-provisioning the device. This may be an unacceptable solution to both the subscriber and the operator as there is a significant risk that the mobile, by error, receives a wrong or incomplete load, and may require servicing. Furthermore, this solution may be unacceptable to the subscriber who does not have access to a personal computer.

[0080] However, since mobile station **1100** is a host communications device that hosts client **15**, client **15** may be provisioned directly by a user of mobile station **1110**.

[0081] Reference is now made to **Figure 14**. **Figure 14** is a version of **Figure 1** in which the provisioning aspects of the present method and apparatus are shown.

[0082] Mobile device **1410** includes a data communications client **1415** and the certified portion of a host device **1440** as outlined above in more detail with reference to mobile device **10**.

[0083] Within data communications client **1415**, applications **1420** run on virtual machine **1421**. One such application is a provisioning application. A provisioning application communicates with a configuration application **1424**, preferably through a configuration user interface **1422** in order to select provisioning information.

[0084] Mobile station **1410** includes a first data store **1450** adapted to store variant configuration information and a second data store **1455** adapted to store provisioning information. Specifically, mobile station **1410** includes various provisioning configurations within the second data store to allow the user to select the provisioning of the device. Once the provisioning the device is selected a first data store can be used to configure the host and client side of the device for the selected provisioning information.

[0085] As will be appreciated by those skilled in the art, provisioning information can be stored on the mobile device at various times. In a preferred embodiment,

however, provisioning information is stored during the manufacturing process. In this way, a mobile device with various service options can be distributed to carriers for resale without a specialized manufacturing process required for each service configuration. For example, if a manufacturer allows a provisioning in a service type A or service type B, second data store **1455** includes both service types, and the service type a client desires can be configured prior to the client being sold the device. This results in savings since a carrier does not need to allocate a number of devices that are to be manufactured using a service type A and second number of devices manufactured using a service type B since it is difficult to determine beforehand the number of such devices that will be sold. In this way, only one device can be manufactured and the provisioning done by the carrier or by the user subsequently.

[0086] If a user indicates through a configuration application **1424** that they wish to change the provisioning to different provisioning information stored in second data store **1455**, configuration application **1424** goes to first data store **1450** to determine configuration information.

[0087] The configuration information from first data store **1450** is then propagated through to applications **1420** either through a global configuration or through selected configuration as will be known to those skilled in the art.

[0088] As with mobile device **10** above, configuration application **1424** communicates through a native interface **1426** and abstraction layer **1428** to first data store **1450** and second data store **1455**.

[0089] Once provisioning information has been changed at the client side, the host side needs to be informed of the change. This can either be done through an explicit message sent to applications **1442**, through a listener **1444** checking whether there has been a provisioning change, or a flag being set in one of the data stores that is checked by host applications **1442**. In all cases, applications **1442** go to the variant configuration information in first data store **1450** and any applications that need to be modified on the host side are changed.

[0090] Data communications client **1415** may further include a clear data store **1430**. Clear data store **1430** can be used to clean the client applications of data that may not be compatible with the new service type that has been provisioned. Similarly, a

clear data store **1446** on host **1440** can be used to clear host applications **1442** of data that may not be compatible with the service type that has been provisioned.

[0091] The above examples include only a first service type and a second service type. However, this is not meant to be limiting and it would be appreciated by those skilled in the art that any number of service types can be stored on the mobile station and a user may move between these various service types.

[0092] Referring now to **Figure 12**, an exemplary upgrade of client **15** is illustrated.

Method of: User upgrade to UPGRADED client service

Pre-Condition(s):	Device is configured to allow service change Current Service Type = BASE Service Lock = NO_LOCK Service Change = NO_CHANGE
Acts and steps:	1. USER activates service configuration menu item. 2. CLIENT examines factory settings for service type 3. CLIENT displays "Contact Operator" informational dialog to indicate to the user that they must do this in order to get service. (Display 1203) 4. USER selects to proceed. 5. CLIENT displays "Verification" informational dialog to indicate user desires to upgrade and indicates the need to reinstall desktop. (Display 1204) 6. USER selects to proceed. 7. CLIENT displays "Loss of Data" question to verify the user has done a backup prior performing upgrade. (Display 1205) 8. USER selects to proceed. 9. CLIENT saves new service type and sets 'service change' flag. 10. CLIENT informs Host applications of required change to point to Client applications 11. CLIENT triggers clear of Host data. (Display 1206) 12. CLIENT informs user the device will be powered off. (Display 1207) 13. CLIENT requests Host to power off the device. 14. USER starts device. 15. HOST processes the change and clears 'service change' flag. 16. USER de-installs current Client desktop to remove base configured desktop 17. User re-installs Client desktop and selects Upgrade configuration. 18. USER connects device to Client desktop 19. Client Desktop synchronizes with wireless data server 20. Client Desktop associates identifier with user's corporate email account 21. Client Desktop downloads keys & service books 22. Client Desktop performs bulk download of data 23. CLIENT registers on network 24. USER starts receiving wireless email and calendar events.
Post Condition(s):	Provisioning recorded as complete. Current Service Type = UPGRADED Service Lock = NO_LOCK Service Change = NO_CHANGE

[0093] Reference is now made to **Figure 12**. **Figure 12** shows user interaction through the provisioning process in order to change to a different service type. In the example of **Figure 12**, the service type being changed to has more functionality than the service type presently provisioned on the device.

[0094] In step **1201** a user selects a client mode from a user interface in the host mode. This brings up a list of options that the user can perform on the client side in step **1202**. In step **1202** the user can select to perform service configuration, which moves the user to an exemplary screen as, illustrated by reference numeral **1203**. In **1203** the device asks whether a user wishes to proceed and indicates that an operator may need to be contacted. If the user decides to proceed the mobile station proceeds to step **1204** in which a verification is requested to ask the user whether they really want to provision the new service type.

[0095] If the user selects YES in step **1204** the mobile station proceeds to step **1205** in which a warning is presented indicating that all personal data may be lost in the example of **Figure 12**. As will be appreciated, information can be lost with provisioning changes. For example, if a new e-mail client is used then personal data such as address books from the old e-mail client may be lost. Other examples of data being lost would be known to those skilled in the art.

[0096] The user next proceeds to step **1206** if the user indicates that they are sure that they are willing to lose the personal data in step **1205**. In step **1206** the mobile station proceeds to change the provisioning information as indicated above with reference to **Figure 14**.

[0097] Once the provisioning is completed the mobile station in step **1208** prompts the user to power down the device in order to complete the process. Alternatively, the mobile device can simply give a warning and then power down on its own.

[0098] In step **1209** the mobile station is powered up and the new client applications or the old client applications with the new provisioning are displayed.

[0099] Referring now to **Figure 13**, an exemplary downgrade of client **15** is illustrated.

User wants to downgrade to base service

Method of:	user downgrade to base service
Pre-Condition(s):	Device is configured to allow service change Current Service Type = BASE Service Lock = NO_LOCK Service Change = NO_CHANGE
Acts and steps:	1. USER activates service configuration menu item. 2. CLIENT examines factory settings for service type. 3. CLIENT displays "Verification" informational dialog to indicate user desires to downgrade. (Display 1303) 4. USER selects to proceed. 5. CLIENT displays "Loss of Data" question to verify the user has done a backup prior performing upgrade. (Display 1304) 6. USER selects to proceed. 7. CLIENT saves current service type 8. CLIENT informs Host applications of required change to point to Host applications 9. CLIENT performs delete of all data contained in the CLIENT file system 10. CLIENT informs Host applications to perform "KillDevice" IT policy request to clear all corporate data in Host file systems. (Display 1305) 11. CLIENT sets the 'service change' flag. 12. CLIENT informs user the device will be powered off. (Display 1306) 13. CLIENT requests Host to power off the device. 14. USER starts device. 15. HOST processes the change and clears 'service change' flag. 16. USER de-installs current Client desktop to remove upgrade configured desktop 17. User re-installs Client desktop and selects base configuration.
Post Condition(s):	Provisioning recorded as complete. Current Service Type = BASE Service Lock = NO_LOCK Service Change = NO_CHANGE

[00100] Referring to **Figure 13**, if provisioning is done to provide a service type with less functionality than the previous service type, similar steps as those in **Figure 12** may be included. In step **1301** a user selects a client function on the host user interface. This brings a menu up as illustrated in step **1302** for various options that the user can perform. If the user selects to perform service configuration in step **1302** the mobile station proceeds to step **1303** in which a verification screen is presented. The verification screen indicates what the service change will entail and asks the user if they are sure they want to proceed.

[00101] If the user proceeds in step **1303** then the mobile station in step **1304** presents a warning screen that certain data may be lost. In the example of **Figure 13** in which a service type with less functionality is being provided, all data for the higher

service options in the old provisioning may be lost and the warning would display this sort of information.

[00102] If the user selects YES in step 1304 the mobile station proceeds to step 1305. In step 1305 the provisioning occurs as presented in Figure 14 above. Once the provisioning is complete the mobile station may prompt the user to power down the device when they power down the device after presenting a warning to the user in step 1306.

[00103] The above therefore presents the user with options to move between various service types defined preferably in the manufacturing process and to modify the configuration of various applications on both the host and client side through a first data store. This presents the advantage that only one mobile device needs to be created during the manufacturing process and the provisioning can be done by either the user or the operator prior to the device being sold. Further, if the user decides that the services they currently have are either insufficient or excessive, service type changes can be made through the selection of provisioning information stored in second data store.

[00104] The multiple configurations on the device therefore save inventory since the operators do not need to store various mobile stations that are manufactured to various provisioning types and further allows the use of a client on a host device where the client can be configured to a user's needs.

[00105] Thus, the host device user is enabled to upgrade or downgrade client service, i.e. to provision the data client.

[00106] The embodiments described herein are examples of structures, systems or methods having elements corresponding to elements of the techniques of this application. This written description may enable those skilled in the art to make and use embodiments having alternative elements that likewise correspond to the elements of the techniques of this application. The intended scope of the techniques of this application thus includes other structures, systems or methods that do not differ from the techniques of this application as described herein, and further includes other structures, systems or methods with insubstantial differences from the techniques of this application as described herein.

CLAIMS

We claim:

1. An apparatus for provisioning a data communications client on a host communications device, the host communications device adapted to operate on a communications network, the apparatus comprising:
 - a first data store adapted to store variant configuration information;
 - a second data store adapted to store provisioning information;
 - a provisioning module adapted to select the provisioning information stored in said second data store as a function of the variant configuration information stored in said first data store and apply the selected provisioning information to provision the data communications client; and
 - a user interface interacting with said provisioning module to enable a user of the host communications device to provision the data communications client.
2. The apparatus as recited in claim 1, wherein the variant configuration information includes a parameter selected from the group consisting of a device variant parameter, a lock parameter, and a default language parameter,
3. The apparatus of claim 2, wherein at least one portion of the variant configuration information is read-only.

4. The apparatus as recited in any of claims 1 to 3, wherein the variant configuration information includes a current variant.
5. The apparatus as recited in any of claims 1 to 4, wherein the variant configuration information includes a change flag.
6. The apparatus as recited in any of claims 2 to 5, wherein at least one portion of the variant configuration information is read-write.
7. The apparatus as recited in any of claims 1 to 6, wherein the provisioning configuration information includes text that appears in the data communications client during a first time start of the host communications device.
8. The apparatus as recited in any of claims 1 to 7, wherein the provisioning configuration information includes data communications client provisioning configuration information.
9. The apparatus as recited in claim 8, wherein the data communications client provisioning information is set by the manufacturer of the data communications client.
10. The apparatus as recited in claim 8, wherein the data communications client provisioning information is set by the operator of the communications network.

11. The apparatus as recited in claim 8, wherein the first data store is further adapted to read information to control application functionality of the data communications client.
12. The apparatus as recited in claim 11, wherein the first data store is further adapted to store the information to control application functionality of the data communications client.
13. The apparatus as recited in claim 12, wherein the first data store is further adapted to set a change service flag.
14. The apparatus as recited in any of claims 1 to 13, wherein the first data store is further adapted to store read-only operator settings.
15. The apparatus as recited in claim 14, wherein the data communications client retrieves the read-only operator settings from said first data store.
16. The apparatus as recited in any of claims 1 to 15, wherein the apparatus is provided in the host communications device.
17. A method for provisioning a data communications client on a host communications device, the host communications device adapted to operate on a communications network, the method comprising the steps of:
 - enabling a user of the host communications device to select an option for the provisioning of the data communications client;
 - storing variant configuration information in a first data store on said host communications device;

storing provisioning information in a second data store on said host communications device;

selecting the provisioning information stored in said second data store as a function of the variant configuration information stored in said first data store based on said option selected in said enabling step; and

applying the selected provisioning information to provision the data communications client.

18. The method of claim 17, wherein said variant configuration information is also applied to native host applications.
19. The method of claim 17 or 18, wherein the method further comprises clearing said data communications client of data using a clear data store prior to said applying step.
20. The method of any of claims 17 to 19, wherein the method further comprises clearing data from host native applications on said host communications device using a clear data store prior to said applying step.
21. The method of any of claims 17 to 20, wherein said storing provisioning information step is performed during a manufacturing process.
22. The method of any of claims 17 to 21, wherein said storing provisioning information step is performed during a configuration process
23. The method of any of claims 17 to 22, wherein the selecting step is performed through a native interface and abstraction layer on the data communications client.

24. The method of claim 18, wherein the applying step for host native applications is performed based on a command received from said data communications client at said native host applications.
25. The method of claim 18, wherein the applying step for host native applications is performed based on a listener module finding a change in provisioning on said data communications client and alerting said native host applications.
26. The method of any of claims 17 to 25, wherein the storing variant configuration information at first data store step further stores information to control application functionality of the data communications client.
27. The method of claim 26, further comprising the step of setting a change service flag at the first data store.

1/14

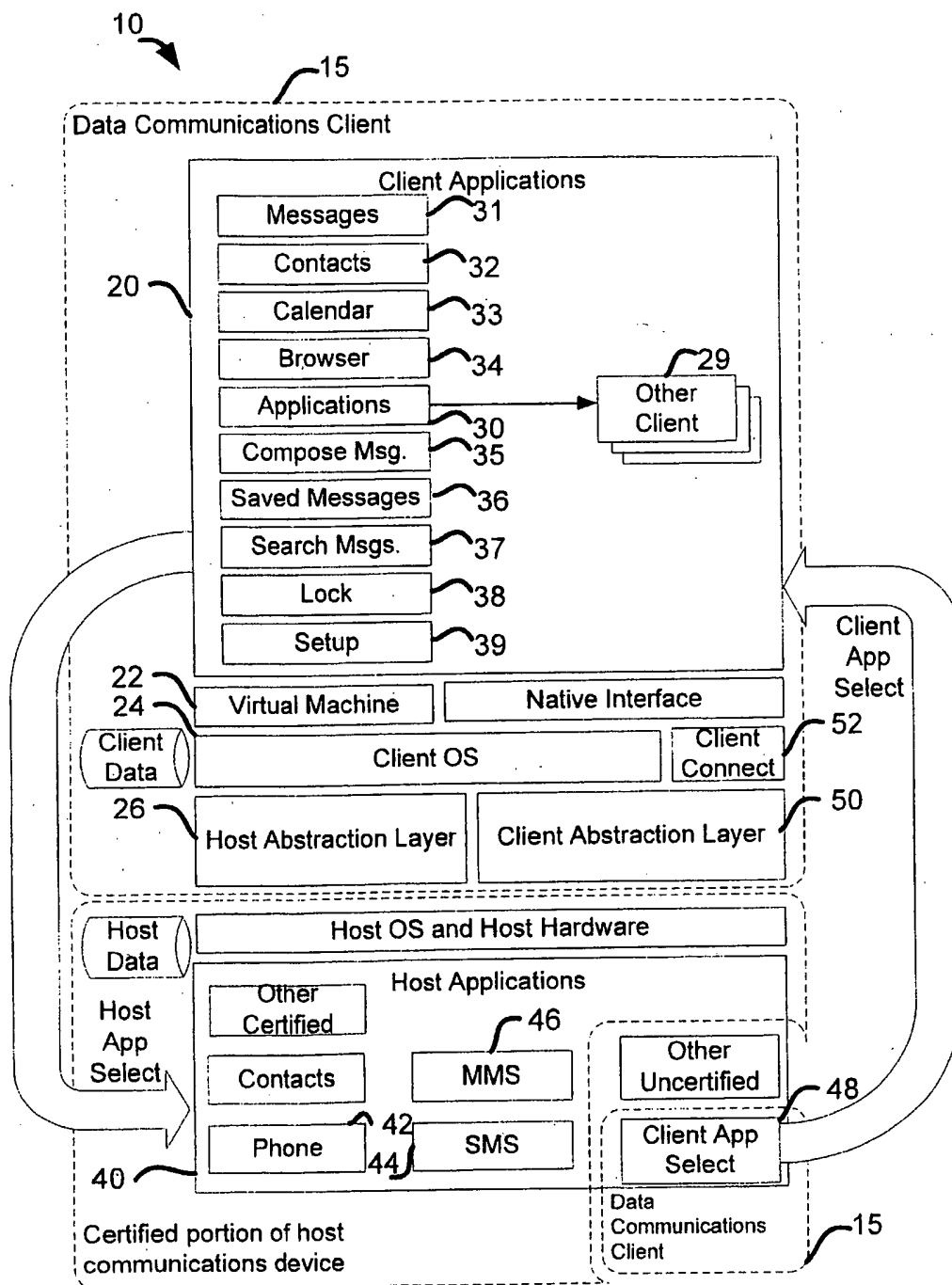


FIG. 1

2/14

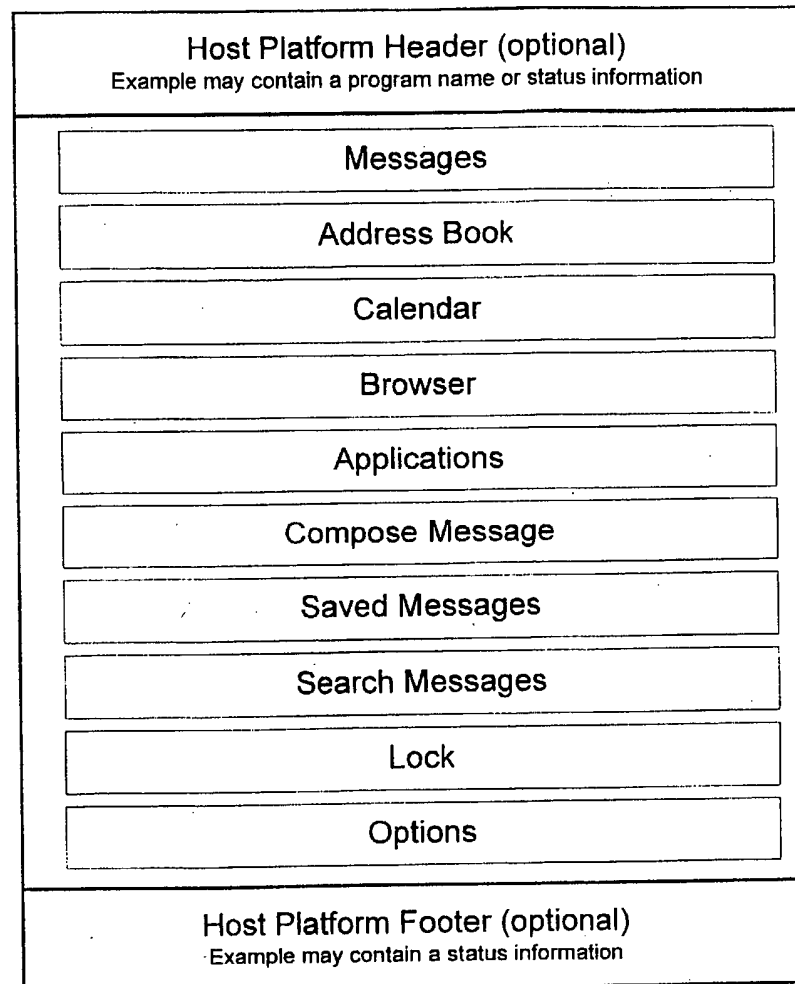


FIG. 2

3/14

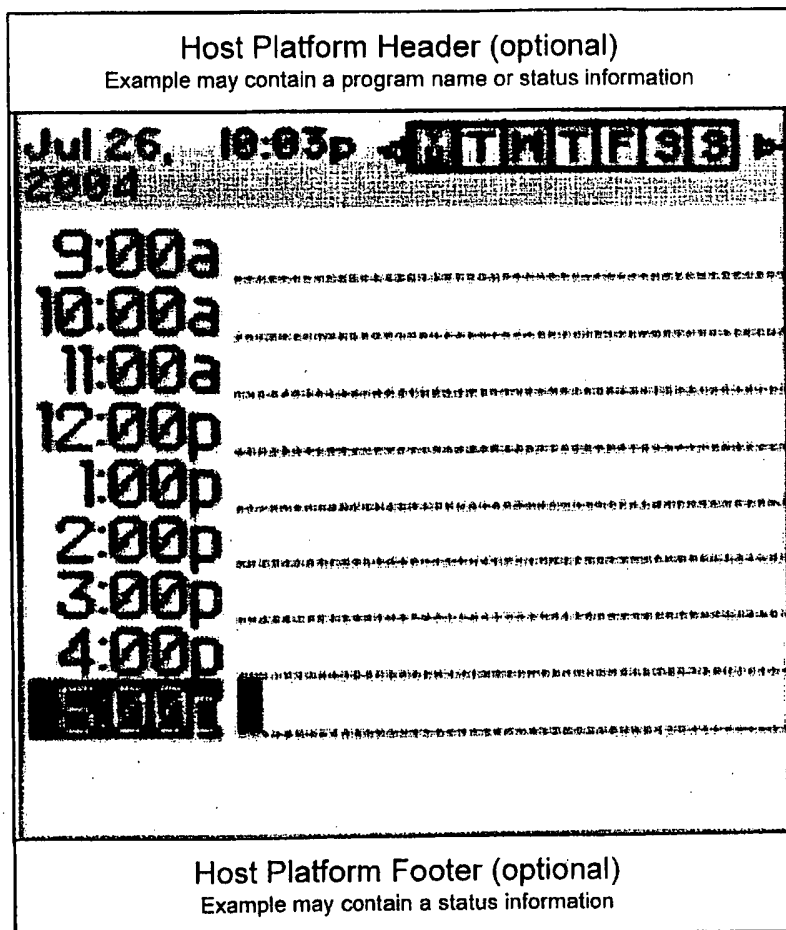


FIG. 3

4/14

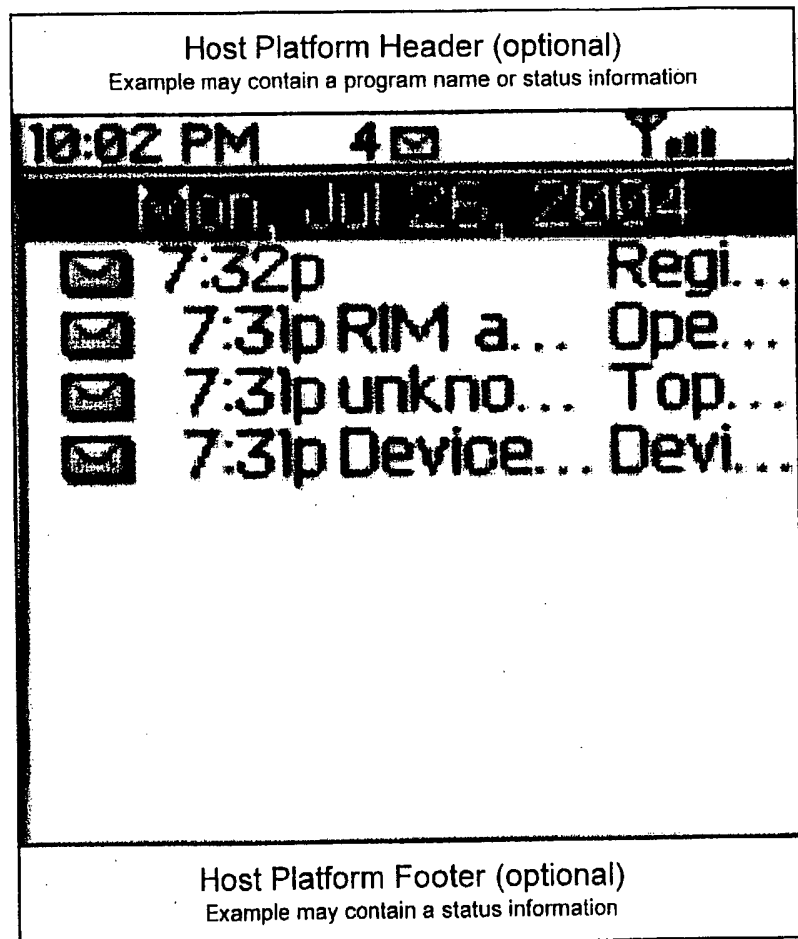


FIG. 4

5/14

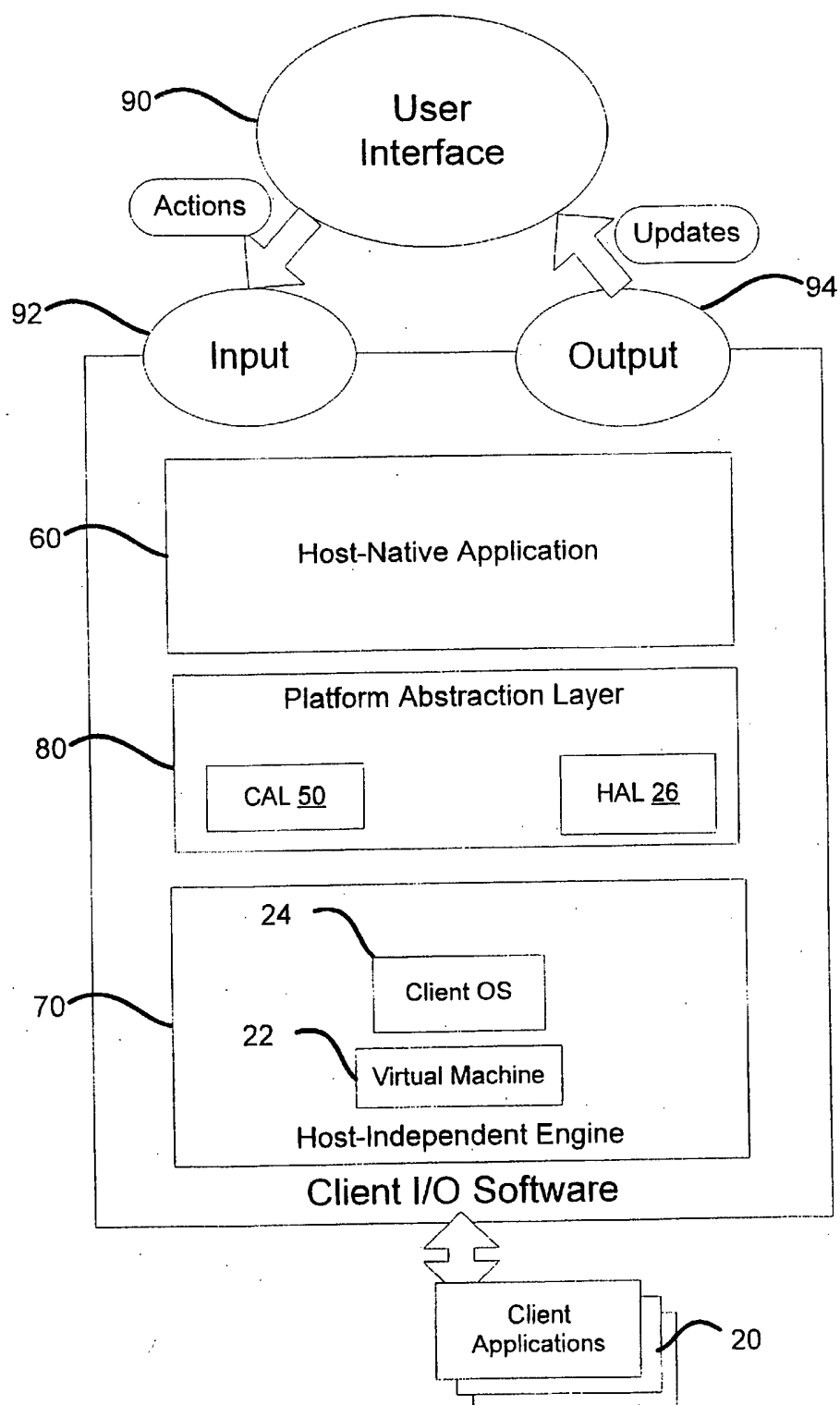
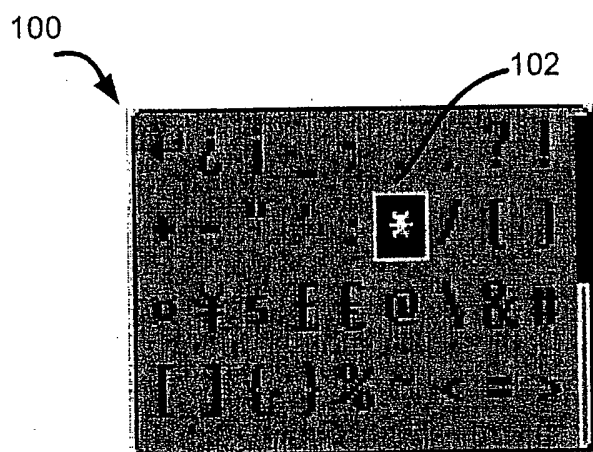


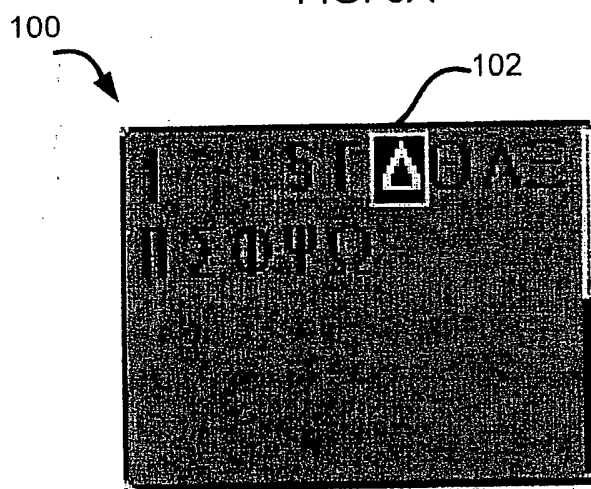
FIG. 5

6/14



HOST SYMBOL 1

FIG. 6A



HOST SYMBOL 2

FIG. 6B

FIG. 6

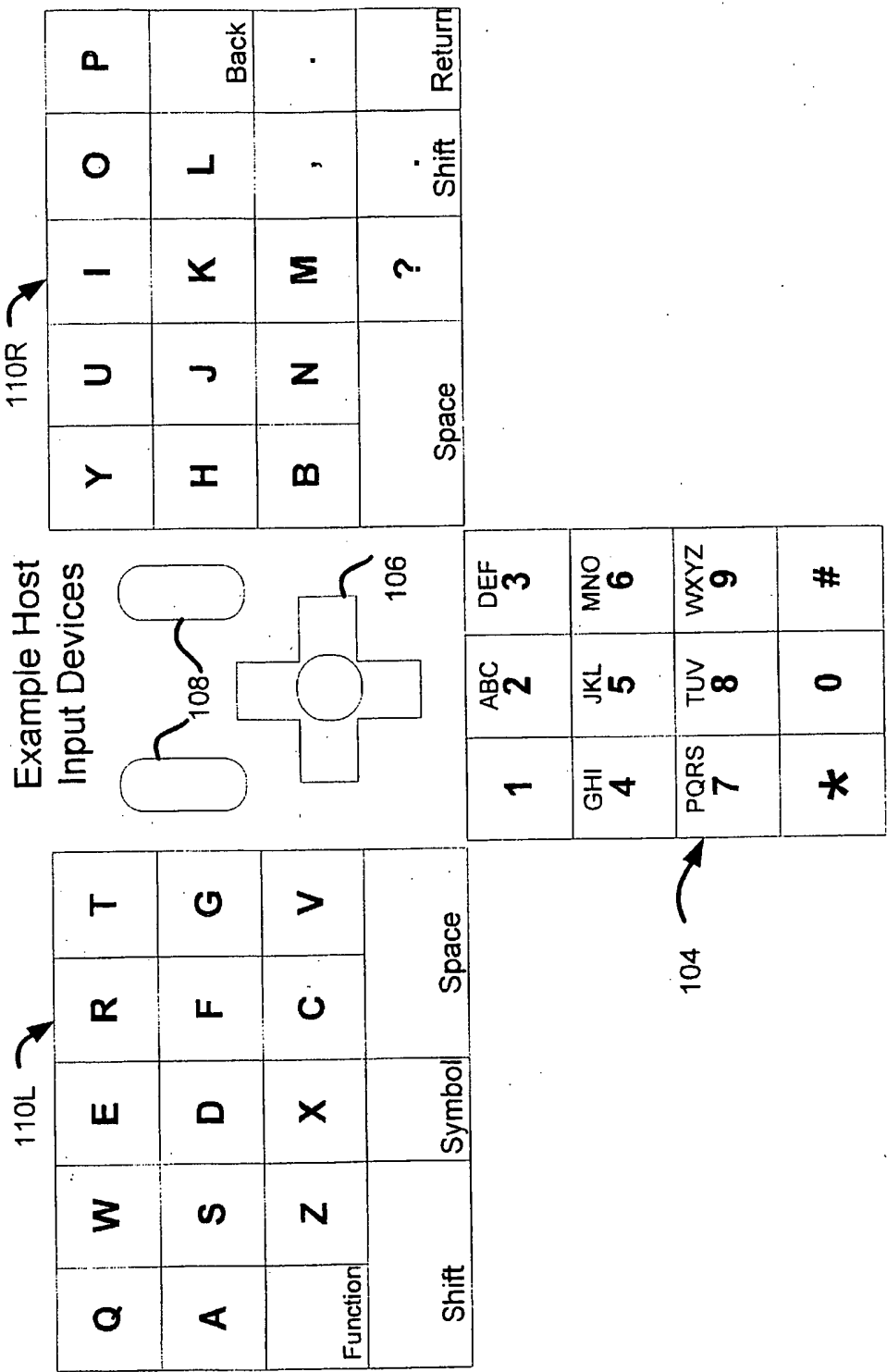
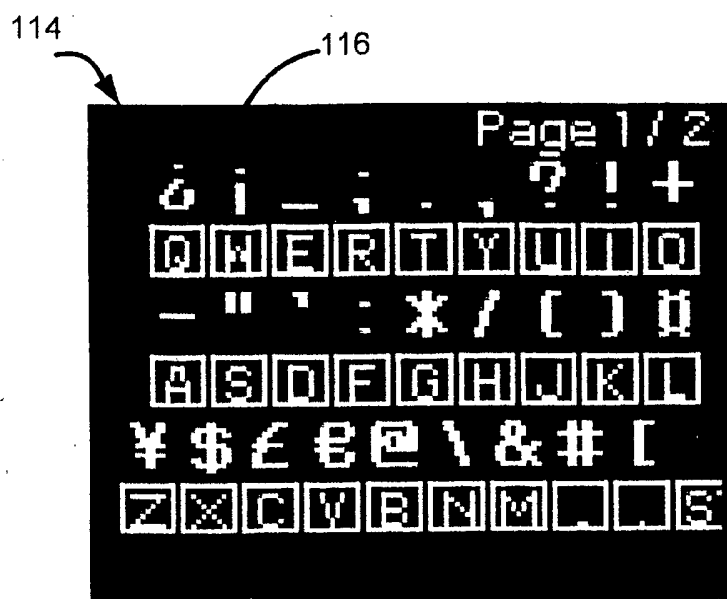
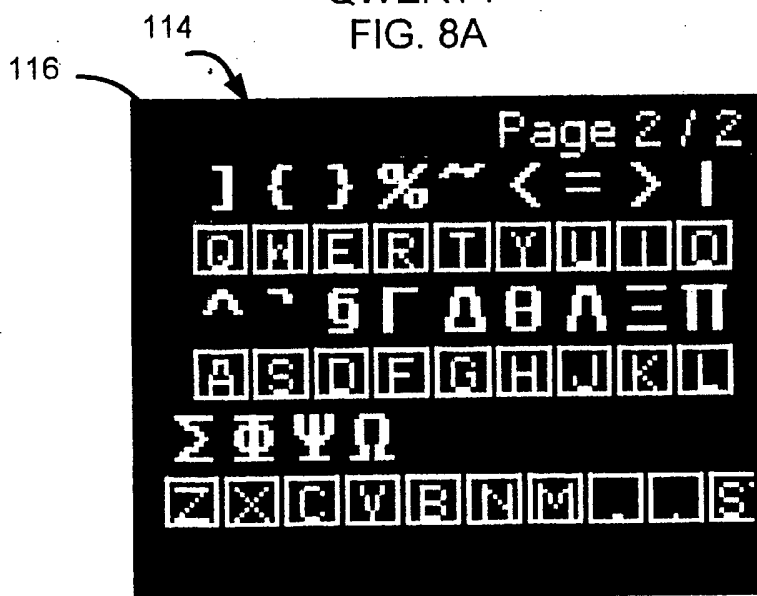


FIG. 7

8/14



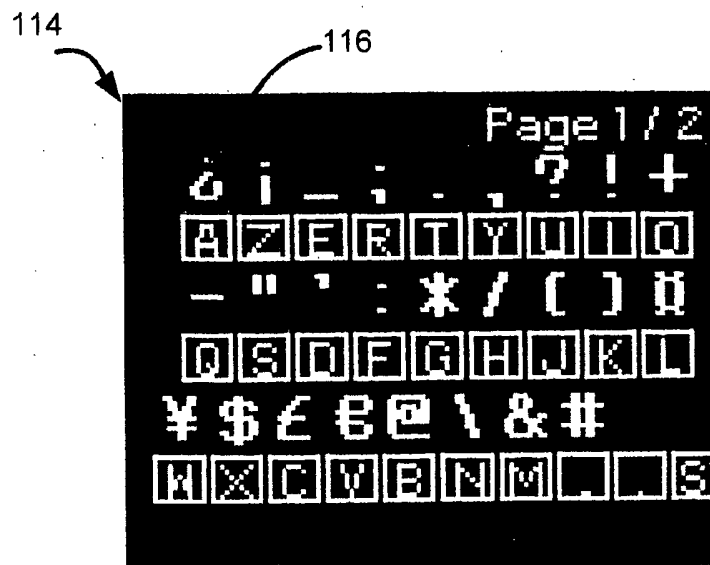
QWERTY
FIG. 8A



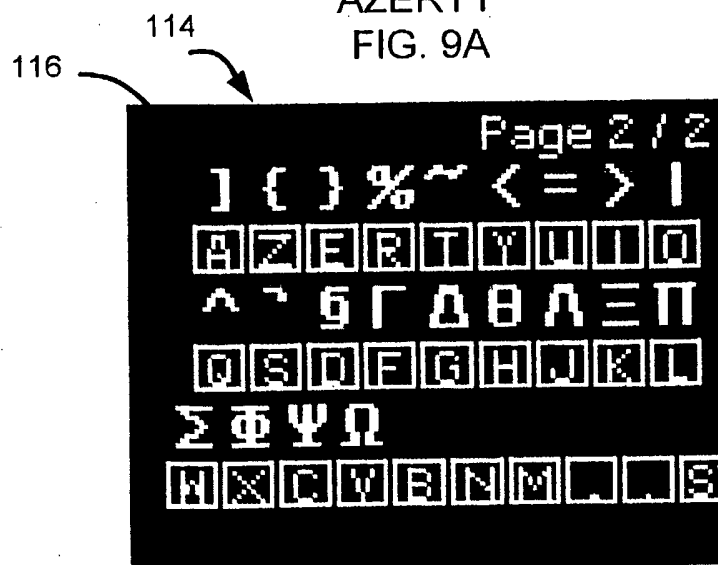
QWERTY
FIG. 8B

FIG. 8

9/14



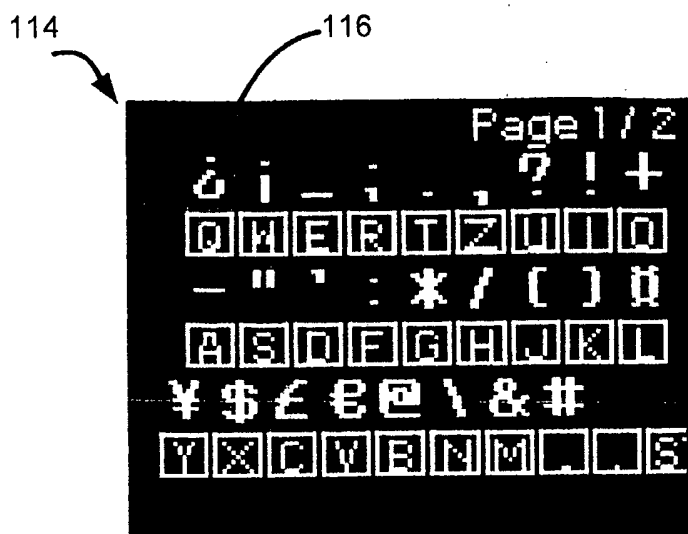
AZERTY
FIG. 9A



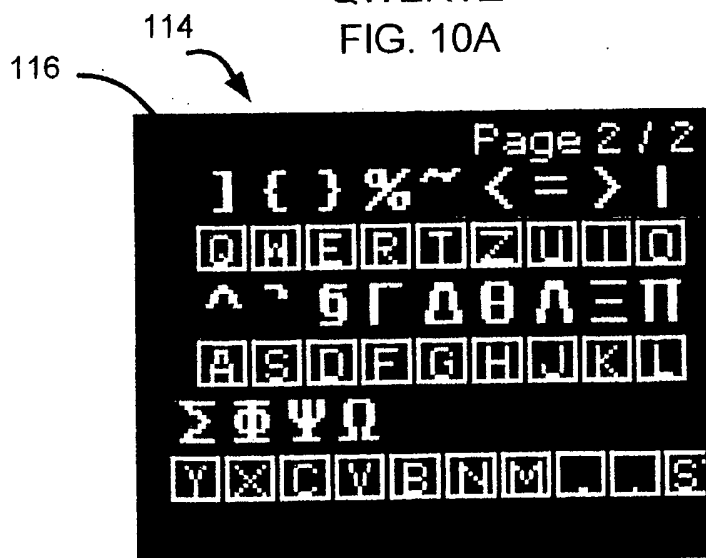
AZERTY
FIG. 9B

FIG. 9

10/14



QWERTZ
FIG. 10A



QWERTZ
FIG. 10B

FIG. 10

11/14

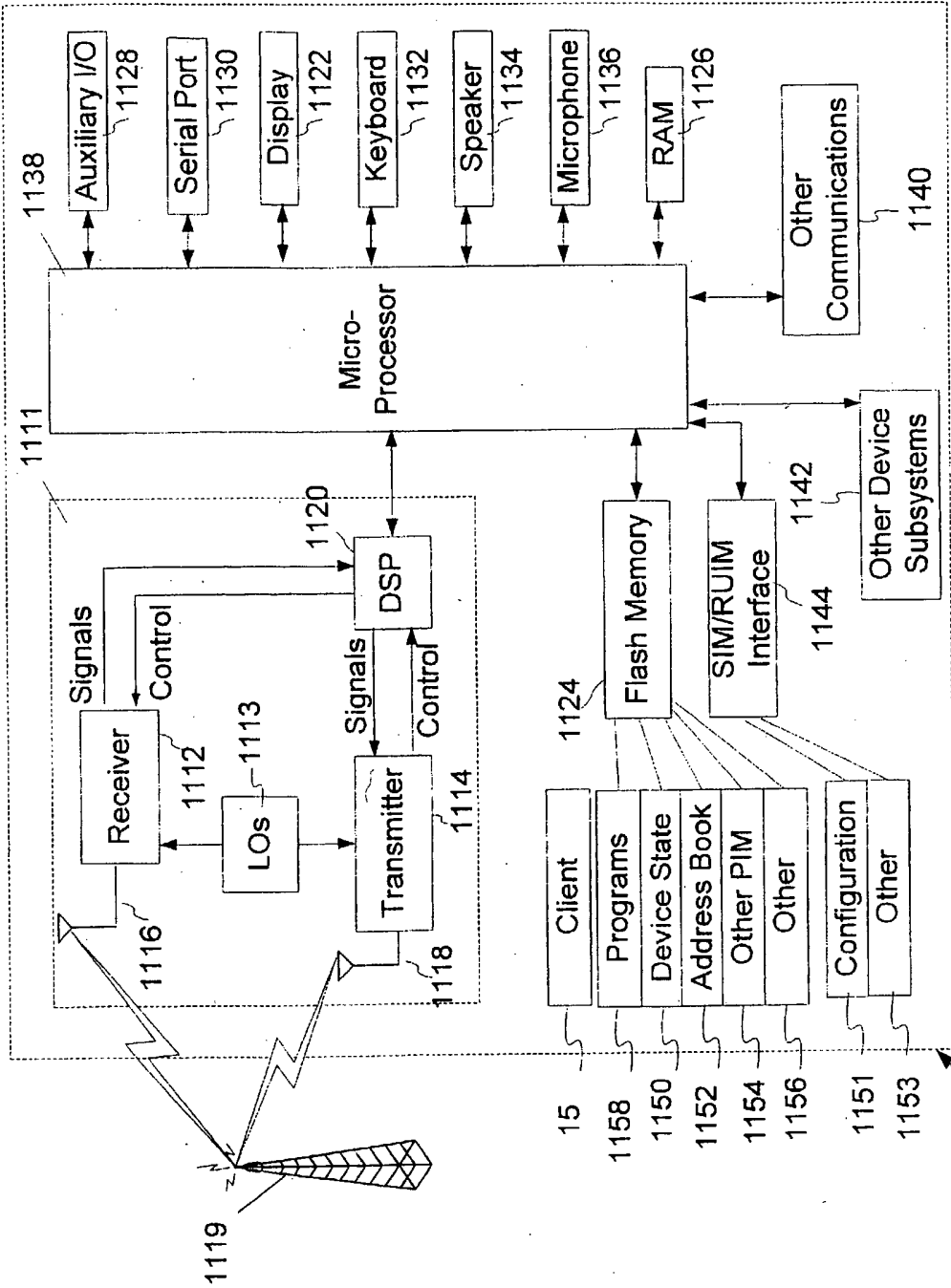


FIG. 11

12/14

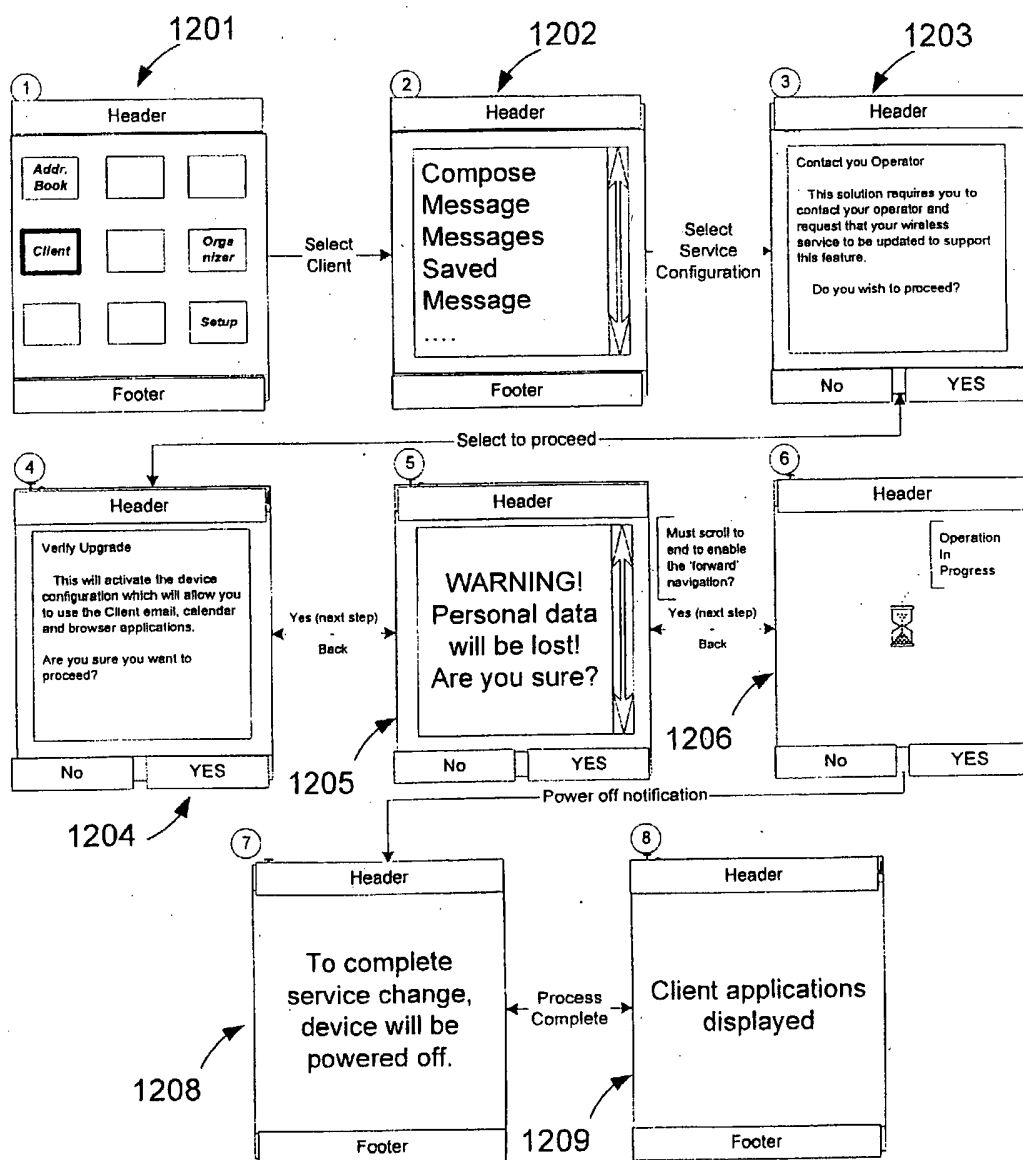


FIG. 12

13/14

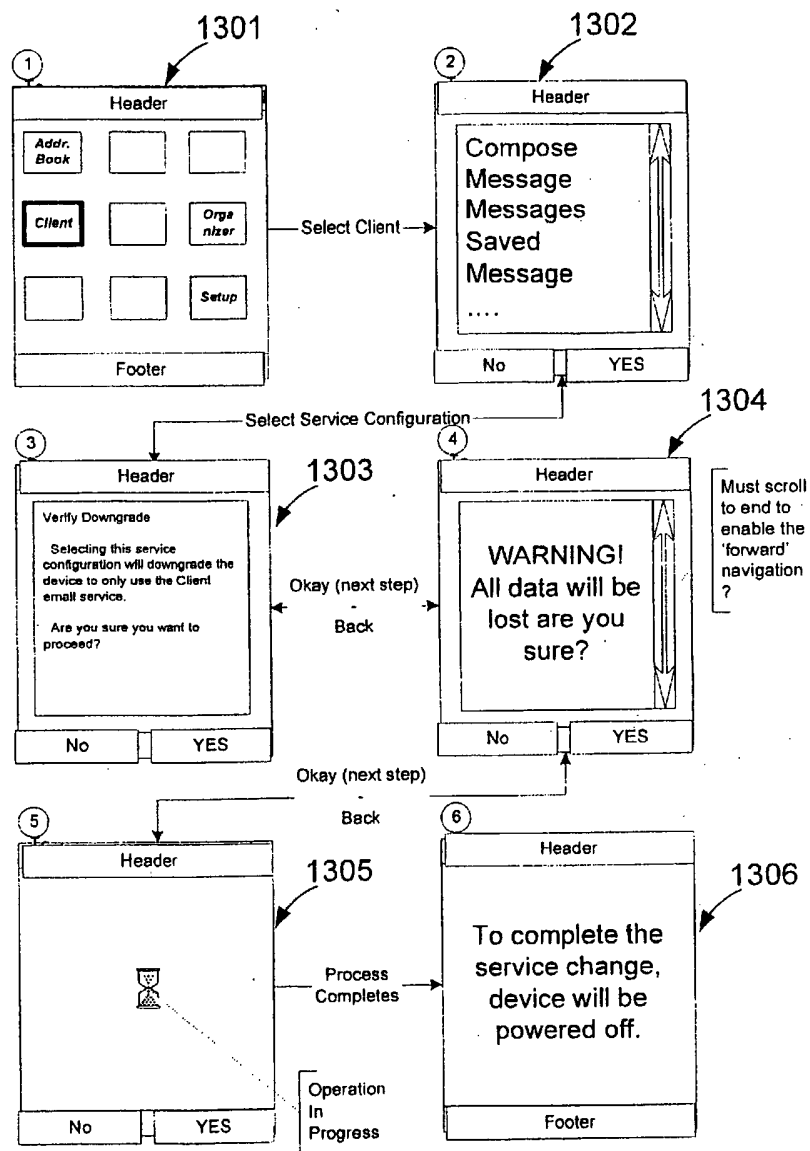


FIG. 13

14/14

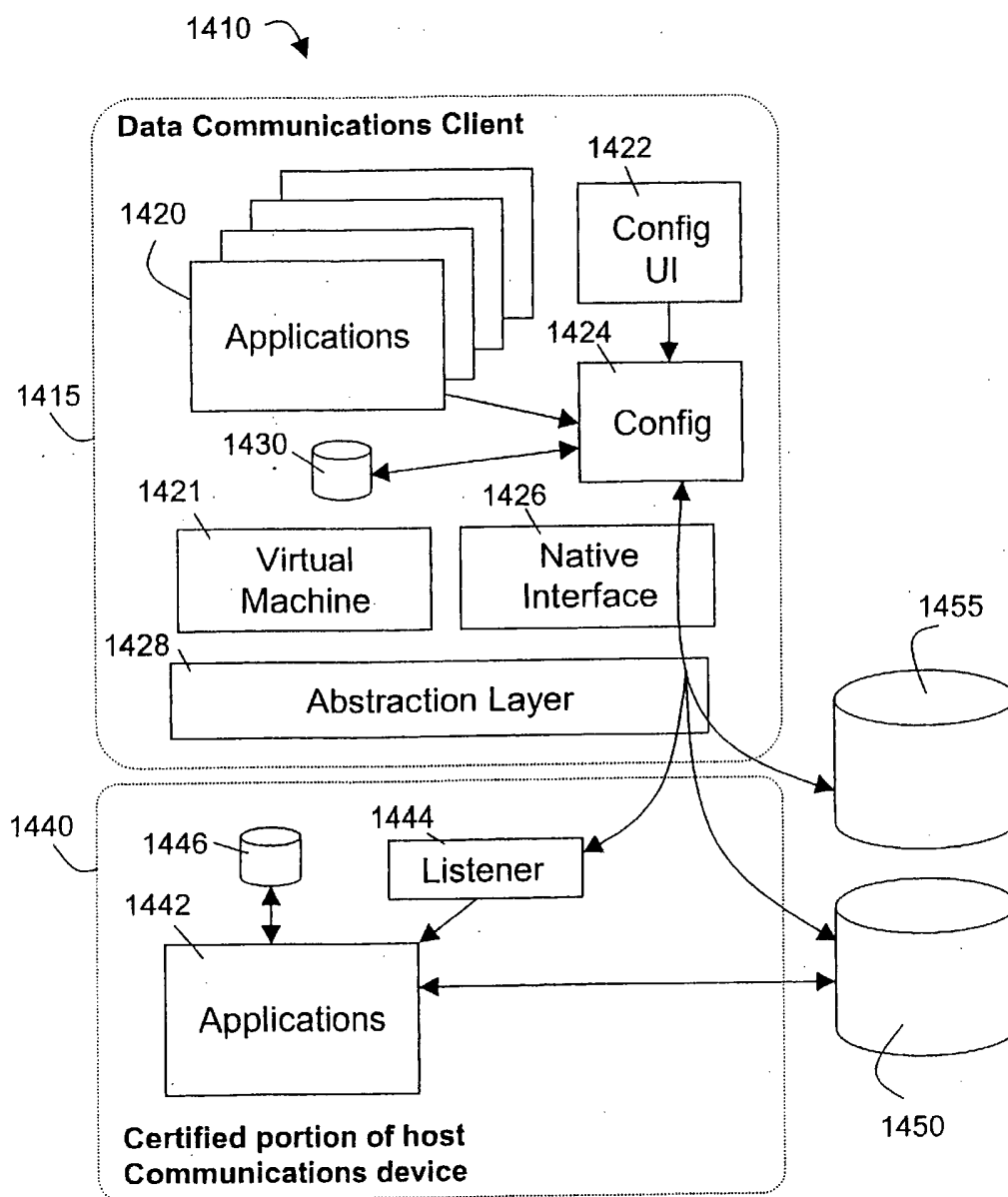


FIG. 14