

[0021] FIG. 5 is a flow diagram representing logic for processing an XML-formatted file for writing data therefrom into a package, in accordance with an aspect of the present invention;

[0022] FIG. 6 is a flow diagram representing logic for generating a package, in accordance with an aspect of the present invention;

[0023] FIGS. 7A and 7B comprise a flow diagram representing logic for creating a device manifest file to describe a package, in accordance with an aspect of the present invention;

[0024] FIG. 8 is a flow diagram representing logic performed by a tool (relmerge) to insert relocation information for executable files, in accordance with an aspect of the present invention;

[0025] FIG. 9 is a block diagram representing the format of a device manifest file that describes a package, in accordance with an aspect of the present invention;

[0026] FIGS. 10A and 10B comprise flow diagrams for building a device manifest file to describe a package, in accordance with an aspect of the present invention; and

[0027] FIG. 11 is a flow diagram representing the creation of a package, in accordance with an aspect of the present invention;

DETAILED DESCRIPTION

EXEMPLARY OPERATING ENVIRONMENT

[0028] FIG. 1 shows functional components of one such handheld computing device 120, including a processor 122, a memory 124, a display 126, and a keyboard 128 (which may be a physical or virtual keyboard, or may represent both). A microphone 129 may be present to receive audio input. The memory 124 generally includes both volatile memory (e.g., RAM) and non-volatile memory (e.g., ROM, PCMCIA cards, and so forth). An operating system 130 is resident in the memory 124 and executes on the processor 122, such as the Windows® operating system from Microsoft Corporation, or another operating system.

[0029] One or more application programs 132 are loaded into memory 124 (or execute in place in ROM) and run on the operating system 130. Examples of applications include email programs, scheduling programs, PIM (personal information management) programs, word processing programs, spreadsheet programs, Internet browser programs, and so forth. The handheld personal computer 120 may also include a notification manager 134 loaded in the memory 124, which executes on the processor 122. The notification manager 134 handles notification requests, e.g., from the application programs 132. Also, as described below, the handheld personal computer 120 includes networking software 136 (e.g., hardware drivers and the like) and network components 138 (e.g., a radio and antenna) suitable for connecting the handheld personal computer 120 to a network, which may include making a telephone call.

[0030] The handheld personal computer 120 has a power supply 140, which is implemented as one or more batteries. The power supply 140 may further include an external power source that overrides or recharges the built-in batteries, such as an AC adapter or a powered docking cradle.

[0031] The exemplary handheld personal computer 120 represented in FIG. 1 is shown with three types of external notification mechanisms: one or more light emitting diodes (LEDs) 142 and an audio generator 144. These devices may be directly coupled to the power supply 140 so that when activated, they remain on for a duration dictated by a notification mechanism even though the handheld personal computer processor 122 and other components might shut down to conserve battery power. The LED 142 preferably remains on indefinitely until the user takes action. Note that contemporary versions of the audio generator 144 use too much power for today's handheld personal computer batteries, and so it is configured to turn off when the rest of the system does or at some finite duration after activation.

[0032] Note that although a basic handheld personal computer has been shown, virtually any device capable of receiving data communications and processing the data in some way for use by a program, such as a mobile telephone, is equivalent for purposes of implementing the present invention.

SELF-DESCRIBING SOFTWARE IMAGE UPDATE COMPONENTS

[0033] The present invention is generally directed towards installing and/or updating software that is stored on small mobile computing devices, such as Microsoft Windows® CE-based portable devices, including those in which the initial software or software update is written to the embedded device's non-volatile memory, e.g., flash memory. Notwithstanding, the present invention provides benefits to computing in general, and thus may apply to other computing devices and other types of storage, including various types of memory and/or other types of storage media such as hard disk drives. For purposes of simplicity, the term "flash" hereinafter may be used with reference to the updatable storage of a device, although it is understood that any storage mechanism is equivalent. Further, the term "image" will generally include the concept of the initial software installation image as well as subsequent software updates to an image, even when only part of an existing image is updated.

[0034] By way of background, contemporary operating systems such as the Windows® CE operating system are modular (componentized) However, the resultant image that contains the correct files and settings is a monolithic operating system image. To this end, at build time, feature variables are mapped to specific files and settings to determine what is contained in the resultant monolithic operating system image. The ability to perform this mapping makes use of two types of build-time configuration files: binary image builder (.bib) and registry (.reg) files. The .bib file contains a list of files which are to be included in the resultant image, and the reg file contains a list of registry (setting) information to be included in the image. The contents of these files are grouped into collections by feature and wrapped in conditional variables which can optionally be set at build time. When a conditional feature variable is set at build time, the associated contents of the .bib and .reg files are included in the image, and as such the user of the system has the ability to select, at a granular feature level, what the resultant image should contain.

[0035] Higher-level logic is also applied to the selection of conditional variables, such that feature-level dependencies