

SELF-DESCRIBING SOFTWARE IMAGE UPDATE COMPONENTS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present invention claims priority to U.S. provisional patent application Ser. No. 60/530,129 filed Dec. 16, 2003, and incorporated herein in its entirety.

[0002] The present invention is related to the following U.S. patent applications, filed concurrently herewith and incorporated herein in their entireties:

[0003] Docket no. 4271/307,649 "Applying Custom Software Image Updates To Non-Volatile Storage in a Failsafe Manner;"

[0004] Docket no. 4281/307,650 "Determining the Maximal Set of Dependent Software Updates Valid for Installation"

[0005] Docket no. 4291/307,651 "Ensuring that a Software Update may be Installed or Run only on a Specific Device or Class of Devices" and

[0006] Docket no. 4311/307,663 "Creating File Systems Within a File In a Storage Technology-Abstracted Manner."

FIELD OF THE INVENTION

[0007] The invention relates generally to computing devices, and more particularly to updating non-volatile storage of computing devices.

BACKGROUND

[0008] Mobile computing devices such as personal digital assistants, contemporary mobile telephones, and hand-held and pocket-sized computers are becoming important and popular user tools. In general, they have become small enough to be extremely convenient, while consuming less battery power, and at the same time have become capable of running more powerful applications.

[0009] During the process of manufacturing such devices, embedded operating system images are typically built into a monolithic image file and stored in non-volatile storage (e.g., NAND or NOR flash memory, a hard disk and so forth) of each device. As a result, updating such a device is necessary or desirable from time-to-time.

[0010] However, a monolithic operating system has a number of disadvantages, including that to install an update, a large amount of resources (e.g., temporary storage and bandwidth) are needed to replace the entire monolithic image. At the same time, installing some subset components of the operating system is a difficult task, for various reasons. What is needed are mechanisms that facilitate the updating of some subset of an operating system image.

SUMMARY OF THE INVENTION

[0011] Briefly, the present invention is directed towards a system and method that provide installation and update packages, wherein each package comprises an encapsulation of a set of files that which are treated the same for the purposes of installation, and wherein the format of the package is self-describing, thereby facilitating the replacement of only component parts of an image. To this end, the

system and method maps operating system features (comprising files, metadata, configuration information and so forth) into packages as part of a software build process.

[0012] In one implementation, packaging logic handles cases where specific files and/or settings are shared between related features, which the user in turn chooses to map to differing packages. The logic generally ensures that individual files/settings are mapped to the correct package, given a number of possible higher level package mapping requests. Further, packages optionally convey dependency information, and thus a mechanism is provided (via feature-level dependency specifications) by which packages acquire the dependency information. Logic resolves conflicts and dependencies below the feature level.

[0013] During the build process, a build manifest file is created by taking a binary image builder file for the operating system image and a component to package mapping file as inputs. The build manifest file specifies the file contents for a particular package. These file contents are reviewed, and any executable code is processed prior to insertion into the package to enable executable code relocation/fix-up on the device at install time. A package generation process creates the device manifest based on information in the build manifest and a package definition file.

[0014] The registry for the operating system image is broken up and assigned to packages based on a similar algorithm, and XML files may be similarly broken up and assigned to specific packages. The result is a number of files for each package that is to be constructed, possibly including a package definition file, a component mapping file, a component relations file, a build manifest file; registry file and an XML settings file. From these files a package generation process constructs a final package file by creating a package collection from the packages, including mapping each package to the package definition, reading the build manifest file for that package and generating the package from that data.

[0015] For a package to be self-describing, a device manifest file is created during the packaging process and stored in the package itself. The device manifest file is used during the installation process. Package dependency and shadow (package settings priority) data is also part of the data accompanying a package, e.g., by writing it into the device manifest file.

[0016] Other advantages will become apparent from the following detailed description when taken in conjunction with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] FIG. 1 is a block diagram generally representing a computer system into which the present invention may be incorporated;

[0018] FIG. 2 is a block diagram representing various components for constructing self-describing update packages, in accordance with an aspect of the present invention;

[0019] FIG. 3 is a flow diagram representing logic for creating a build manifest file from a binary image file in accordance with an aspect of the present invention;

[0020] FIG. 4 is a flow diagram representing logic for creating a registry settings-related file from registry settings in accordance with an aspect of the present invention;