RiP Dev

Installer 4 Repository Format

12.08.2008
Revision number: r1050

RiP Dev KMK Research Sàrl Grand-Chêne 1-3, CH-1002 Lausanne, Switzerland **http://ripdev.com** **info@ripdev.com**

RiP Dev

Nuts & Bolts

Introduction

Installer 4 features a change in the repository format, first time in the history of the product. The goal is to minimize the traffic for the repository owners (and users), extend the possibilities and minimize the work required for the repository owner.

Basics of Operation

The provided repository PHP code essentially does all the work for you, but it's always nice to understand how things work. If you're not interested, please proceed below for the information on how the package is built.

Root Repository Index

The "root" of repository is now a property list file (like before), with the only two differences:

- The only included information about the package are the essential fields and not everything like before, to minimize the traffic load and the impact on the user.
- The packages list returned is being filtered based on the version of the firmware supplied to the repo index by the client.

Individual Package Info

Every package now has an individual property list on the server containing "extended" info, such as download link, hash, file size, more info URL and more. When user chooses the package in the Installer, the info is fetched and cached on the client side.

The new thing added to this version of Installer is the ability to supply a custom URL pointing to a special web page that will override the "standard" package information page. This allows greater flexibility for presentational purposes, as you can design the look yourself (if you can comprehend the additional web server traffic, that is.

The other new thing is the ability to supply an URL to the 60x60 px PNG file to be used as a package icon.

Package File Structure

The package is now self-contained and is required to have a file named **Install.plist** in the root of the archive. This file is essentially a package entry like in the old repository, but the repository script supplied will automatically parse that file to extract necessary information about the package and insert it into the repository root index and create individual package info files. Also, if **Install.png** is found in the archive root, it is treated as an icon for the package and should be less than or equal to 60x60 px size.

Install.plist itself is a property list (*must be in plain text* and not binary format, as PHP repo script cannot parse binary-encoded property lists), containing a *dictionary* with various keys describing the package.

Here is the description of the fields that should (or can) be found within Install.plist:

- **identifier** (required): package identifier, in the Java format: com.domain.product-identifier
- **description** (required): brief description about the package. It will be truncated in the packages list and displayed fully on the package information page.
- **name** (required): the name of the package, ex: My Great Package

RiP Dev

- **version** (required): the version of the package. This version of Installer enforces strict versioning rules to allow better version comparison. The valid format is: *AA.B[.C][S][BB][-PACK]*, where AA is major version (up to 2 digits), B is the minor revision (1 digit only), C is the sub-revision (1 digit only), S is the release stage (must be either 'd' (development), 'a' (alpha), 'b' (beta) or 'f' (final)), BB is the build number (up to 2 digits) and PACK is the packaging number. The components in the block parenthesis are not required. An example of valid versions: 1.0, 17.2, 1.0.1, 1.3.4b4, 1.9.2f17, 1.0.3-14, 1.0.2b7-4. An example of invalid versions: 1.12, svn114, 1.17b14. A packaging number is merely a sign for the Installer that the file was changed but the version itself didn't - for example, if the package maintainer messed up with some scripts and would like the package to appear as an update.
- **scripts** (required): a dictionary containing arrays containing arrays of script commands; this is the same as with the "old" Installer. The complete list of the commands is forthcoming. Currently, five types of scripts are supported: *install, uninstall, update, preflight, postflight*.
- **sponsor** (optional): the name of the sponsor for this package.
- **maintainer** (optional): the name of the maintainer for this package.
- **contact** (optional): e-mail address of the person responsible for this package
- **customInfo** (optional): URL to the web page to be used as a custom info page for the package, overriding the default one.
- **url** (optional): URL to the location of the package information page (displayed when user clicks on the "more info" button in the package description).
- **icon** (optional): URL to the PNG file (60x60 px or less) with the custom package icon. If the icon is not specified, the repository default icon will be used for the package. If repository icon is not set, a generic package icon will be used.
- **dependencies** (optional): an array of strings containing package identifiers this package depends on. The Installer will attempt to install these before yours; if certain packages are not available, your package will not be installed as well.
- **minOSRequired** (optional): a string containing minimum firmware version this package is supported on. Example: 2.0, 2.0.1, 2.1. The repository code will filter the packages based on this field (if present). If not present, the package is assumed to be runnable in all firmware versions.

Script Commands

Description forthcoming. For now, keep in mind that the scripts are completely compatible with the Installer 3.x script commands. If you miss a certain command and would like to have it implemented, please contact us at feedback@ripdev.com.

Preparing The Repository

To run the new repository, you need a server supporting PHP. PHP 4 should suffice, but we didn't test it there yet. A generic setup with the PHP 5 would be sufficient. Here is how to get your repo running in a few easy steps:

- **Upload** the contents of the Repository folder into the web directory of your choice.
- **Rename** config.inc.default.php to config.inc.php.
- **Edit** config.inc.php so it contains reasonable configuration values.
- **Edit** Info.plist so it describes your repository (if you want, you can add an *icon* key to it to supply a repository icon png URL).
- Make sure the info/ subdirectory is **writable** by your Web server user (e.g., permissions are set to 777).
- **Put** some categories and packages in the packages/ subdirectory (each directory inside is a separate category).

RiP Dev

- **Run** regenerate.php script by accessing it in your Web browser. It will generate all necessary index files for your packages. Run regenerate.php every time you add or remove a package to the repository.
- That should be it. Try **accessing** your repo by appending ?debug=1 to the URL.

Web Server Request Errata

Just for your information, Installer 4 will always append certain fields to all requests it makes to the outside world, regardless whether it is a request for an icon, repo index, or the package itself. It is up to you how to use this information; below is a description of various GET variables passed to your Web server by Installer 4.

- **platform**: contains a string describing the platform Installer 4 is running on. Could be “iPhone”, “iPod touch” or “iPhone Simulator” (at the moment of the writing). Can be later used to filter packages based on platform (not yet implemented in the current repo code).
- **firmwareVersion**: contains a string with the device firmware version, for example “2.0”. Used by the repo code to filter packages.
- **deviceUUID**: a unique string identifying the device which will not change across restores and resets but which is anonymous enough to be unable to be traced back to a particular piece of information about the device (such as IMEI, MAC address, phone number or whatnot). Can be safely used as a “unique device” tracking identifier.
- **installerVersion**: a string with the version of the Installer application, for example, “4.0b2”.
- **locale**: a preferred language code as set by the user, for example, “en”, or “ru”. Can be used to supply localized descriptions for the packages, etc. Currently ignored by the repo code.