Wavelink Avalanche Software Package Builder Version 3.0.0 User's Guide

WLPackageBuilder-20050419-01

Revised 4/6/09

Wavelink Corporation
6985 South Union Park Avenue, Suite 335
Midvale, Utah 84047
Telephone: (801) 316-9000
Fax: (801) 255-9699
Email: info@roisys.com
Website: http:\\www.wavelink.com

Email: sales@wavelink.com

Table of Contents

Chapter 1: Introduction

The purpose of this document is to provide an explanation of the installation, configuration, and use of Wavelink Package Builder.

This chapter provides the following information:

- document assumptions
- document conventions
- an overview of Avalanche Package Builder

Document Assumptions

This section describes the assumptions behind this document and the typographical conventions used in this document.

Document Assumptions

This document is intended for IT professionals and assumes the following level of knowledge:

- The user is comfortable in a Windows application environment.
- The user is familiar with the wireless hardware in use, specifically the mobile device.
- The programming languages necessary to create the applications to be used in the mobile device.
- The user is familiar with the Avalanche system environment.

Document Conventions

Document Conventions

This document uses the following typographical conventions:

Courier New

Any time you interact with an Avalanche option, such as a button, or type specific information into a text box, such as a file path name, that option appears in the `Courier New` text style. This text style is also used for keyboard commands that you press.

Examples:

Click `Next` to continue.

Press `CTRL+ALT+DELETE`.

Bold

Any time this document refers to an option, such as descriptions of the choices in a dialog box, that option appears in the **Bold** text style.

Examples:

Click `Open` from the **File** menu.

Click `Download` from the **HexFiles** menu.

Italics

Any time this document refers to another section within the manual, that section appears in the *Italic* text style.

Example:

See the *Troubleshooting* section for possible causes of this problem.

About the Wavelink Avalanche System

Wavelink Avalanche is a client management system that automatically deploys software and configuration updates to mobile devices. Avalanche uses "push/pull" technology to install, update, and manage the software and

configurations of wireless and other mobile devices. The Wavelink Avalanche system includes three primary components:

- **Avalanche Manager.** This tool provides centralized client management within a network. The Avalanche Manager consists of two applications: Avalanche Manager Agent and the Avalanche Management Console. The Agent performs the management functions on the LAN or WAN, while the Management Console provides an administrative interface to one or more Agents

- **Avalanche Enabler.** This tool is an agent that runs on each mobile device to allow device management via Avalanche Manager.

- **Avalanche-enabled software packages.** These software packages are intended to run on the mobile device and include both Wavelink Telnet, Wavelink Studio Client and third-party applications.

The Avalanche Enabler must be installed on the mobile device using the mobile device's native transfer medium. Once the Enabler is installed, the mobile device can communicate directly with the Avalanche Manager over either a wireless or serial connection.

The Enabler and the Avalanche Manager work together to synchronize software on the mobile device to an associated profile on the Management Console. The profile can include one or more distinct software packages, as well as RF and TCP/IP configurations, and can apply to multiple devices.

The Enabler itself can also be updated using Avalanche, which can permanently eliminate the need to manually update specific mobile devices in the future. In addition, the Enabler is application-neutral, allowing organizations to easily pre-load the Enabler at the factory or at a staging location.

Advantages of Using Avalanche Software Packages

The following list summarizes the benefits of using Avalanche Software Packages:

- Packages easily download over a wireless connection rather than being limited to the device's native transfer format (which often requires one or more NVM image downloads over a physical connection).

- Devices can receive the software automatically from a central location without requiring physical manipulation of each device or even actual presence on site. Both “push” and “pull” technology is supported for flexible control over software distribution.

- Updates to the packages occur quickly because only the modified components transfer to the mobile device.

- Avalanche provides a unified management tool and software distribution format to the many different mobile device architectures that are available. The capability to manage platform-dependent parameters such as RF and TCP/IP configurations is also provided, eliminating the need to manipulate these parameters internally.

- Multiple software packages can reside in the devices, giving the user a menu-based means of selecting applications.

- Packages can be configured to automatically perform one-time operations. As an example, RF firmware and/or driver updates are distributed for Avalanche in an auto-run package. Auto-run packages automatically download to each mobile device, re-flash the radio firmware, install an updated driver, and then continue with normal operations without any end user intervention.

- Packages can include plug-in modules that give the user GUI-based configuration and management of the software directly from the Management Console.

- Avalanche software packages are limited in size only by the total storage capacity of the target devices rather than the more limited size of a typical NVM download image.

- Packages can be preconfigured with a “selection criteria” to target specific devices. This ensures that the right software gets to the right device. Files within the package can also have “selection criteria” to further manage the software.

- Files can be uploaded to the PC that the Avalanche Agent resides on.

For detailed information about the advantages of using Avalanche software packages, see the *Avalanche Manager System Guide*.

About the Package Builder

The Package Builder is a tool that simplifies the package creation process. This utility performs several functions:

- It gives you a visual method for creating and editing software packages. It simplifies the configuration of the package through a GUI-based interface.
- It creates a file structure to contain all application files, support files, and plug-in configuration files that are required to run and maintain the application.
- GUI interface provides a window into the software package structure, showing attributes that can be set for each package element.
- It automatically generates the package control file.
- It places all package files into default or configured locations in the file structure.
- Once the software is packaged, it can easily be transferred to a distribution medium, mass produced, and installed by administrators.

Package Builder Features

The following list describes several features that Package Builder offers:

- **Package Versions**. This feature allows you to select the version of the software package, simplifying the creation process.
- **Package Types**. This feature allows the creation of additional package types including: Application, Support, Auto Run and Config packages.
- **Package File Flags**. This feature provides support for the package file flags ALWAYS, DYMANIC, and OPTIONAL providing package flexibility.
- **Retrievable Files**. This feature allows you to specify files that you want to retrieve from the mobile device.
- **Script File Builder**. This feature allows you to create script files with an .ini extension to include in your software package.
- **Command Line**. This features gives you the option to rebuild packages from the command line.

Chapter 2: Installing Package Builder

This chapter provides the following information:

- *System Requirements*
- *Installing the Package Builder*
- *Uninstalling the Package Builder*

System Requirements

The requirements for installing and using the Package Builder are:

- Windows 2000 or XP Professional
- JRE 1.5x

NOTE Required disk space is dependent on the types of software packages you are creating.

Installing the Package Builder

This section contains instructions for installing the Package Builder on a Windows platform.

Package Builder Installation

There are two Package Builder installations:

- Package Builder
- Package Builder with JRE 1.5

If you have JRE 1.5 installed on the machine that will be running Package Builder, you should install the Package Builder installation.

If you do not have JRE 1.5 or later versions installed on the machine that will be running Package Builder, you should install the Package Builder with JRE 1.5 installation.

To install the Package Builder:

1 Obtain the installation package.

NOTE You can obtain the Package Builder at the Wavelink web site: http://www.wavelink.com.

---

2 Open the zip files and double-click `WLPackageBuilder_300.exe` or the `WLPackageBuilder_300_jre.exe` depending on the installation package you downloaded.

The *Introduction* dialog box appears (Figure 2-1).

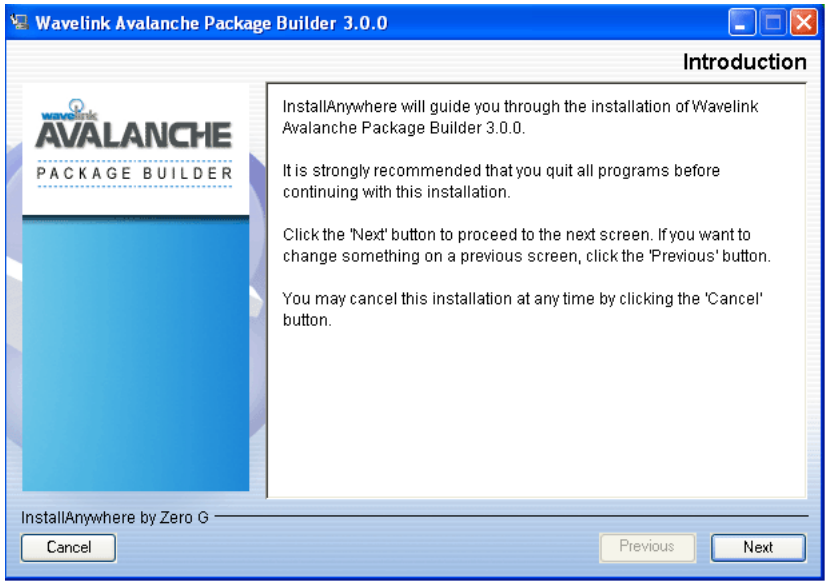

Figure 2-1. *Installation Introduction Screen*

3 Click `Next`.

The *Choose Install Folder* dialog box appears (Figure 2-2).

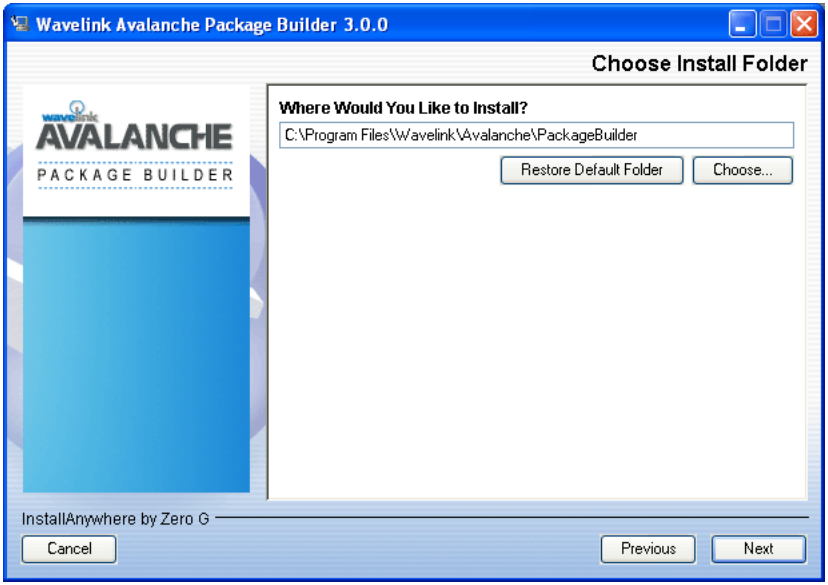

Figure 2-2. *Choose Install Folder Screen*

4 Click `Choose` to select the location where you want to install the files.

-or-

Click `Restore Default Folder` to install the files in the default `C:\Program Files\Wavelink\Avalanche\PackageBuilder`.

5 Click `Next`.

The *Pre-Installation Summary* dialog box appears (Figure 2-3).

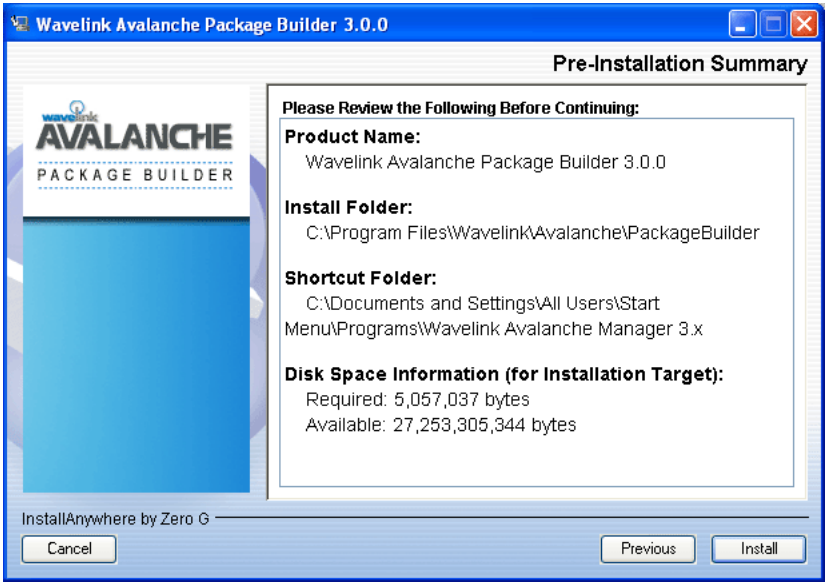

Figure 2-3. *Pre-Installation Summary Screen*

6 Click `Install` to install the Avalanche Package Builder.

Once the installation is complete, the *Get User Input* dialog box appears (Figure 2-4).

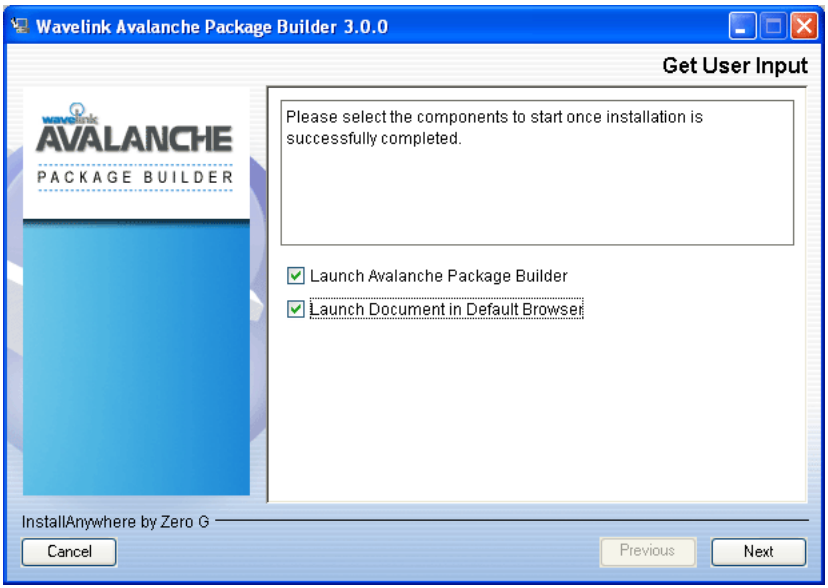

Figure 2-4. *Get User Input Screen*

7 Select **Launch Avalanche Package Builder** and **Launch Documentation in Default Browser**.

8 Click `Next` to launch your selections.

The Package Builder program launches. The Package Builder documentation opens in the default browser.

Uninstalling the Package Builder

This section contains instructions for uninstalling the Package Builder on a Windows platform.

To uninstall the Package Builder:

1 From the **Start Menu**, open the **Control Panel**.

2 Select **Add or Remove Programs**.

3 Locate and select Wavelink Package Builder 3.0.

4 Click `Change/Remove`.

Chapter 3: Package Builder

The Package Builder allows you to create Avalanche software packages. This chapter includes information on the following topics:

- *Overview of Avalanche Software Packages*
- *Building a Package*
- *Validating a Package*
- *Making a Package*
- *Installing Packages*
- *Modifying a Package*

Overview of Avalanche Software Packages

An Avalanche software package includes the following components:

- **Primary application files**. These files include application, configuration, and support software intended to run on a mobile device.
- **Plug-in management tools**. These include configuration utilities that allow Management Console operators to easily configure and manage the software package.
- **Package control file**. This file contains a master description of the package, including the package title, revision code, "selection criteria" that determines which device types can receive the package, and other information. The selection criteria can encompass many device characteristics, including mobile device models, physical characteristics, IP and/or MAC addresses, etc.

Additionally, Version 3 packages may contain the following components:

- **Retrievable Files.** These are files you want to retrieve from the mobile device and copy to the Avalanche Agent. Retrievable files are supported in the 3x Enabler.
- **Post Install File.** This feature allows you to create script files using the Package Builder.

- **Uninstall Command**. This is an executable that may be added to perform specific uninstall instructions, such as the removal of registry settings.
- **Custom EULA**. This feature allows you to include a customer user license agreement that must be made available as an HTML-formatted `License.text` file.

Types of Avalanche Software Packages

You can create four types of packages using Avalanche Package Builder.

- **Application**. These packages can be selected from the Application Menu screen on the mobile device. An example of an Application package is the Telnet Client.
- **Support**. These packages deliver files and do not add new items to the Application Menu screen on the mobile device. An example of a Support package is a package that updates an existing file.
- **Auto Run**. These packages automatically run after download. An example of an Auto Run package is a firmware update.
- **Config**. These packages contain a configuration utility which can modify a configuration file uploaded from a mobile device. This configuration file can then be downloaded back to the mobile device. An example of an Config package is a Text editor to modify a text file.

NOTE Config packages require special support from the Enabler. If supported by the enabler, the device will enable the Get Configuration and Set Configuration options within the Avalanche List View. For information about building a Config package type, please contact Wavelink Customer Service.

Building a Package

NOTE Before you begin creating software packages, assemble all necessary package files in a location you can browse to from the Package Builder.

The tasks that are required to build a package vary depending on the specific types of packages you want build. Creating an Application package differs

from creating a Support package. The information in this chapter is based on building an Application package and includes instructions on the different tasks you can use to build packages. For examples of building other package types refer to *Chapter 5: Example Packages* on page 51.

This section provides information on the following tasks used to build packages:

- *Launching the Package Builder*
- *Configuring Package Properties*
- *Entering Package Information*
- *Adding Package Files*
- *Adding Configuration Utilities*
- *Adding Retrievable Files*
- *Building a Post Install Script File*

Launching the Package Builder

You can launch the Package Builder from the location where you installed it.

To launch the Package Builder

1 Navigate to the location where the Package Builder is installed.

2 Double-click the `AvalanchePackageBuilder.exe` file.

-or-

Double-click the Package Builder shortcut on your desktop.

The Package Builder launches.

Configuring Package Properties

In the *Package Builder Properties* dialog box, you can define specific properties of the software package. For detailed information on Package Builder Properties refer to *Package Builder Properties* on page 34.

To configure package properties:

1 From the **Tools** menu, select **Properties**.

The *Package Builder Properties* dialog box appears.

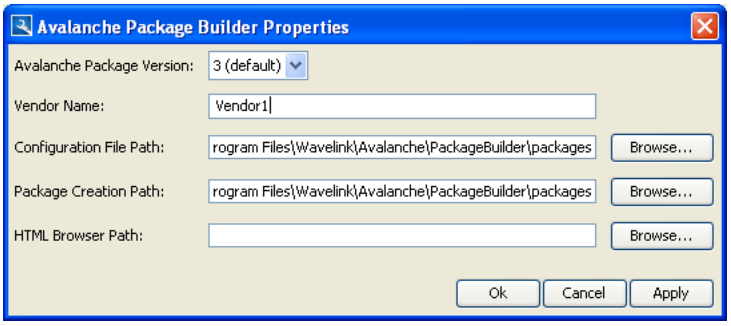

Figure 3-1. *Avalanche Package Builder Properties*

2 From the drop-down list, select the Avalanche Package Version.

- **Package Builder 1**. Use this version if you are running Avalanche Manager 3.2 or previous versions. Options, such as including retrievable files and script files, are disabled when using this version. This package builder is CTT driven.
- **Package Builder 3**. This version (default) is PPF driven and has more functionality than version 1, including support for retrievable files.

3 Enter a **Vendor Name**.

NOTE If you do not enter a Vendor Name, your package build will fail.

---

4 Enter the **Confirmation Path** and the **Package Creation File** path.

-or-

Click `Browse`, navigate to the path and click `OK` to select the file destination.

5 Enter the **HTML Browser Path**. If you do not enter an HTML browser, the default browser will be used.

6 Click OK to exit.

Entering Package Information

You can enter general information about the software package in the Package Information tab. For detailed information about the Package Information tab, refer to *Package Information Tab* on page 35.

To enter package information:

1 Click the Package Information tab. (Figure 3-2).

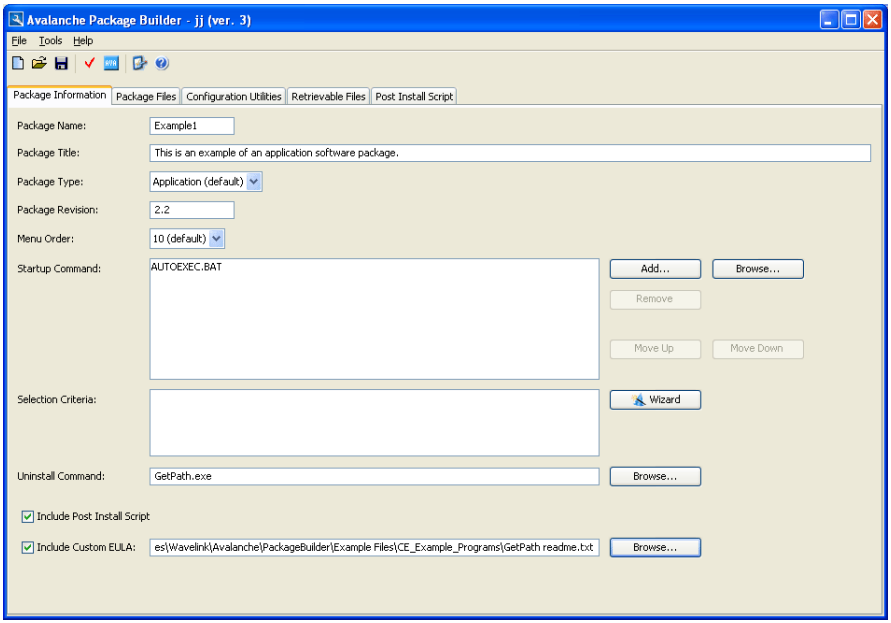

Figure 3-2. *Entering Package Information*

2 Enter the **Package Name**. This field is limited to eight characters.

3 Enter the **Package Title**. This field is limited to 80 characters.

4 Select the **Package Type** from the drop-down menu.

For detailed information on package types, refer to *Types of Avalanche Software Packages* on page 14.

5 Enter the **Package Revision** number. This field is limited to 15 characters.

6 Select the **Menu Order** number from the drop-down menu.

7 Use one of the following methods to enter a **Startup Command**:

- Click `Add` and type the command in the *Add Startup Command* dialog box.
- Click `Browse`, navigate to the command, and click `Open`.

NOTE A Startup Command is required for Application and AutoRun packages and can be executable, script, or batch files.

---

8 Click `Wizard` to open the *Selection Criteria Builder*.

9 Use the drop-down menu and buttons to create the selection criteria for the package.

NOTE For detailed information on the Selection Criteria Builder refer *Chapter 7: Selection Criteria Builder* on page 99.

---

10 Use one of the following methods to save the package information:

- Click the **Save** icon, enter the **File Name** of the package in the *Save As* dialog box, and click `Save`.
- From the **File** menu, select **Save**, enter the **File Name** of the package in the *Save As* dialog box, and click `Save`.

Optional Package Information Configurations

- **Uninstall Command**. If you want to enter an uninstall command, use one of the following methods**:**
  - Type the uninstall command in the **Uninstall Command** text box.
  - Click `Browse`, navigate to an uninstall command file, and click `Open`.

NOTE The uninstall command must be a .exe file for Windows and CE platforms. Version 3 DOS Enablers will support batch, COM, and executable files. The uninstall command must be included as a package file.

- **Include Post Install Script**. If you want to include a script (.ini) file in your package, select the **Include Post Install Script.** This enables the Post Install Script tab and allows you to create the script files.
- **Include Custom EULA**. If you want to include a EULA, select the **Include Custom EULA** check box and click `Browse` to locate the EULA file.

Adding Package Files

As part of the package definition, you must include the package files that will download to the mobile device.

You can use the **Package Files** tab to enter details about the files and select which files to include in the package. The tab is divided into two regions:

- **Default Files Properties**
- **Source File List**

For detailed information about the Package Files tab, refer to *Package File Tab* on page 41.

Default Files Properties

In this region, you can configure the properties (such as, installation drives and paths) that will apply to the package files.

To configure the Default Files Properties:

1 In the Package Builder, click the **Package Files** tab (Figure 3-3).

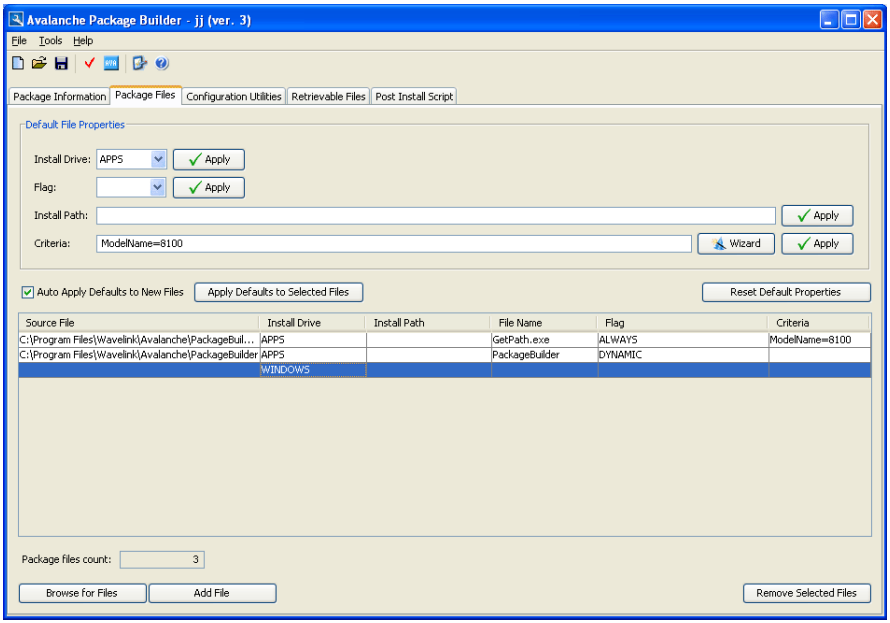

Figure 3-3. *Adding Package Files*

2 Select an **Install Drive** from the drop-down list.

3 Select a type of **Flag** from the drop-down list.

4 Enter the **Install Path**, relative to the **Install Drive.**

5 Click `Wizard,` to open the **Selection Criteria Builder**.

6 Use the drop-down menu and buttons to create the selection criteria for this package.

NOTE For detailed information on the **Selection Criteria Builder** refer to *Chapter 7: Selection Criteria Builder* on page 99.

Source File List

In this region, you can add package files, view the package files currently attached to software package, and apply default properties.

To add package files:

1 Use one of the following methods to insert the package files:

- Click `Browse for Files`, navigate to the desired files, and click `Add`.
- Click `Add Files` and enter file names manually in the blank Source File line.

2 Repeat step 1 to add additional files.

Optional Package File Configurations

- **Apply.** These buttons apply the associated field configuration to a selected file**.**
- **Auto Apply Defaults to New Files**. When this check box is selected, all files added to the Source List will be set with the default properties configured in the Default File Properties region.
- **Apply Defaults to Selected Files**. This button applies the defaults to selected files.
- **Reset Default Properties**. This button resets the Default File Properties fields to the original default properties.
- **Right-click**. You can right-click on the Source List to select all files.
- **Using Wildcards**. The "`*`" and the "`?`" wild cards are supported at the final directory destination and one directory above.
  - `*` represents one or more characters
  - `?` represents one character only

  A file path containing `*.txt` would include all files that end with .txt. A file path containing `\Work\*\file?.txt` would include all files under any subdirectory of Work that had `file<any character>.txt` as its filename (for example, `file1.txt`).

Adding Configuration Utilities

Configuration utilities are optional external utilities used to modify the client application. When you plug a configuration utility into the package definition, it will be incorporated into the Management Console for use by the operator.

NOTE You can access plug-in utilities from the Management Console by right-clicking on the software package and selecting `Configure Package`.

To add a configuration utility to the package definition:

1 In the Package Builder, click the Configuration Utilities tab (Figure 3-4).

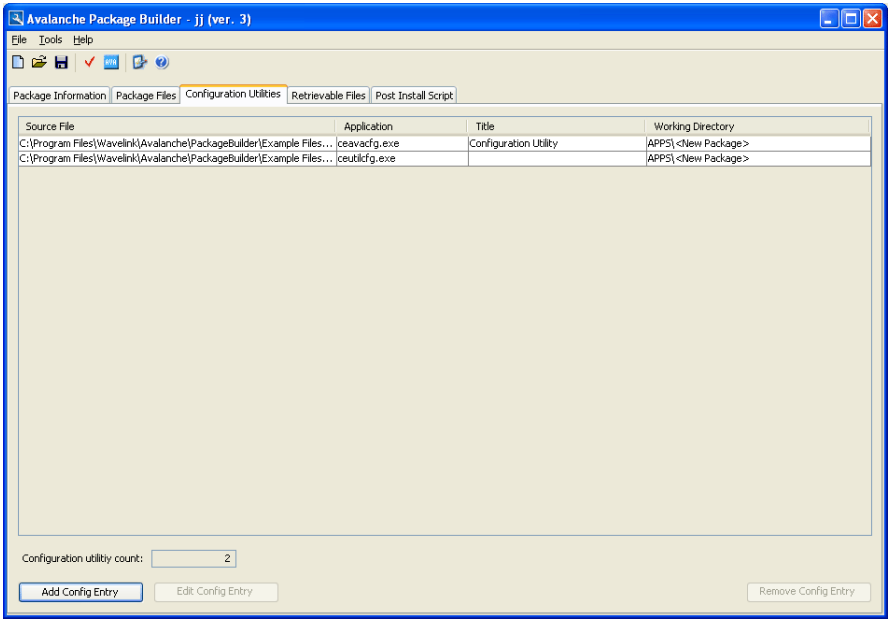

Figure 3-4. *Adding Configuration Files*

2 Click `Add Config Entry`.

The *Configuration Utility Editor* dialog box appears (Figure 3-5).

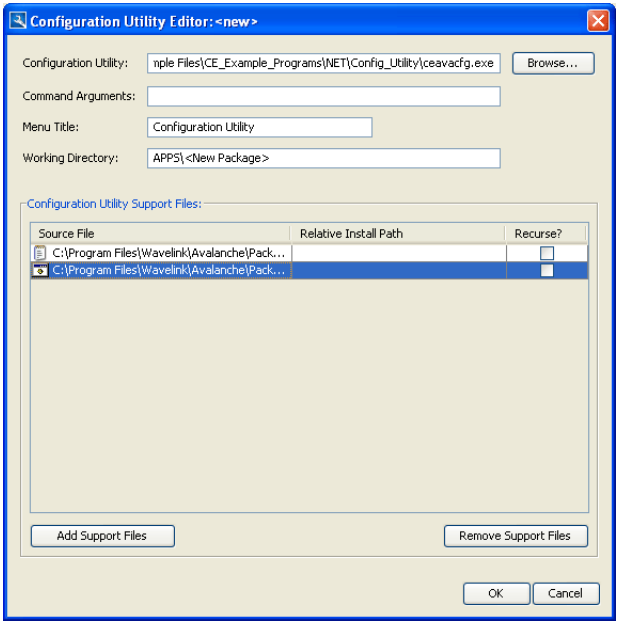

Figure 3-5. *Configuration Utility Editor*

3 Click `Browse`, navigate to a desired configuration file, and click `Select` to include a **Configuration Utility** file.

4 Click `Add Support Files`, navigate to the desired support file, and click `Add` to include any support files. (A dll is a good example of a support file.)

5 When you are finished adding support files, click `OK`.

The new configuration utility appears in the Configuration Utilities tab.

Optional Support File Configurations

- **Remove Selected Entry**. If you want to remove any configuration utility files, select the file in the Configuration Utilities tab and click `Remove Selected Entry`.
- **Using Wildcards**. The "`?`" wild cards are supported at the final directory destination and one directory above.
  - `*` represents one or more characters

- `?` represents one character only

A file path containing `*.txt` would include all files that end with .txt. A file path containing `\Work\*\file?.txt` would include all files under any subdirectory of Work that had `file<any character>.txt` as its filename (for example, `file1.txt`).

- **Command Line**. If you want to add command line arguments to a specific utility, select the utility within the Configuration Utilities tab, click on the `Edit Config Entry` button, and type the desired arguments in the **Command Arguments** field.

- **Menu Title**. If you want to add a name within the **Configure Package** context menu for the plug-in utility, select the utility within the Configuration Utilities tab, click on the `Edit Config Entry` button and type the desired title in the **Menu Title** field. There is no default name.

  This name appears as the utility name in the Management Console.

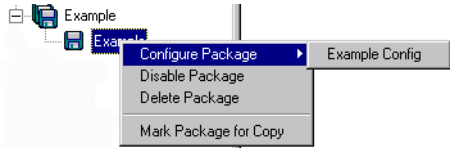

Figure 3-6 *Configure Package Context Menu (Avalanche Management Console)*

- **Working Directory**. If you want to set a working path as the current directory when the application runs, select the utility within the Configuration Utilities tab, click the `Edit Config Entry` button and type the desired working path in the **Working Path** field. The relative directory resides within the software package home directory.

  The package builder creates these directories automatically.

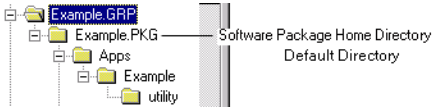

Figure 3-7 *Working Path*

Adding Retrievable Files

Adding retrievable files to your software package allows you to specify files that you want to retrieve from the mobile device. You can select where to save the files on the Agent and whether to delete the files from the mobile unit. Files are retrieved upon connection of the mobile device.

To add retrievable packages:

1 In the Package Builder, click the Retrievable Files tab (Figure 3-8).

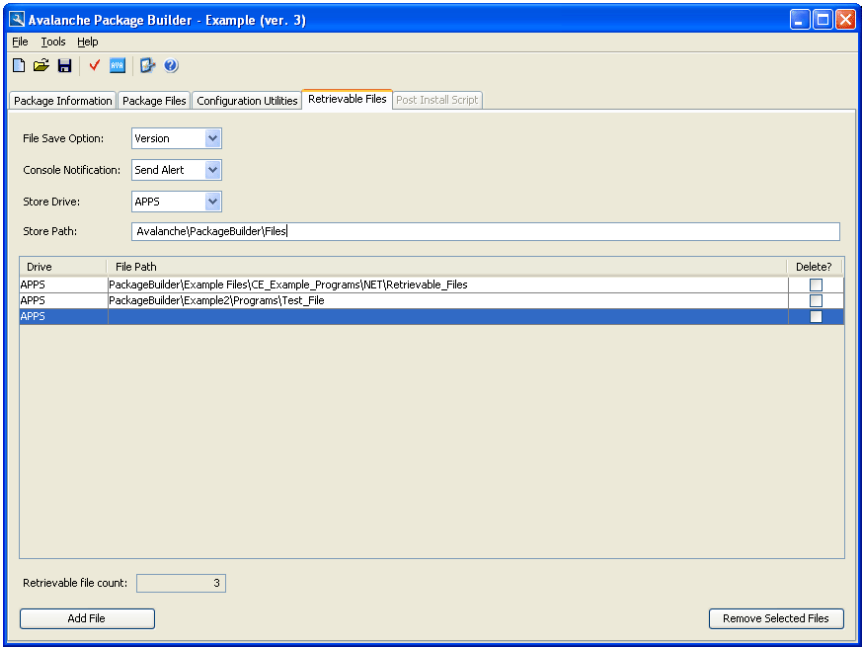

Figure 3-8. *Adding Retrievable Files*

2 From the drop-down list, select a **File Save Option**.

3 If you want an Avalanche Console notification, select a **Console Notification** from the drop-down list..

4 From the drop-down list, select the **Store Drive**.

5 Enter a **Store Path** relative to the **Store Drive.**

6 Click `Add Files` to add the retrievable files.

7 Enter the file information in the **File Path** text box.

Building a Post Install Script File

For complete information about building a Post Install Script file, refer to *Chapter 6: Post Install Script File* on page 77.

Validating a Package

When you validate the package, the Package Builder checks the package elements and then, if the package is valid, creates a package configuration file with a `.cfg` extension, placing it into the directory defined in the **Configuration File Path** field in the Package Builder Properties dialog box.

To validate a package:

1 In the Package Builder, click `Validate Package` from the **Tools** menu.

-or-

Click the **Validate Package** shortcut icon.

2 If the package is not previously named, enter a name for the package in the *Save As* dialog box that appears and click `OK`.

The *Build Status* dialog box appears (Figure 3-9 and Figure 3-10). This is a summary that describes the package features and lists any problems with the build, such as a missing client file.

If there are problems with the build, the *Build Status* dialog box displays the following message:

`[`*`Name of Software Package`*`] is not valid.`

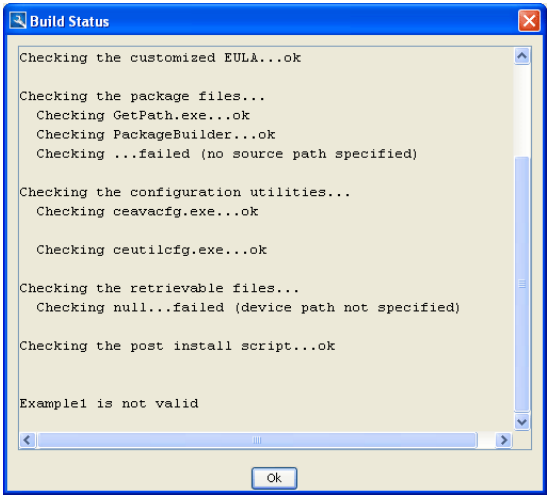

Figure 3-9. *Package Is Not Valid*

NOTE If your software package is not valid, review the Build Status summary to locate the portion of the package that failed.

If there are no problems with the build, the Build Status dialog box displays the following message:

`[Name of Software Package] is valid.`

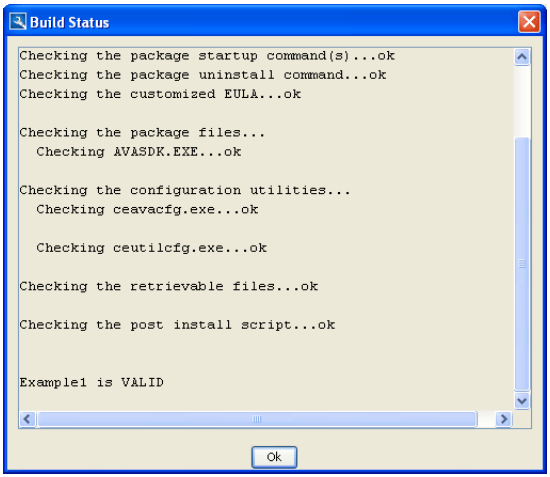

Figure 3-10. *Package Is Valid*

Making a Package

When you make a package, you create an `.ava` file that can be installed into Avalanche Manager.

To create an .ava file for a package:

1 Access the **Tools** menu and select `Make Package`.

-or-

Click the Make Package shortcut icon.

A dialog box appears that allows you to name the `.ava` file and select the location where you want to save the file.

2 Type the alpha-numeric name for the package.

3 Select the directory where you want to save the `.ava` file.

4 Click `Select`.

The `.ava` file is saved to the location that you selected.

You can now use the `.ava` file to install the package in Avalanche Manager.

Installing Packages

This section provides information about installing software packages in Avalanche Manager.

Installing the Package in Avalanche Manager

You use the .ava file to install the software package in Avalanche Manager.

To install the package in Avalanche Manager using the .ava file:

1. Launch the Management Console and connect to the Agent.
2. In the Management Console, access the **Software Management** menu and select `Install Software Package...`.

   The **Install Software Package Wizard** appears.
3. Type the path the `.ava` file for the package that you want to install, or use the browse button to search for and select the `.ava` file.
4. Click `Next`.
5. In the *License Agreement* dialog box that opens, select the **Yes, I agree** option.
6. Click `Next`.
7. In the *Select the Software Collection* dialog box, select the software collection in which you want to install the package.
8. Click `Next`.

   The **Install Software Package Wizard** displays the progress of the package installation.
9. After the installation is complete, click `Finish`.

   The package now appears in the **Management Console** beneath the software collection that you selected.

NOTE For more information about installing and using Avalanche software packages, see the help files that are available in **Help** menu of Avalanche Manager or see the Avalanche Enabler documentation relevant to your mobile device.

Modifying a Package

The Package Builder allows you to modify packages as needed. By default, the utility stores all packages in the `\<Package Builder Home Directory>\PackageBuilder\packages` subdirectory.

To modify an existing package:

1. Launch the Package Builder.
2. In the Package Builder, click `Open` from the **File** menu.
3. Navigate to and select the desired package configuration file (.cfg) and click `Open`. By default, package configuration files are located in the `\<Package Builder Home Directory>\Package Builder\packages` subdirectory.

   Once opened, the settings for the selected package appear in the Package Builder.
4. Modify all settings as desired.
5. Click `Validate Package`.
6. Click `Make Package`.
7. Click `Save` from the **File** menu to save updated package configuration file settings.

Chapter 4: Software Package Builder User Interface

This chapter provides detailed information about the menus and screen regions of the Avalanche Software Package Builder.

Menu Information

This section provides information about the File, Tools, and Help menus in the Avalanche Software Package Builder user interface.

File Menu

The commands available under the File menu are as follows:

New	Creates new Software Package directory structure and new Software Package Configuration Profile (.cfg).
Open	Opens an existing Software Package Configuration Profile.
Save	Saves the current Software Package Configuration Profile.
Save As	Saves the current Software Package Profile to the same name or a different name.
Exit	Closes the Package Builder.

Tools Menu

The commands available under the Tools menu are as follows:

Properties This command lists the package version, vendor name, configuration file path, package creation path, HTML Browser path. For more detailed information refer to Table **4-1.**

Validate Package This function verifies the integrity of the package. A successful validation is required in order to build the package. Some of the items verified are the name of the package, the version number, number of package files, and presence of configuration utilities.

Make Package This feature brings up a dialog box that allows you to save the package as an .ava file. You can use the .ava file to install the software package directly into Avalanche Manager.

Help Menu

The commands available under the Help menu are as follows:

Help Docs F1 Launch the Package Builder documentation within a browser.

About Display the current version of the Package Builder.

Tool Bar Icons

The Tool Bar features buttons that perform the following tasks:

Create new package. This option allows you to create a new software package

Open package. This option allows you to open an existing software package.

Save package. This option allows you to save the current software package.

Verify package. This function verifies the integrity of the package. A successful validation is required in order to build the package. Some of the items verified are the name of the package, the version number, number of package files, and presence of configuration utilities.

Build package. This feature brings up a dialog box that allows you to build the package as an .ava file. You can use the .ava file to install the software package directly into Avalanche Manager.

Edit package properties. This option allows you to edit the software package properties.

Help. This option launches the Package Builder User Manual within a browser.

Package Builder Screen Regions

This section provides information about the screen regions in the Avalanche Software Package Builder.

Package Builder Properties

You can access the *Package Builder Properties* dialog box from the **Tools** menu of Package Builder (Figure 4-1).

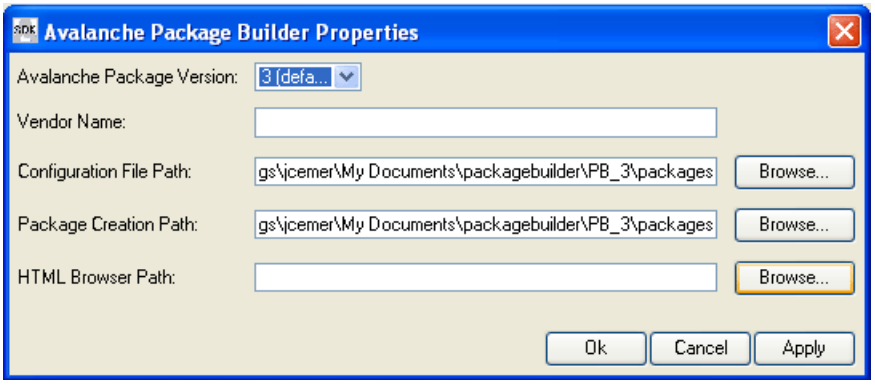

Figure 4-1. *Package Builder Properties*

Table 4-1 describes the fields and options in the *Package Builder Properties* dialog box.

Field	Description
Avalanche Package Version	This is the version of package. You can select to build version 3 or version 1 packages. Some features of version 3 packages (such as retrievable files) are not available for use with version 1. You should use Package Builder version 1 if you are using Avalanche Manager 3.2 or previous versions. Default: Version 3
Vendor Name	This feature passes the version name to the Avalanche Manager. It is displayed when the user clicks on the third party package within the Manager. This is a required field.
Configuration File Path	This is the path where the Package Builder profile (.cfg) file resides. This file contains the definition of the software package. This is a required field.

Table 4-1: *Package Builder Properties*

Field	Description
Package Creation Path	This is the path where the Package Builder places the Avalanche software package directory structure. For example, a package named Example would result in a directory structure called Example.pkg to reside in this directory. This is a required field.
HTML Browser Path	This is the path of the web browser used to launch product documentation. Default: Your default web browser.

Table 4-1: *Package Builder Properties*

NOTE Once the package properties are set, they are saved until modified again. You will not need to enter this information again, unless you want to change the values.

Package Information Tab

You can enter the basic software package information Package Information Tab (Figure 4-2).

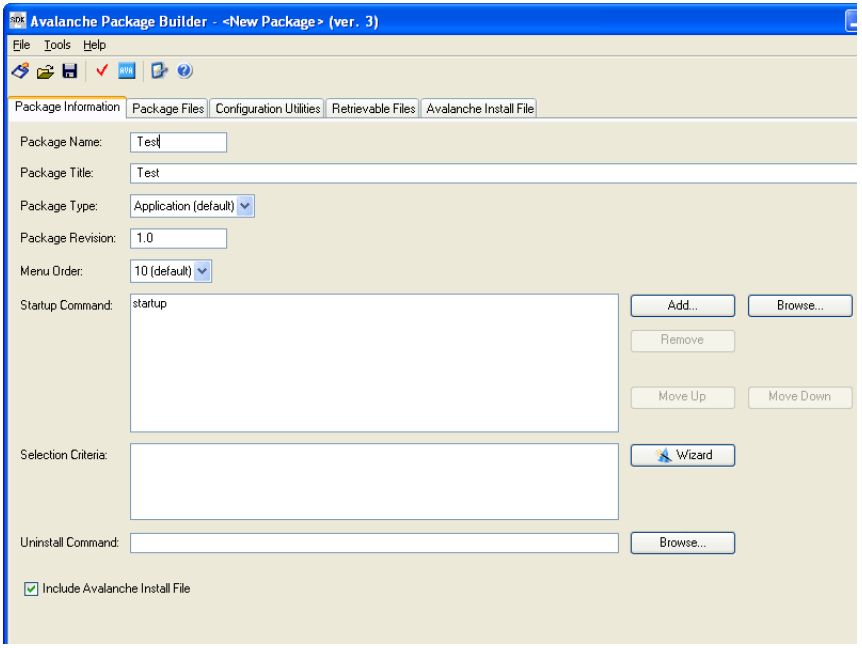

Figure 4-2. *Package Information Tab*

Table 4-2 describes the fields and options in the Package Information tab.

Field	Description
Package Name	This name determines the “internal name” of the package, must be eight characters or less in length, and must contain only valid characters for a DOS file name. (This will ensure compatibility with all mobile device platforms.) The internal name (i.e.,.PKG directory) determines the package name that appears in the Tree View of the Avalanche Manager and the directory name for the application on the flash drive of the mobile device.
Package TItle	This is the descriptive title of the application, consisting of any “displayable” characters. It is recommended that you limit the title to 80 characters or less to allow the full title to display on smaller mobile device screens.

Table 4-2: *Package Information Tab*

Field	Description
Package Type	This is the package type. Select one of the following values: **Application.** These packages will be used by the end user and can be selected from the Application Menu screen on the mobile device. An example of an Application package is the Telnet Client. **Support**. These packages deliver support files. The end user will not use these files and the package does not add new items to the Application Menu screen on the mobile device. An example of a Support file is an update to an existing file. **Auto**. These packages run automatically after download and are “one-time only” packages, meaning that once the files are installed, they are not used again. An example of an auto run package is a firmware update. **Config**. These packages contain a configuration utility that will modify the files being retrieved. An example of a Config package is a Text editor to modify a text file. **NOTE**: Config packages require special Enabler support to work. For information about building a Config package type, please contact Wavelink Customer Service. Default: Application
Package Revision	The number or string that displays the package version number. This field has a 15 character limit. **NOTE:** The Avalanche Manager automatically generates cookies to track package changes. It uses cookies, rather than the revision code, to determine whether the mobile device needs a package update. This field is required.
Menu Order	A number used to sort applications in the Application Menu screen on the mobile device. Lower-numbered entries appear at the top of the screen. This number determines the placement of your application within the DOS Enabler menu. This number has no effect in the CE Avalanche environment. This field is required.
Startup Command	This command (or set of ordered commands) is the file that is executed when the package is launched by the Enabler. This field is required for Application and Autorun package types.

Table 4-2: *Package Information Tab*

Field	Description
Selection Criteria Builder Wizard	This field allows you to create selection criteria. For more information see *Chapter 7: Selection Criteria Builder* on page 99.
Uninstall Command	For Windows and CE platforms, the Uninstaller Command must be an executable built for the targeted platform/ processor. When completed, the command (executable) must have a return code of 0 (zero) or else the package will not uninstall. The Version 3 DOS Enabler supports executable, batch and COM files. The uninstall file must be included as part of the package files for the uninstall command to function. This field is optional.
Include Post Install Script	Enables the Post Install Script file tab. This option allows you to create a script file as part of the package. For more information, refer to *Chapter 6: Post Install Script File* on page 77.
Include Custom EULA	This option allows you to include a custom End User License Agreement file. The manufacturer EULA must be available as an HTML formatted `License.txt` file. HTML formatting tags should be used to make the EULA information readable. This document does not have complete HTML capabilities and does not support links and embedded objects. Example: `<HTML> //Required` `<H3>LIcense Agreement</H3> //Bold Title` `<p> //new paragraph` `Legal information starts here` `  //New line` `New line of legal information` `<p>` `<H3>Section 2</H3>` `<p>` `More information` `</HTML> //Required` This option is optional.

Table 4-2: *Package Information Tab*

Selection Criteria Builder

The Selection Criteria Builder (Figure 4-3) allows you to create a set of rules called selection criteria, which you can apply to individual software collections and individual network profiles, define which mobile devices will receive designated updates. The Selection Criteria Builder allows you to create criteria by inputting selection variables, operators, and actual values.

NOTE For more information, detailed descriptions, and examples of the Selection Criteria Builder elements, refer to *Chapter 7: Selection Criteria Builder* on page 99.

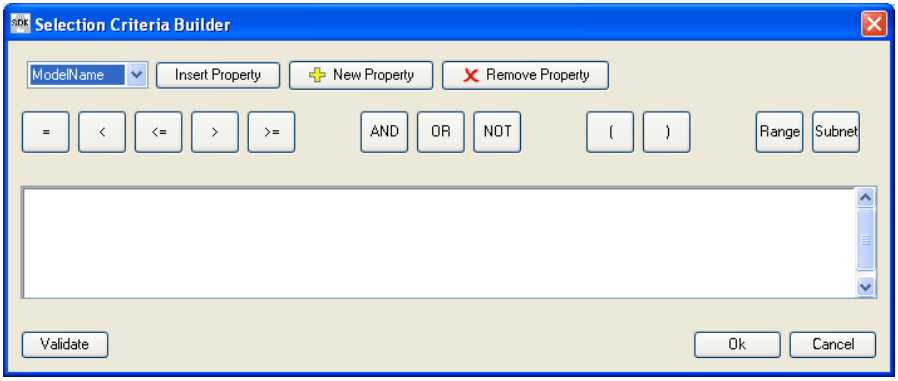

Figure 4-3. *Selection Criteria Builder*

Table 4-3 describes the selection criteria variables you can use in the Selection Criteria Builder.

Variable	Description
`ModelName`	The standard model name for a mobile device.
`ModelCode`	A number set by the device manufacturer and used internally by the BIOS to identify the hardware.
`Series`	The general series of a device.
`KeyboardName`	A string depicting which style of keyboard the mobile device is using (46key, 35key etc.).
`KeyboardCode`	A number set by the device manufacturer and used internally by the BIOS to identify the keyboard type.

Table 4-3: *Selection Criteria Variables*

Variable	Description
Rows	The number of display rows the mobile device supports.
Columns	The number of display columns the mobile device supports.
IP	IP address of the mobile device.
MAC	MAC address of the mobile device.
Group	Mobile device groups can be created and designed within Avalanche.
LastContact	The parser for the LastContact property allows specifying absolute time stamps and relative time stamps. This forces constant re-evaluation as the time base changes.

Table 4-3: *Selection Criteria Variables*

Table 4-4 describes the variable operators you can use in the Selection Criteria Builder.

Operator	Description
NOT (!)	Unary operator that negates the boolean value that follows it.
AND (&)	Binary operator that results in TRUE if and only if the expressions before and after it are also both TRUE.
OR (|)	Binary operator that results in TRUE if either of the expressions before and after it are also TRUE.
RANGE (-)	Binary operator allows for a range of values such as IP addresses.
SUBNET (\)	Binary operator allows for a comparison with subnets or CIDR notation.
(=)	Binary operator that results in TRUE if the two expressions on either side of it are equivalent.
>	Binary operator that results in TRUE if the expression on the left is greater than the expression on the right.
<	Binary operator that results in TRUE if the expression on the left is less than the expression on the right.
>=	Binary operator that results in TRUE if the expression on the left is greater than or equal to the expression on the right.
<=	Binary operator that result in TRUE if the expression on the left is less than or equal to the expression on the right.

Table 4-4: *Selection Criteria Operators*

NOTE For more information, detailed descriptions, and examples of the Selection Criteria Builder elements, refer to *Chapter 7: Selection Criteria Builder* on page 99.

Package File Tab

The Package File tab allows you to create the lists the files to be downloaded to the mobile device (Figure 4-4).

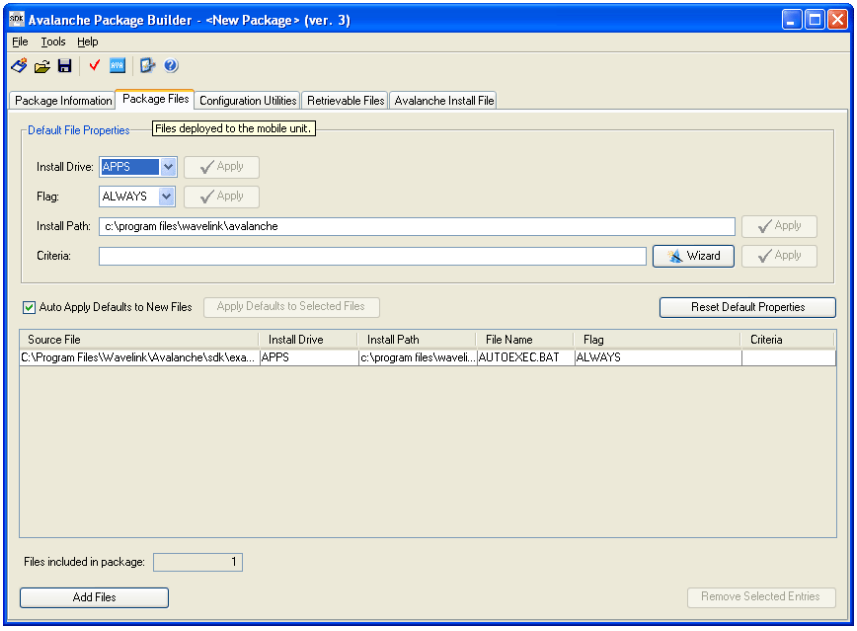

Figure 4-4. *Package File Tab*

Table 4-5 describes the fields and options in the Package File tab.

Field	Description
Install Drive	When you select the download files, you can optionally set the directories where the Package Builder will load the files. Refer to the Table 4-6 for more information. Default: APPS

Table 4-5: *Package File Tab*

Flag	This determines how the files will download. **Always**. Files are always downloaded upon connection. **Optional**. Files are not required to be present when building packages. **Dynamic**. Files can be modified by Configuration Utility without breaking package license. Default: No flag
Install Path	This is the installation path, relative to the Install Drive. This field is required if you are using a drive letter as an Install Drive.
Selection Criteria Builder Wizard	This Wizard allows you to build criteria for your software packages. Refer to the *Chapter 7: Selection Criteria Builder* on page 99 for more information. This field is optional.
Auto Apply Defaults to New Files	When you click `Auto Apply Defaults`, all default settings will be applied to new files added to the Package File tab.
Browse to Files	This button allows you to add files by browsing for the location of the files. Click this button, navigate the location of the files and select the file you want to add.
Add Files	This button allows you to add files manually. Click this button and enter the file path, flags, etc. in the blank text box. You can also use wildcards in the final directory. For example: `\program files\work\*.bat` would include all batch files within the `.\workdirectory. *.*`, `Filename.*`, and `*.extension` are all permitted.
Remove Selected Entries	This button allows you to remove entries from the software package. Select the entry you want to move and click Remove Selected Entries.

Table 4-5: *Package File Tab*

Table 4-6 provides information on each of the Install Drive options:

Install Drive	Description
APPS	The location on the mobile device under which software packages typically reside, for example, the package install directory. The default location for a client file is APPS.
WINDOWS	The WINDOWS drive is the windows system directory on WinCE devices
WORKS	The WORK drive is typically a location for temporary files.

Table 4-6: *Install Drive Types*

Install Drive	Description
AVA	The AVA drive is the location where the Avalanche Enabler is installed.
A, B, C, D, etc.	Single-letter directory names represent specific drive letters within the mobile device. Use these drive letters if a file needs to be placed in an absolute path within the mobile device. A drive letter must have an associated Install Path to be complete.

Table 4-6: *Install Drive Types*

Configuration Utilities Tab

Configuration files are utilities that modify the software package. The Configuration Utilities tab (Figure 4-5) allows you to select configuration files and attach them to the package. Each entry lists an external application to be plugged into the Management Console for use by the operator.

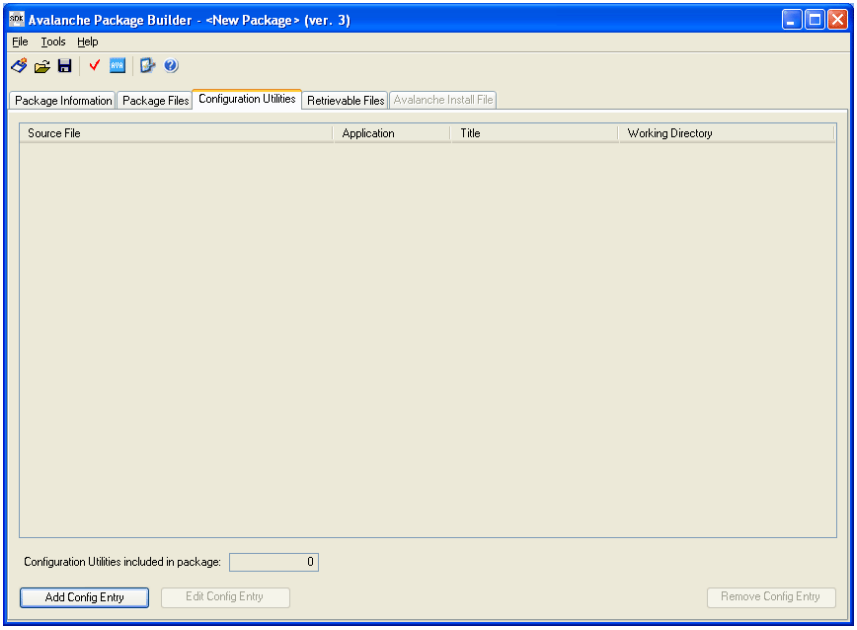

Figure 4-5. *Configuration Utilities*

Table 4-7 describes the options in the Configuration Utilities tab.

Field	Description
Add Config Entry	This button opens the *Configuration Utilities Editor* dialog box. This dialog box allows you select the Config Entry you want to attach to the software package. Refer *Configuration Utility Editor* on page 44 to for more information.
Edit Config Entry	This button allows you to edit the Config Entry.
Remove Config Entry	This option allows you to remove Config Entry files from the software package.

Table 4-7: *Configuration Utilities*

Configuration Utility Editor

The Configuration Utility Editor (Figure 4-6) allows you add to support files to the software package.

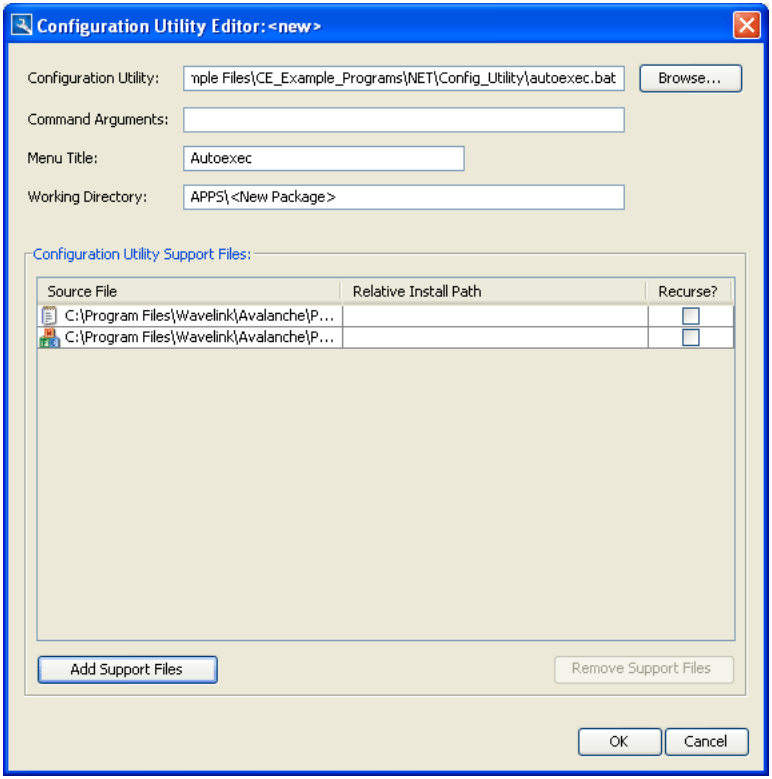

Figure 4-6. *Configuration Utility*

Table 4-8 describes the fields and options in the *Configuration Utilities Editor* dialog box.

Field	Description
Configuration Utility	This button allows you to navigate to and select the configuration utility file. This field is required.
Command Argument	You can add command line arguments to the configuration utility in this text box. This field is optional.
Menu Title	This text box adds a name for the plug-in utility. There is no default name. This field is required.
Working Directory	This text box sets a working path as the current directory when the application runs. The default directory for the utility is the software package home directory plus the name of the package. The home directory is used when the Working Path is left blank.
Add Support Files	This button allows you to navigate to and add support files.
Remove Support Files	This button removes support files associated with a configuration utility file.

Table 4-8: *Configuration Utility Editor*

Retrievable Files Tab

In the Retrievable Files tab (Figure 4-7), you can include retrievable files to the software package. These files will be retrieved from the mobile device upon connection to the Avalanche Manager.

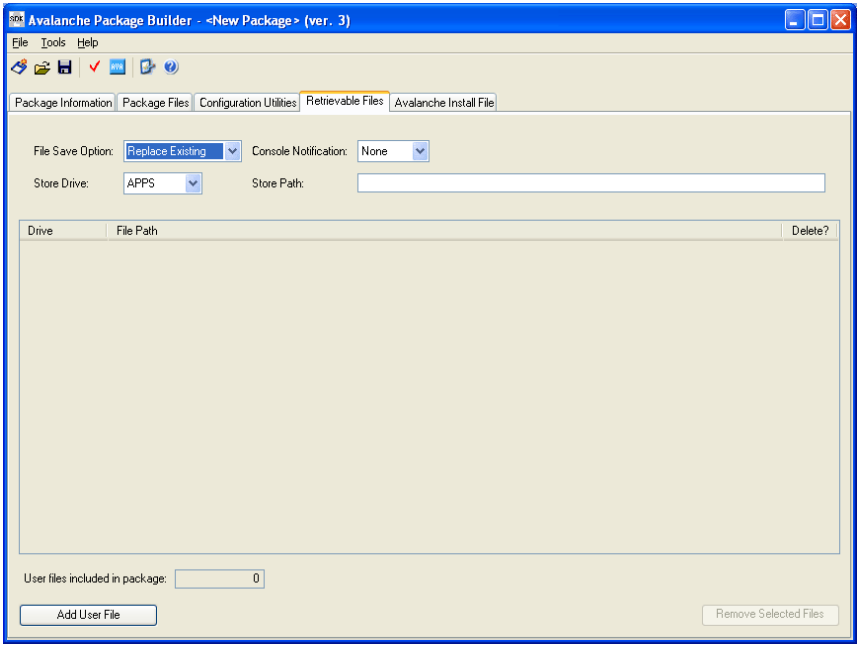

Figure 4-7. *Retrievable Files*

Table 4-9 describes the fields and options in the Retrievable Files tab.

Field	Description
File Save Option	There are three methods you can use to save the retrieved files: **Replace Existing**. This method replaces all previous files. **Append to Existing**. This method allows keeps the previous files and adds new files. **Version**. This method saves the files by versions. Default: Version
Console Notification	When this option is selected, a notification is sent to the Avalanche Console when files are retrieved from the mobile device. Default: None

Table 4-9: *Retrievable Files*

Store Drive	When you select the retrievable files, you can optionally set the directories where the Package Builder will load the files. For more information on each of the drives refer to Table 4-6. Default: APPS
Store Path	This is the specific path, relative to the Store Drive, where the Package Builder will load the files. The Store Path can contain one or more substitution variables that will be replaced with existing real values. The substitution variables are case-sensitive. If a substitution value does not exist, it will be replaced by an empty (null-length) string. This allows the generation of unique repositories for each MU. Variables: %m = The MUs MAC address (supported in Avalanche 3.4 or later versions) %g = The MUs GUID (supported in Avalanche 3.4 or later versions) %t = The MUs TerminaID string (support in Avalanche 3.5 or later versions) %i = The MUs reported IP address. (supported in Avalanche 3.5 or later versions) This field is required if you are using a drive letter as an Install Drive.
Add	This button allows you to add retrievable files to the software package.
Remove	This button allows you to remove selected files from the software package.

Table 4-9: *Retrievable Files*

Post Install Script File Tab

You can use the Post Install Script tab (Figure 4-8) to build script files.

Refer to *Chapter 6: Post Install Script File* on page 77 for detailed information on creating script files.

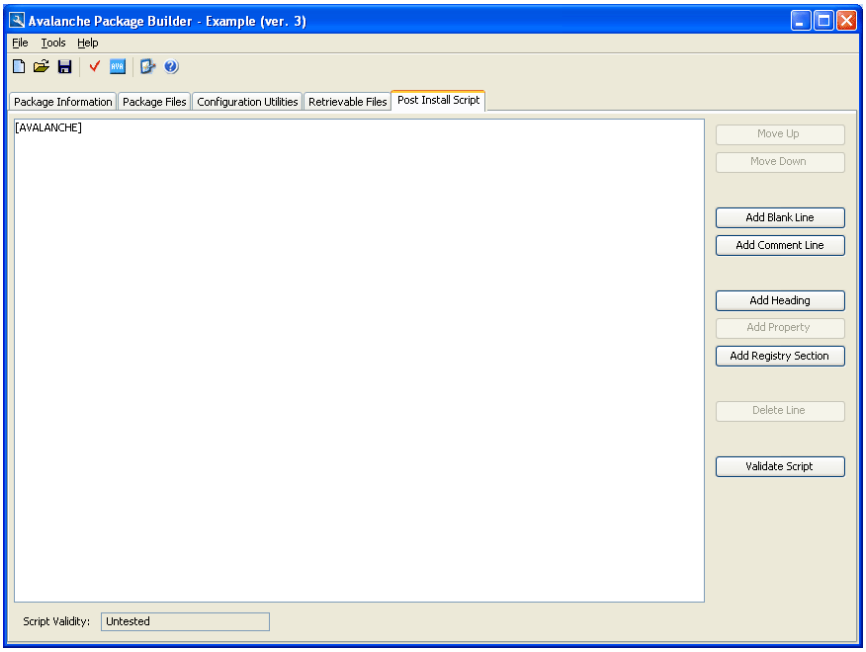

Figure 4-8. *Script File Builder*

Table 4-10 describes the options in the Post Install Script tab.

Field	Description
Move Up	This button allows you to move a line of code up.
Move Down	This button allows you to move a line of code down.
Add Blank Line	This button allows you to add a blank line in the text box.
Add Comment Line	This button allows you to add a line for comments in the text box.
Add Heading	This button allows you to select a header from a drop-down list or to create a custom header. Custom headers allow for future additions to the capabilities of the script file, however do not put in unknown values as this may have unpredictable results.
Add Property	This button allows you to define the Key and Value of a new property
Add Registry Setting	This button allows you to create Registry Keys and define keys and values for that key.

Delete Line	This button allows you to delete a line of code in the text box.
Validate Script	This button tests the script to see if it meets the specifications.

Table 4-10: *Post Install Script*

Chapter 5: Example Packages

This chapter provides examples for the following packages:

- *Application Package Example*
- *Auto Run Package Example*
- *Support Package with Post Install Script Example*
- *Configuration Utility Example*
- *Retrievable File Example*

The step-by-step instructions allow you build the sample packages in the Package Builder. Example files used to build the sample packages are located in the folder where you installed Package Builder. (The default installation is C:\Program Files\Wavelink\Avalanche\PackageBuilder.)

NOTE For information about building a Config package type, please contact Wavelink Customer Service.

Application Package Example

This example provides information for building a sample Application package containing a getpath file. To build this sample, complete the following tasks:

- Enter Package Properties
- Enter Package Information
- Add Package Files
- Validate the package
- Make the package
- Install the package in Avalanche Manager

Enter Package Properties

1 From the **File** menu, select `New`.

2 From the **File** menu, select `Save As`.

3 Name the package `getpath` and click `Save As` to save the .cfg file. This allows the developer to retrieve the configuration for this package.

4 From the **Tools** menu, select `Properties`.

5 Select `Version 3` from the **Avalanche Package Version** drop-down list.

6 Enter `Path Today` in the **Vendor Name** text box.

7 Enter `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` in the **Configuration File Path** text box.

   This is where the `getpath.cfg` file resides. This is the default location when the Package Builder installation path is used.

8 Enter `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` in the **Package Creation Path** text box.

   This is where the package components used to make the AVA package resides. In this case, you will see a `getpath.PKG` directory within this folder after the package has been made.

Summary

The following is a summary of the Package Properties:

- Avalanche Package Version: `3`
- Vendor Name: `Path Today`
- Configuration File Path: `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages`
- Package Creation Path: `C:\Program Files\Wavelink\Avalanche\Package Builder\packages`

Enter Package Information

1 In the Package Information tab, enter `getpath` in the **Package Name** text box.

2 Enter `Get Path` in the **Package Title** text box.

3 From the **Package Type** drop-down list, select `Application`.

4 Enter `1.00` in the **Package Revision** text box.

5 Leave the **Menu Order** drop-down at the default selection `(10)`.

This number determines the placement of your application within the DOS Enabler menu. This number has no effect in the CE Avalanche environment, however it is a required parameter.

6 Enter `getpath.exe` in the **Startup Command** text box.

7 Enter `Series = C` in the **Selection Criteria** text box. You can enter this information manually or use the Wizard.

Summary

The following is a summary of the Package Information:

- Package Name: `getpath`
- Package Title: `Get Path`
- Package Type: `Application`
- Package Revision: `1.00`
- Menu Order: `10`
- Startup Command: `getpath.exe`
- Selection Criteria: `Series = C`

Add Package Files

1 Select the Package Files tab.

2 To add WM2003 version of `getpath.exe`, click `Browse to Files`, navigate to and select `C:\Program`

`Files\Wavelink\Avalanche\PackageBuilder\Example Files\CE_Example_Programs\WM2003\File_Placement` (if the default installation path was used.)

The file will appear in the Source File list.

3 To add .NET version of `getpath.exe`, browse to `C:\Program Files\Wavelink]Avalanche\PackageBuilder\Example Files\CE_Example_Programs\NET\File_Placement` (if the default installation path was used.).

The file will appear in the Source File list.

Summary

The following is a summary of Package Files:

- Source File: `C:\Program Files\Wavelink\Avalanche\PackageBuilder\Example Files\CE_Example_Programs\WM2003\File_Placement\GetPath.exe.`
- Install Drive: APPS
- Install Path:
- File Name: `getpath.exe`
- Flag:
- Selection Criteria:

Validate the Package

- From the **Tools** menu, select `Validate Package` or press `F6`.

  The Build Status should read "`getpath is Valid`."

Make the Package

1 From the **Tools** menu, select `Make Package` or press `F7`.

A *Save* dialog box appears

2 Select C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages as the Save in folder.

3 Enter `getpath` as the file name.

4 Click `Save`.

5 The Build Status should read "Package build was successful."

`getpath.AVA` should now reside in `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` folder.

6 From the **File** menu, select `Save` to save the package configuration.

Install the Package

1 Install `getpath.ava` in a software collection entitled **GetPath** within the Avalanche Manager.

2 Enable the package and download the getpath package to the mobile device.

3 Launch the GetPath package from the Avalanche desktop.

4 Click `Get Application Path` to acquire the path and file name.

The path should read `Files\Wavelink\Avalanche\Apps\getpath\GetPath.exe`. You will need to scroll within the display field to see the whole path.

Auto Run Package Example

This example provides information for building a sample Auto Run package. To build this sample, complete the following tasks:

- Enter Package Properties
- Enter Package Information
- Add Package Files
- Validate the package
- Make the package

- Install the package in Avalanche Manager

Enter Package Properties

1 From the **File** menu, select `New`.

2 From the **File** menu, select `Save As`.

3 Name the package `getauto` and click `Save As` to save the .cfg file. This allows the developer to retrieve the configuration for this package.

4 From the **Tools** menu, select `Properties`.

5 Select `Version 3` from the **Avalanche Package Version** drop-down list.

6 Enter `Auto Town` in the **Vendor Name** text box.

7 Enter `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` in the **Configuration File Path** text box.

   This is where the `getauto.cfg` file resides. This is the default location when the Package Builder installation path is used.

8 Enter `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` in the **Package Creation Path** text box.

   This is where the package components used to make the AVA package reside. In this case, you will see a `getauto.PKG` directory within this folder after the package has been made.

Summary

The following is a summary of the Package Properties:

- Avalanche Package Version: `3`
- Vendor Name: `Auto Town`
- Configuration File Path: `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages`

- Package Creation Path:
  `C:\Program Files\Wavelink\Avalanche\Package Builder\packages`

Enter Package Information

1. In the Package Information tab, enter `getauto` in the **Package Name** text box.
2. Enter `Get Path Auto` in the **Package Title** text box.
3. From the **Package Type** drop-down list, select `Auto Run`.
4. Enter `1.00` in the **Package Revision** text box.
5. Leave the **Menu Order** drop-down at the default selection `(10)`.

   This number determines the placement of your application within the DOS Enabler menu. This number has no effect in the CE Avalanche environment, however it is a required parameter.
6. Enter `getpath.exe` in the **Startup Command** text box.
7. Enter `Series = C` in the **Selection Criteria** text box. You can enter this information manually or use the Wizard.

Summary

The following is a summary of the Package Information:

- Package Name: `getauto`
- Package Title: `Get Path Auto`
- Package Type: `Auto Run`
- Package Revision: `1.00`
- Menu Order: `10`
- Startup Command: `getpath.exe`
- Selection Criteria: `Series = C`

Add Package Files

1 Select the Package Files tab.

2 To add WM2003 version of `getpath.exe`, click `Browse to Files`, navigate to and select `C:\Program Files\Wavelink\Avalanche\PackageBuilder\Example Files\CE_Example_Programs\WM2003\File_Placement` (if the default installation path was used.)

The file will appear in the Source File list.

3 To add .NET version of `getpath.exe`, browse to `C:\Program Files\Wavelink]Avalanche\PackageBuilder\Example Files\CE_Example_Programs\NET\File_Placement` (if the default installation path was used.).

The file will appear in the Source File list.

Summary

The following is a summary of Package Files:

- Source File: `C:\Program Files\Wavelink\Avalanche\PackageBuilder\Example Files\CE_Example_Programs\WM2003\File_Placement\GetPath.exe.`
- Install Drive: APPS
- Install Path:
- File Name: `getpath.exe`
- Flag:
- Selection Criteria:

Validate the Package

- From the **Tools** menu, select `Validate Package` or press `F6`.

The Build Status should read "`getauto is Valid`."

Make the Package

1 From the **Tools** menu, select `Make Package` or press `F7`.

A *Save* dialog box appears.

2 Select `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` as the Save in folder.

3 Enter `getauto` as the file name.

4 Click `Save`.

5 The Build Status should read "`Package build was successful.`"

The `getauto.ava` file should now reside in `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` folder.

6 From the **File** menu, select `Save` to save the package configuration.

Install the Package

1 Install `getauto.ava` in a software collection entitled **GetAuto** within the Avalanche Manager.

2 Enable the package and download the getpath package to the mobile device.

3 From the **File** menu on the Enabler desktop, select **Connect** to receive the getauto package.

The `GetPath.exe` file will launch automatically. Auto Run packages do not display icons on the Avalanche Enabler desktop.

NOTE An Auto Run package will only launch once.

Support Package with Post Install Script Example

This example provides information for building a sample Support Package with a Post Install Script. To build this sample, complete the following tasks:

- Enter Package Properties
- Enter Package Information
- Create Post Install Script
- Validate the package
- Make the package
- Install the package in Avalanche Manager

Enter Package Properties

1. From the **File** menu, select `New`.
2. From the **File** menu, select `Save As`.
3. Name the package `script1` and click `Save As` to save the .cfg file. This allow the developer to retrieve the configuration for this package.
4. From the **Tools** menu, select `Properties`.
5. Select `Version 3` from the **Avalanche Package Version** drop-down list.
6. Enter `Script People` in the **Vendor Name** text box.
7. Enter `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` in the **Configuration File Path** text box.

   This is where the script1.cfg file resides. This is the default location when the Package Builder installation path is used.
8. Enter `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` in the **Package Creation Path** text box.

This is where the package components used to make the AVA package resides. In this case, you will see a script1.PKG directory within this folder after the package has been made.

Summary

The following is a summary of the Package Properties:

- Avalanche Package Version: `3`
- Vendor Name: `Script People`
- Configuration File Path:
  `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages`
- Package Creation Path:
  `C:\Program Files\Wavelink\Avalanche\Package Builder\packages`

Enter Package Information

1. In the Package Information tab, enter `script1` in the **Package Name** text box.
2. Enter `Post Install Script1` in the **Package Title** text box.
3. From the **Package Type** drop-down list, select `Support`.
4. Enter `1.00` in the **Package Revision** text box.
5. Leave the **Menu Order** drop-down at the default selection `(10)`.

   This number determines the placement of your application within the DOS Enabler menu. This number has no effect in the CE Avalanche environment, however it is a required parameter.
6. Enter `getpath.exe` in the **Startup Command** text box.
7. Enter `Series = C` in the **Selection Criteria** text box. You can enter this information manually or use the Wizard.
8. Select the **Include Post Install Script** check box.

Summary

The following is a summary of the Package Information:

- Package Name: `script1`
- Package Title: `Post Install Script1`
- Package Type: `Support`

- Package Revision: `1.00`
- Menu Order: `10`
- Selection Criteria: Series = C
- Include Post Install Script: Enabled

Create Post Install Script

1. Select the Post Install Script tab.
2. Click `Add Blank Line`.
3. Click `Add Heading`.
4. Select `[EXECUTE]` from the **Value** drop-down list and click `OK`.
5. Click `Add Property`.
6. Enter `"reboot"` (include quotes) in the **Key** text box.
7. Enter `"yes"` (include quotes) in the **Value** text box.
8. Click `OK`.

   The property appears in the text box.
9. Click `Add Property`.
10. Enter `"RebootType"` (include quotes) in the **Key** text box.
11. Enter `"warm"` (include quotes) in the **Value** text box.
12. Click `OK`.

    The property appears in the text box.
13. Click `Add Blank Line`.
14. Click `Add Registry Section`.
15. Enter `HKEY_LOCAL_MACHINE\Software\testtoday\script1` in the Registry Key text box.

NOTE Do not enter the [] brackets. These are added automatically.

16 Enter the following information in the **Key** and **Value** text boxes.

Key	Value
"InstallDir"	"\platform\script1"
"RegFilePath"	"\Application"
"ConfigDir"	"\Application\Avalanche\script1"

17 Click OK.

18 Click Validate Script.

The Script Validity message box should indicate the script is valid.

Validate the Package

- From the **Tools** menu, select Validate Package or press F6.

  The Build Status should read "script1 is Valid."

Make the Package

1 From the **Tools** menu, select Make Package or press F7.

A *Save* dialog box appears

2 Select C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages as the Save in folder.

3 Enter script1 as the file name.

4 Click Save.

5 The Build Status should read "Package build was successful."

The script1.ava file should now reside in C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages folder.

6 From the **File** menu, select `Save` to save the package configuration.

Install the Package

1 Install `script1.ava` in a software collection entitled **Scripts** within the Avalanche Manager.

2 Enable the package and download the getpath package to the mobile device.

3 From the **File** menu on the Enabler desktop, select `Connect` to receive the script1 package.

Support packages do not display icons on the Avalanche Enabler desktop.

A *Reboot Required* dialog box appears.

4 Click `Yes` to perform the warm boot that was defined in the Post Install Script file

5 After the reboot, use a registry editor, such as Microsoft Embedded C++4.0 IDE, to view the registry key and associated sub-keys that were added.

Configuration Utility Example

This example provides information for building a sample package containing a Configuration Utility. To build this sample, complete the following tasks:

- Enter Package Properties
- Enter Package Information
- Add Package Files
- Add Configuration Utilities
- Validate the package
- Make the package
- Install the package in Avalanche Manager

Enter Package Properties

1 From the **File** menu, select `New`.

2 From the **File** menu, select `Save As`.

3 Name the package `ceutil` and click `Save As` to save the .cfg file. This allows the developer to retrieve the configuration for this package.

4 From the **Tools** menu, select `Properties`.

5 Select `Version 3` from the **Avalanche Package Version** drop-down list.

6 Enter `Config City` in the **Vendor Name** text box.

7 Enter `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` in the **Configuration File Path** text box.

   This is where the `ceutil.cfg` file resides. This is the default location when the Package Builder installation path is used.

8 Enter `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` in the **Package Creation Path** text box.

   This is where the package components used to make the AVA package resides. In this case, you will see a `ceutil.PKG` directory within this folder after the package has been made.

Summary

The following is a summary of the Package Properties:

- Avalanche Package Version: `3`
- Vendor Name: `Config City`
- Configuration File Path: `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages`
- Package Creation Path: `C:\Program Files\Wavelink\Avalanche\Package Builder\packages`

Enter Package Information

1 In the Package Information tab, enter `ceutil` in the **Package Name** text box.

2 Enter `Config Utility` in the **Package Title** text box.

3 From the **Package Type** drop-down list, select `Application`.

4 Enter `1.00` in the **Package Revision** text box.

5 Leave the **Menu Order** drop-down at the default selection `(10)`.

This number determines the placement of your application within the DOS Enabler menu. This number has no effect in the CE Avalanche environment, however it is a required parameter.

6 Enter `ceavacfg.exe` in the **Startup Command** text box.

7 Enter `Series = C` in the **Selection Criteria** text box. You can enter this information manually or use the Wizard.

Summary

The following is a summary of the Package Information:

- Package Name: `ceutil`
- Package Title: `Config Utility`
- Package Type: `Application`
- Package Revision: `1.00`
- Menu Order: `10`
- Startup Command: `ceavacfg.exe`
- Selection Criteria: `Series = C`

Add Package Files

1 Select the Package Files tab.

2 To add WM2003 version of `ceavacfg.exe` and `file1.txt`, click `Browse to Files`, navigate to and select `C:\Program`

`Files\Wavelink\Avalanche\PackageBuilder\Example Files\CE_Example_Programs\WM2003\Config_Utility` (if the default installation path was used.)

The files will appear in the Source File list.

3 To add .NET version of `ceavacfg.exe` and `file1.txt`, browse to `C:\Program Files\Wavelink]Avalanche\PackageBuilder\Example Files\CE_Example_Programs\NET\Config_Utility` (if the default installation path was used.).

The files will appear in the Source File list.

4 Click the `Flag` column for file1.txt and select `DYNAMIC` from the drop-down list.

You select the dynamic flag because the configuration file is a changeable file.

Summary

The following is a summary of Package Files:

First file in list:

- Source File: `C:\Program Files\Wavelink\Avalanche\PackageBuilder\Example Files\CE_Example_Programs\WM2003\Config_Utility\ceavacfg.exe.`
- Install Drive: APPS
- Install Path:
- File Name: `ceavacfg.exe`
- Flag:
- Selection Criteria:

Second file in list:

- Source File: `C:\Program Files\Wavelink\Avalanche\PackageBuilder\Example Files\CE_Example_Programs\WM2003\Config_Utility\file1.txt.`

- Install Drive: APPS
- Install Path:
- File Name: `file1.txt`
- Flag: DYNAMIC
- Selection Criteria:

Add Configuration Utilities

1. Select the Configuration Utilities tab.
2. Click `Add Config Entry`.
3. Click `Browse` to select a Configuration Utility.

   - To add WM2003 version of `ceutilcfg.exe`, navigate to and select `C:\Program Files\Wavelink\Avalanche\PackageBuilder\Example Files\CE_Example_Programs\WM2003\Config_Utility` (if the default installation path was used.)

   The file will appear in the Source File list.

   - To add .NET version of `ceutilcfg.exe`, navigate to and select to `C:\Program Files\Wavelink]Avalanche\PackageBuilder\Example Files\CE_Example_Programs\NET\Config_Utility` (if the default installation path was used).

   The file will appear in the Source File list.
4. Leave the **Command Arguments** text box blank.
5. Enter `Change Message` in the **Menu Title** text box.
6. Leave the working directory as `APPS/ceutil.`
7. Do not add any configuration utility support files. The ceutil.cfg file does not require any supporting files or dlls.
8. Click `OK`.

Summary

The following is a summary of Configuration Utilities

- Source File: `C:\Program Files\Wavelink\Avalanche\PackageBuilder\ExampleFiles\CE_Example|Programs\WM2003\Config_Utility\ceutilcfg.exe`.
- Application: `ceutilcfg.exe`
- Title: `Change Message`
- Working Directory: `APPS\ceutil`

Validate the Package

- From the **Tools** menu, select `Validate Package` or press `F6`.

  The Build Status should read "`ceutil is Valid`."

Make the Package

1. From the **Tools** menu, select `Make Package` or press `F7`.

   A *Save* dialog box appears

2. Select C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages as the Save in folder.
3. Enter `ceutil` as the file name.
4. Click `Save`.
5. The Build Status should read "Package build was successful."

   `ceutil.AVA` should now reside in `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` folder.
6. From the **File** menu, select `Save` to save the package configuration.

Install the Package

1. Install `ceutil.ava` in a software collection entitled **ConfigUtility** within the Avalanche Manager.
2. Enable the package and download the ceutil package to the mobile device.

3 Launch the Config Utility package from the Avalanche desktop.

4 Click `Get Config Msg` to view the default message "`CE and Avalanche are fun`."

5 Right-click the `ceutil` package and select `Configure Package`.

6 Click `Change Message`.

7 Change the message to "`Your message changed!`" and click `OK`.

8 From the **File** menu on the Enabler desktop, select `Connect`.

9 Launch the Config Utility package from the Avalanche Enabler desktop and press `Get Config Msg`.

10 Verify that the message reads `"Your message changed!"`.

Retrievable File Example

This example provides information for building a sample package containing a retrievable file. To build this sample, complete the following tasks:

- Enter Package Properties
- Enter Package Information
- Add Package Files
- Add Retrievable Files
- Validate the package
- Make the package
- Install the package in Avalanche Manager

Enter Package Properties

1 From the **File** menu, select `New` to create a new software package.

2 From the **File** menu, select `Save As`.

3 Name the package `Retrieve` and click `Save As` to save the .cfg file. This allows the developer to retrieve the configuration for this package.

4 From the Tools menu, select Properties.

5 Select Version 3 from the Avalanche Package Version pull-down.

NOTE The retrievable files feature requires a version 3 enabler.

---

6 Enter `One Electronics` in the **Vendor Name** field.

7 Enter `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` in the **Configuration File Path** field. This is the default location when the Package Builder installation path is used.

8 Enter `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` for the **Package Creation Path**.

This is the default when the default Package Builder installation path is used. This is where the package components used to make the AVA package reside. IN this case, you will see a Retrieve.PKG directory within this folder after the package has been made.

Summary

The following is a summary of the Package Builder Properties:

- Avalanche Package Version: 3
- Vendor Name: One Electronics
- Configuration File Path:
  `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages`
- Package Creation Path:
  `C:\Program Files\Wavelink\Avalanche\Package Builder\packages`

Enter Package Information

1 In the Package Information tab, enter `Retrieve` in the **Package Name** text box.

2 Enter `Retrieve File` in the **Package Title** text box.

3 From the **Package Type** drop-down list, select `Application`.

4 Enter `1.00` in the **Package Revision** text box.

5 Leave the **Menu Order** drop-down at the default selection `(10)`.

This number determines the placement of your application within the DOS Enabler menu. This number has no effect in the CE Avalanche environment, however it is a required parameter.

6 Enter `rtvdata.exe` in the **Startup Command** text box.

7 Enter `Series = C` in the **Selection Criteria** text box. You can enter this information manually or use the Wizard.

Summary

The following is a summary of the Package Information:

- Package Name: `Retrieve`
- Package Title: `Retrieve FIle`
- Package Type: `Application`
- Package Revision: `1.00`
- Menu Order: `10`
- Startup Command: `rtvdata.exe`
- Selection Criteria: `Series = C`

Add Package Files

1 Select the Package Files tab.

2 To add WM2003 version of `rtvdata.exe`, click `Browse to Files`, navigate to and select `C:\Program`

`Files\Wavelink\Avalanche\PackageBuilder\Example Files\CE_Example_Programs\WM2003\Retrievable_Files` (if the default installation path was used.)

The file will appear in the Source File list.

3 To add .NET version of `rtvdata.exe`, browse to `C:\Program Files\Wavelink]Avalanche\PackageBUilder\Example Files\CE_Example_Programs\NET\Retrievable_Files` (if the default installation path was used.).

The file will appear in the Source File list.

Summary

The following is a summary of Package Files:

- Source File: C:\Program Files\Wavelink\Avalanche\PackageBuilder\Example Files\CE_Example_Programs\WM2003\Retrievable_files\rtvdata.exe.
- Install Drive: APPS
- Install Path:
- File Name: rtvdata.exe
- Flag:
- Selection Criteria:

Add Retrievable Files

1 Select the Retrievable Files tab.

2 From the **File Save Option** drop-down list, select `Version`.

Version appends a data-time stamp to the end of the file name.

3 From the **Console Notification** drop-down list, select `Send Alert`.

Sending an alert will display an event under the Administration menu of the Avalanche console and add the event into the Avalanche Agent Activity Log each time the file is retrieved from the mobile device.

4 From the **Store Drive** drop-down list, select `C`.

The Avalanche Agent and Console should reside on this machine.

5 Enter `\retrieve\%m` in the **Store Path** text box.

The %m option creates a subfolder using the MAC address of the mobile device as the name of the subfolder. All retrievable files for a give mobile device are placed within the subfolder representing the MAC address of the mobile device.

6 Click `Add File`.

7 Click the **Drive** column and select `C` from the drop-down list.

8 Click the **File Path** column and enter `\file2.txt`.

9 Leave **Delete** option unchecked.

Summary

The following is a summary of Retrievable Files:

- File Save Option: `Version`
- Console Notification: `Send Alert`
- Store Drive: `C`
- Store Path: `\retrieve\%m`
- Drive (on mobile unit): `C`
- File Path (on mobile unit): `\file2.txt`
- Delete: `Not selected`

Validate the Package

- From the **Tools** menu, select `Validate Package` or press `F6`.

The Build Status should read "`Retrieve is Valid.`"

Make the Package

1 From the **Tools** menu, select `Make Package` or press `F7`.

A *Save* dialog box appears

2 Enter `Retrieve` as the file name.

3 Click `Save`.

4 The Build Status should read "Package build was successful."

`Retrieve.AVA` should now reside in `C:\Program Files\Wavelink\Avalanche\PackageBuilder\packages` folder.

5 From the **File** menu, select `Save` to save the package configuration.

Install the Package

1 Install the `Retrieve.AVA` file in a software collection entitled **RetrieveData** within the Avalanche Manager.

2 Enable the package and download the Retrieve package to the mobile device.

3 Launch the Retrieve File package from the Avalanche Enabler desktop.

4 Append a number two (2) to the end of **Default Retrieve Data** and click the `Write Data to Retrieve` button.

5 From the **File** menu of the Enabler desktop, select `Connect`.

You should see a file name similar to `file2.txt_050414181438` in the `C:\retrieve\<mobile device MAC address>folder`. The number is the date-time stamp. 050414 is the date in yy/mm/dd and 180138 is the time in hh:mm:ss.

6 Open the file in Notepad to confirm the contents.

Chapter 6: Post Install Script File

This chapter provides information about Post Install Script files in the Avalanche Software Package Builder. This chapter includes the following sections:

- *Script File Overview*
- *Section Headers and Commands*
- *Building an Example Script File*

Script File Overview

A Post Install Script file (script file) is an optional software package file with a .ini extension that assists the installation of the software package on mobile units. The Post Install Script File tab of Package Builder allows you to create the script files with a .ini extension. The script file contains a number of sections that specify a number of the installation parameters for software package (for example, the name and location of registry entries for the software package).

A script file must follow a certain format, or the installation of the software package will fail. The script file is processed by a program that recognizes the switches and syntax within the script file.

A script file has two primary components:

- **Section Header**. Indicates a particular feature of the software package that is being specified (for example, shortcuts or registry keys).
- **Command**. Indicates a command within a section. Each command follows a specific format.

You can open and modify an Avalanche script file in any text-editing application.

Section headers

Specific headers divide the Avalanche script file into sections that specify various parameters for the Avalanche software package. For example, the [CPYFILE] header precedes commands to copy certain files of the software package to a specific location (for example, the Flash drive).

Each section header must be enclosed in brackets ([]). The brackets are delimeters that indicate the beginning and the end of a section header.

The following is a list of section headers that are currently supported:

- **[AVALANCHE]** The section header that appears at the beginning of each Avalanche script file.
- **[STRINGS]** The section header precedes the configuration of strings that will appear throughout the Avalanche script file.
- **[REGFILE]** This section header precedes the specification of the location of a backup copy of the registry settings for the Avalanche software package. Any registry settings that are created after this entry are stored in the REG file that is specified in this section.
- **[CPYFILE]** This section header precedes the specification of a the name and location of the CPY file. Any files that are copied (see the [COPY] section) after the declaration in this section are backed up to the specified CPY file. (For Symbol devices only.)
- **[RUNFILE]** This section header precedes commands that create shortcuts (RUN files) to executable in the Avalanche software package. (For Symbol devices that support RUN files from a Flash location, as opposed to LNK files.)
- **[HHP_AUTORUN]** This section header precedes commands that are used to add autorun entries to the autorun file of Hand Held Product (HHP) mobile units. This section header must be entered using the Custom Header option.
- **[INI _FILE]** This section header precedes commands that add information to specific script files.
- **[SHORTCUT]** This header precedes commands that specify where shortcuts to components of the application are created.
- **[RENAME]** This section header precedes commands that rename certain files in the Avalanche software package during the installation process.
- **[COPY]** This section header precedes commands that copy entire directories in the Avalanche software package to a specific location.

- **[DELETE]** This section header precedes commands that will delete files or directories after the Avalanche software package is installed.
- **[ATTRIB]** This section header precedes commands that assign read/write attributes to a file.
- **[EXECUTE]** This section header precedes commands that specify reboot options that the mobile unit should use after the Avalanche software package is installed.
- **[HKEY_*]** This section header precedes commands that create registry entries. The header itself indicates the actual registry location (the * is a wildcard that represents the rest of the registry location).

Placement of Section Headers

The [AVALANCHE] header must be the first header in the document. The [AVALANCHE] section header informs the script processor that it is processing a legitimate Avalanche script file.

If strings are used in the script file, then the [STRINGS] section should follow the [AVALANCHE] section.

It is also acceptable to use a section header multiple times throughout a document. For example, you might have several [COPY] sections within a script file. However, you should be aware of how each section operates, as certain sections may affect other sections of the script file (for example, the [REGFILE] and [CPYFILE] sections). The [AVALANCHE], [CPYFILE], [EXECUTE], and [REGFILE] section headers may only be entered once.

Using Comment Delimiters

The script file uses the following comment delimiters:

- **$**. Used to indicate a string (see the description of the [STRINGS] section for more information).
- //. Used to comment out text. You can use this delimiter to add comments or notes to a script file. The script processor the point between where the delimiter is placed and the end of the line.
- **“**. Used to separate segments of a command to make them easier to read or recognize. The script processor does not process quotation (““) marks.

Section Headers and Commands

This section provides information about the commands that can be used within each section header of the script file.

[AVALANCHE]

The Avalanche section header must appear at the beginning of each Avalanche script file. It specifies that the script file is indeed an Avalanche script file. The [AVALANCHE] section of the script file may contain commands that specify whether the package is backed up on the device and whether the application that the package installs is restarted after an update.

Avalanche Command Format

The commands in the [AVALANCHE] section of the script file use the following format:

```
BackupPackage = [Yes | No]
RestartOnUpdate = [Yes | No ]
```

where:

- `BackupPackage` is a fixed constant.
- `[Yes | No]` specifies whether the package is backed up to the mobile device during the package installation. (The package will be backed up to a location that allows it to survive a cold boot.)
- `RestartOnUpdate` is a fixed constant.
- `[Yes | No]` indicates whether the application that the package installs is restarted after it is updated.

If you do not specify these commands in the script file, then Avalanche uses the default settings.

The default for `BackupPackage` is `Yes`.

The default for `RestartOnUpdate` is `No`.

[STRINGS]

The [STRINGS] sections allows you to specify text strings that can be used throughout the rest of the script file. For example, you might use the string "StartMenu" to specify the \Windows\Start Menu location on mobile units. Once this is specified in the [STRINGS] section, the rest of the document will allow you to specify the location by using the "StartMenu" string.

NOTE If the Avalanche script file uses strings, then the [STRINGS] section should immediately follow the [AVALANCHE] section in the script file. Also, any string that is repeated in the [STRINGS] section must be specified before it can be used.

String Command Format

Each string definition is placed on a new line in the Avalanche script file and uses the following format:

```
[string] = [dir]
```

where:

- `[`*`string`*`]` is the string that will be used throughout the document.
- `[`*`dir`*`]` is the directory that the string specifies.

For example:

```
Startup = \Windows\Startup
```

specifies that the "Startup" string can be used to indicate the \Windows\Startup directory.

Using Strings and the String Delimiter

Once you have specified a string in the [STRINGS] section of the Avalanche script file, you can use the string throughout the rest of the script file, even in other lines of the [STRINGS] section.

The script processor uses the dollar ($) symbol as the denotation for the beginning and ending of a specified string. Whenever you use a string value in the rest of the script file, the string value must be enclosed in this symbol.

For example:

```
$Startup$
```

would be used to indicate the \Windows\Startup directory (as specified in the previous section) throughout the Avalanche script file.

NOTE Some Wavelink script files also use quotations (“”) to further delineate strings. The processor for the script file does not recognize or process quotation marks. Unlike the dollar symbol ($), quotation marks are a value that you can use at your own discretion.

Reserved Strings

The following strings are pre-defined in the script processor and should only be used in the context that they are defined:

- **AVA**. Used to indicate the Avalanche Enabler installation directory.
- **APPS**. Used to indicate the directory where Avalanche-deployed applications are stored on the mobile unit.

NOTE The script processor also recognizes “.” as the APPS directory. For an example, see the script file in Figure 1.

- **WORK**. Used to indicate the directory where Avalanche working files are stored.
- **TEMP**. Used to indicate the directory where Avalanche temporary files are stored.
- **FLASH**. Used to indicate the directory where Avalanche files are stored on a mobile unit’s Flash drive.

[REGFILE]

The `[REGFILE]` section of the script file specifies the name and location of the REG file for the Avalanche software package. Any registry entries created after this entry in the script file are stored in the REG file that is specified in this section.

Registry keys and values are created in the `[HKEY_*]` sections of the script file.

REGFILE Format

An entry in the `[REGFILE]` section has the following format:

```
name = [path]
```

where:

- `name` is a fixed constant.
- `[path]` is the path and name of the REG file.

For example:

```
name = \Platform\TelnetCE.reg
```

creates the REG file (TelnetCE.reg) in the \Platform directory.

[CPYFILE]

The [CPYFILE] section of the script file specifies the name and location of the CPY file for the Avalanche software package. A backup copy of any files that are copied (see the [COPY] section) after this section in the script file are backed up to the specified CPY file (for Symbol devices only).

CPYFILE Format

An entry in the [CPYFILE] section has the following format:

```
name = [path]
```

where:

- `name` is a fixed constant.
- `[path]` is the path and name of the CPY file.

For example:

```
name = \Platform\TelnetCE.cpy
```

creates the CPY file (TelnetCE.cpy) in the \Platform directory.

[RUNFILE]

The [RUNFILE] section of the script file is used to create shortcuts to the applications and utilities of a software package. It is used for Symbol mobile units that support launching RUN files from a Flash location, as opposed to using LNK files.

[RUNFILE] Format

An entry in the [RUNFILE] uses the following format:

```
[shortcut] = [exe]
```

where:

- `[shortcut]` is the path (including the file name) where the RUN file is created.
- `[exe]` is the path (including the file name) to the executable.

For example:

```
$AppStartUp$\AvaMon.run = $WinProg$\Avascript t.exe
```

creates a shortcut called AvaMon.run in the $AppStartUp$ location that launches the AvaINIt.exe program in the $WinProg$ location.

[HHP_AUTORUN]

The [HHP_AUTORUN] section of a script file is used to add an autorun entry into the autorun file of HHP (Hand-Help Product) mobile units. This section header must be entered using the Custom Header option.

[HHP_AUTORUN] Format

Entries in the [HHP_AUTORUN] section of a script file have the following format:

```
[file] = [target]
```

where:

- `[file]` is the path to the existing autorun file.

- `[target]` is the EXE file to add to the autorun file, along with the support parameters for EXE. Support parameters are separated by bars (|).

For example:

```
\IPSM\Autorunscript = Program=\IPSM\Avalanche\Avascript
t.exe | Args = | Wait = 1 | StartOption = 2
```

adds the AvaInit.exe file (with specified Wait and StartOption arguments) in the IPSM\Avalanche directory to the autorun.ini file in the \IPSM directory.

[INIFILE] Format

An entry in the [INIFILE] section uses the following format:

```
[INI] = [SECTION], [ENTRY], [VALUE]
```

where:

- `[INI]` is the path to the INI file that will be modified. (If the INI file does not exist, then it will be created.)
- `[SECTION]` is the section of the script file to which the data will be added. (This can be an empty value.)
- `[ENTRY]` is any valid entry name in the script file. (This can be an empty value.)
- `[VALUE]` is a valid script value.

For example:

```
\Flash\Autorun.ini = program1,Program,monitor.exe
```

adds `Program=monitor.exe` to the `[Program1]` section of the Autorun.ini file in the \Flash directory.

[SHORTCUT]

The [SHORTCUT] section of the script file specifies the where shortcuts to components or utilities of the Avalanche software package will be created.

[SHORTCUT] Format

An entry in the [SHORTCUT] section uses the following format:

```
[file name] = [path]
```

where:

- [*file name*] is the path and name of the shortcut.
- [*path*] is the path and name of the component or utility that the shortcut launches.

For example:

```
$StartMenu$\TelnetCE.lnk =
\Avalanche\APPS\TelnetCE8140\TelnetCE.exe
```

creates a shortcut to the TelnetCE.exe application in the Start menu. (Previously, "StartMenu" was a string that specified the \Windows\Start Menu.)

[COPY]

The [COPY] section of the script file specifies the copying of files or directories from one location to another.

[COPY] Format

An entry in the [COPY] section uses the following format:

```
[source] = [path]
```

where:

- [*source*] is the source file(s).
- [*path*] is the location where the file or file(s) are copied. (The path includes the name of the file, which allows you to rename the file in the copy location.)

NOTE You can use wildcards (*) to specify a range. For example, "*.*" specifies all of the files within a directory.

For example:

```
$RAMInstallDir$\*.* = $AvaBackupDir$\*.*
```

copies all of the files (*.*) in the Avalanche installation directory (\AVA) on the RAM drive to the Avalanche backup directory (the "AvaBackupDir" string represensts the \Application\Avalanche directory).

[COPY] Optional Parameters

The [COPY] section of the script file may contain optional parameters that exclude files from a group copy or prevent certain files from being overwritten during a copy. These optional parameters must precede the copy command to which they apply.

The first optional parameter is exclude. An exclude parameter uses the following format:

```
exclude = [file]
```

where:

- `exclude` is a fixed constant.
- [*`file`*] is the file that you want to exclude from the copy command that follows.

For example:

```
exclude = AS-Enabler.dat
$RAMInstallDir$\*.* = $AvaBackupDir$\*.*
```

prevents the AS-Enabler.dat file from being copied in the group of files that are being copied from the $RAMInstallDir$ location to the $AvaBackupDir$ location.

The second optional parameter is overwrite. An overwrite parameter uses the following format:

```
overwrite = [yes | no]
```

where:

- `overwrite` is a fixed constant.

- `[yes | no]` specifies whether to overwrite a file during the preceding copy operation.

For example:

- `overwrite = no`
  `$RAMInstallDir$\Autorun.exe = $EXECDIR$\Autorun.exe`

specifies that during the copy operation from the $RAMInstallDir$ location to the $EXECDIR$ location, the Autorun.exe file (if it exists) should not be overwritten,

[DELETE]

The [DELETE} section of the script file specifies files that are deleted after the Avalanche software package is installed.

Entries in the [DELETE] section have the following format:

```
file = [path]
```

where:

- `file` is a constant
- `[path]` is the path and name of the file that is being deleted.

For example:

file = $Startup$\RestoreAva.lnk

deletes the RestoreAva.lnk file that is placed in the \Windows\Startup directory during installation. (Previously, the “Startup” string was used to specify the \Windows\Startup directory.)

[ATTRIB]

The [ATTRIB] section of a script file allows you to assign read/write attributes to a file in the package.

[ATTRIB] Format

An entry in the [ATTRIB] section uses the following format:

```
[attrib] = [path]
```

where:

- `[attrib]` is the attribute that you want to assign the file, either `read` or `write`.
- `[path]` is the path (including the file name) of the file to which you want to assign the attribute.

For example:

```
write = $CABDIR$\WLEnabler.ARM.CAB
```

assigns the write attribute to the WLEnabler.ARM.CAB file.

As another example:

```
read = $CABDIR$\WLEnabler.ARM.CAB
```

assigns the read-only attribute to the WLEnabler.ARM.CAB file.

[EXECUTE]

The [EXECUTE] section of the script file specifies reboot options that may need to be applied once the enabler or an Avalanche package has been installed.

[EXECUTE] Format

An entry in the [EXECUTE] section of the script file has the following format:

```
Reboot = [Auto | Yes]
RebootType = [Warm | Cold]
```

where:

- `Reboot` is a fixed constant
- `[Auto | Yes]` specifies whether to automatically reboot the mobile unit (`Auto`) or to prompt the user to reboot the mobile unit (`Yes`.)
- `RebootType` is a fixed constant.
- `[Warm | Cold]` specifies the type of reboot that should occur.

For example:

```
Reboot = Auto
RebootType = Warm
```

specifies that the mobile unit should automatically script initiate a warm reboot.

[HKEY_*]

The [HKEY_*] section of a script file creates registry entries. The wildcard (*) symbol is used to denote the rest of the registry key.

For example:

```
[HKEY_LOCAL_MACHINE\Software\Wavelink\Avalanche]
```

is the header of a section that contains commands that specify entries that will be created in the HKEY_LOCAL_MACHINE\Software\Wavelink\Avalanche registry key.

[HKEY_*] Format

An entry in an [HKEY_*] section of a script file uses the following format:

```
[Key] = [Value]
```

where:

- `[Key]` is the entry that will be created in the registry.
- `[Value]` is the value that will be assigned to the registry key.

For example:

```
[HKEY_LOCAL_MACHINE\Software\Wavelink\TelnetC]
RegFilePath = \Platform
```

creates the `RegFilePath` key in entry in the HKEY_LOCAL_MACHINE\Software\Wavelink\TelnetCE and assigns the value of the key as `\Platform`.

NOTE If a REG file is specified in the [REGFILE] section, then the keys created in the [HKEY_*] sections will be added to the REG file. The [REGFILE] must precede the specification of any [HKEY_*] sections that contain keys that should be added to the REG file.

Additional Information and Examples

Table 6-1 provides information and examples about each section header of the script file.

[AVALANCHE]	Required at the beginning of a script file. This section header may only appear once in a script file.

[STRINGS]	This section must precede the strings that will appear throughout the script file. The following strings are reserved in Avalanche:
	AVA
	APPS
	WORK
	TEMP
	FLASH
	The Shortcut section of this example shows how the reserved Avalanche string, APPS, is used in a script file.
	Syntax: [string]=[dir]
	Example:
	`[AVALANCHE]`
	`[STRINGS]`
	`"start"="\windows\startup"`
	`"StartMenu"="\windows\Start Menu"`
	`[SHORTCUT]`
	`"$start$\pword.lnk"="\Windows\pword.exe"`
	`$StartMenu$\pxl.lnk"="\Windows\pxl.exe`
	`"$APPS$\.pxl.lnk"="\windows\pxl.exe"`
	NOTE: The shortcut section uses the strings that are specified in the Strings section.

Table 6-1: *Script File Section Headers and Commands*

[REGFILE]	This feature applies to Symbol devices only. This section header may appear only once in a script file.
	Syntax: name=[path plus registry file name]
	Example:
	`[AVALANCHE]`
	`[REGFILE]`
	`"Name"="\Application\testreg.reg"`
	`[HKEY_LOCAL_MACHINE\Software\Wavelink\testreg]`
	`"InstallDir"="\platform\testreg"`
	`"RegFilePath"="\Application"`
	`"ConfigDir"="\Application\Avalanche\testreg"`
	NOTE: Registry settings in the HKEY section are place in the testreg.reg registry file.

[CPYFILE]	This feature applies to Symbol devices only. This section header may only appear once in a script file.
	Syntax: name=[path plus CPY file]
	Example:
	`[AVALANCHE]`
	`[CPYFILE]`
	`"Name"="\Application\testreg2.cpy"`
	`[COPY]`
	`"$APPS$\oldreg\testreg.reg"="\Platform\testreg.reg"`
	NOTE: COPY entire are placed in the testreg2.cpy file.

Table 6-1: *Script File Section Headers and Commands*

[RUNFILE]	This feature applies to Symbol devices only. Contact Symbol to determines the flash location on a given device that supports this feature.
	Syntax: [*path plus shortcut*]=[*path plus executable filename*]
	Example:
	`[AVALANCHE]`
	`[RUNFILE]`
	`"\Windows\StartUp\enabler.run"="\Application\Avalanche\Enabler.exe"`
	NOTE: This is a syntactical example but does not specify a specific Symbol flash location that supports the RUNFILE features. However, this example will confirm the operation of this feature. In this example, an enabler.run is created in the `\Windows\Startup` directory and would launch the `Enabler.exe` if `\Windows\Startup` were the specified Symbol flash location.

[INIFILE]	Syntax: [*ini*]=[*section*],[*entry*],[*value*]
	Example:
	`[AVALANCHE]`
	`[INIFILE]`
	`"\ProgramFiles\Wavelink\Avalanche\Apps\tn0390000\TELNETCE.ini "="TEST", "Launch", "Yes"`
	NOTE: This example adds a Lauch=Yes value-entry combination to the TEST section within the TELNETCEINI file.

[SHORTCUT]	Syntax: [*path plus shortcut filename*]=[*path plus executable file name*]
	Example:
	`[AVALANCHE]`
	`[SHORTCUT]`
	`"\platform\pword.lnk"="\Windows\pword.exe"`
	`"\playform\pxl.lnk"="\Windows\pxl.exe`
	`"$APPS$\pxl.lnk"="\windows\pxl.exe`

Table 6-1: *Script File Section Headers and Commands*

[RENAME]	Syntax: [*path plus filename to be renamed*]=[*path plus new filename*]
	Example:
	`[AVALANCHE]`
	`[RENAME]`
	`"\ProgramFiles\Wavelink\Avalanche\Apps\treg3\testreg3.ini"="ProgramFiles\Wavelink\Avalanche\Apps\treg3.reg3.ini`

[COPY]	This feature can be used in conjunction with a CPYFILE section on Symbol devices. This feature may also be used by itself on both Symbol and non-Symbol devices.
	Syntax: [*path to source file or files*] = [*destination directory to which a file or files will be copied*]
	`[AVALANCHE]`
	`[COPY]`
	`"\Application\testreg.reg"="\Platform\testreg.reg"`
	`"\Program Files\data\*.*"="\Application\data\*.*"`
	NOTE: Wildcards, such as *.* and *.exe are support with the [COPY] section header.

[DELETE]	Syntex: file=[*path plus filename to be deleted*]
	Example:
	`[AVALANCHE]`
	`[DELETE]`
	`"file"="\application\testreg.reg"`

[ATTRIB]	Syntax:[*attribute*]=[*path plus filename to which the attribute is assigned*] Example: [AVALANCHE] [ATTRIB] "read"="\application\attrib.txt" Viewing Attrib Section Aid Figure shows that an attempt to delete attrib.txt prompts you as to whether you really want to delete the read-only file. This tells you that the attribute was set to read-only. You can also pull the file from the device via ActiveSync and view its file attributes within Windows.

Table 6-1: *Script File Section Headers and Commands*

[EXECUTE]	This section header may appear only once in a script file.
	Syntax: `reboot= [yes | auto]`
	`RebootType= [warm | cold]`
	Example:
	`[AVALANCHE]`
	`[REGFILE]`
	`"Name"="\Application\testreg.reg"`
	`[HKEY_LOCAL_MACHINE\Software\Wavelink\testreg]`
	`"InstallDir"="\platform\testreg"`
	`"RegFilePath"="\Application"`
	`"ConfigDir"="\Application\Avalanche\testreg"`
	`[EXECUTE]`
	`"reboot"="yes"`
	`"RebootType"="cold"`

[HKEY_*]	This feature can be used in conjunction with a REGFILE section on Symbol devices. This feature may also be used by itself on both Symbol and non-Symbol devices. The example below will show it used by itself.
	Syntax: `[registry key name]`
	`[registry sub-key entry]`
	`[registry sub-key entry]`
	Example:
	`[AVALANCHE]`
	`[HKEY_LOCAL_MACHINE\Software\Wavelink\pkgbldr]`
	`"InstallDir"="\platform\pkgbldr"`
	`"RegFilePath"="\Application"`
	`"ConfigDir"="\Application\Avalanche\pkgbldr"`

[CUSTOM]	The word CUSTOM within the brackets represents an arbitrary header name.
	Syntax: `[entry] = [value]`
	Example:
	`[AVALANCHE]`
	`[LAUNCH]`
	`"App"="Yes"`
	`"Config"="Yes"`

Table 6-1: *Script File Section Headers and Commands*

Important Notes

- An Avalanche section header is required.

- Double quotes around Keys and Values are optional.
- REGFILE, RUNFILE, CPYFILE are support by Symbol mobile devices only.
- AVALANCHE, REGFILE, CPYFILE, and EXECUTE section headers should only appear once in the script file.
- AVA, APPS, WORK, TEMP, FLASH are reserved and defined. You should not be able to redefine the strings.

Building an Example Script File

The following steps build an example script file using the Package Builder.

To build a script file:

1. Launch the Package Builder.
2. Check the **Include Post Script file** option in the Package Information tab.

   This enables the Post Install Script file tab.
3. Select the Post Install Script file tab.

   The section header `[AVALANCHE]` is placed in the text box by default.
4. Click `Add Heading` and select `[STRINGS]` from the **Value** drop-down menu.
5. Click `Add Property` and enter "`start`" in the **Key** text box and "`\windows\startup.reg`" in the **Value** text box.
6. Click `OK`.

   The property is placed below the [`STRINGS`] header.
7. Click `Add Comment` and enter a comment in the text box.
8. Click `OK`.
9. Click `Add Heading` and select [`EXECUTE`] from the **Value** drop-down menu.

10 Click `Add Property` and enter "`reboot`" in the **Key** text box and "`yes`" in the **Value** text box.

11 Click the Add Registry Section button and enter "`HKEY_LOCAL_MACHINE\Software\Wavelinke\TelnetCE`" in the **Registry Key** field.

12 Enter the following Keys and Values in the text boxes.

Key	Value
"InstallDir"	"\platform\packagebuilder"
"ConfigDir"	"\Application\Avalanche\packagebuilder"

13 Click `OK`.

The registry section is placed in the text box.

14 Click `Add Heading` and select `[SHORTCUT]` from the **Value** drop-down menu.

15 Using the `Add Property` option, enter the following **Keys** and **Values** in the text boxes.:

Key	Value
"\platform\pword.lnk"	"Windows\pword.exe"
"\platform\pxl.lnk"	"\Windows\pxl.exe"

16 Click `Validate Script` to validate your file.

Your script should look like the following example (Figure 6-1).

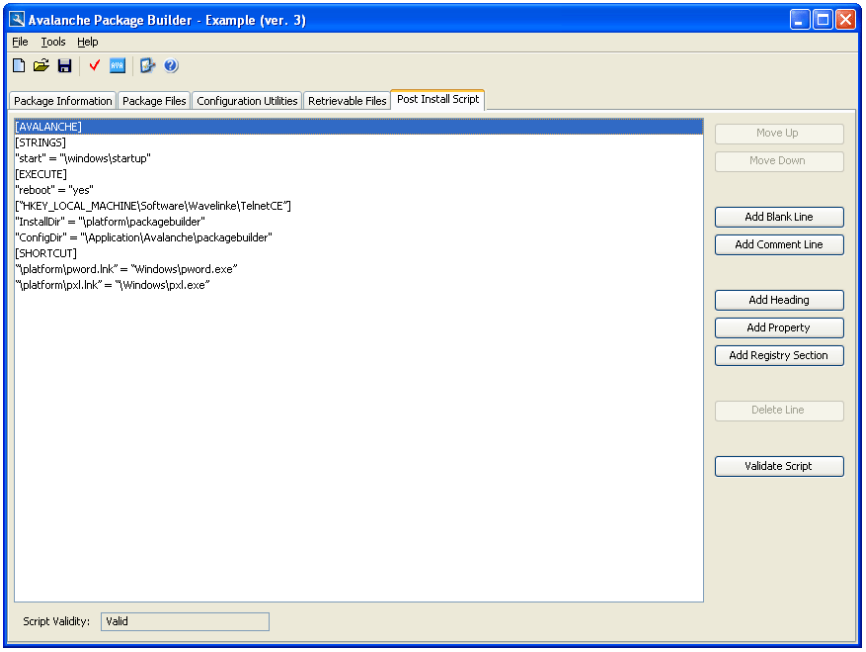

Figure 6-1. *Building the Script File*

```
[AVALANCHE]

[STRINGS]
“Start”=”\Windows\startup”

[EXECUTE}
“reboot”=”yes”

HKEY_LOCAL_MACHINE\Software\Wavelinke\TelnetCE
“InstallDir”=”\platform\packagebuilder”
“ConfigDir”=”\Application\Avalanche\packagebuilder”

[SHORTCUT]
“\platform\pword.lnk”=”\Windows\pword.exe”
“\platform\pxl.lnk”=”\Windows\pxl.exe”
```

Chapter 7: Selection Criteria Builder

This chapter provides detailed information about creating selection criteria using the Selection Criteria Builder in Package Builder and includes the following sections:

- *Selection Criteria Builder Overview*
- *Creating Selection Criteria*

Selection Criteria Builder Overview

A set of rules called selection criteria, define which mobile devices will receive designated updates. Using Package Builder, you can apply this selection criteria to an entire package (using the Selection Criteria Builder in the Package Information tab) or to individual files (using the Selection Criteria Builder in the Package Files tab).

When creating selection criteria, ensure that general criteria is applied to the entire software package and more specific criteria is applied to individual files. For example, you could apply selection criteria to an entire software package that restricts the package distribution to a certain operating system. You could also apply selection criteria to individual package files that restricts file distribution to specific devices.

A selection criteria string is a single expression (much like a mathematical expression) that takes a set of variables corresponding to different aspects of a mobile device and compares them to fixed values. The syntax includes parentheses and boolean operators to allow flexible combination of multiple variables.

By default, the selection criteria string is empty, which allows all packages and files to download to all mobile devices. You can modify this criteria at any time.

Creating Selection Criteria

You can use the Selection Criteria Builder Wizard to build a valid selection criteria string. You can also use the Selection Criteria Builder Wizard to validate the selection criteria string.

You can build the selection criteria string by selecting or typing string elements one element at a time. The string elements include:

- Selection variables such as `ModelName` or `KeyboardName`. These variables determine the type of restriction placed on the package or profile. For example, by using a `ModelName` variable, you can restrict the package or profile to a specific class of mobile devices, based on their model numbers. You may use any property that you have assigned a device as a selection criteria variable.

- Operators such as EQ (=), AND (&), and OR (|) that are used to assign a value to a selection variable or to combine multiple variables.

NOTE Parentheses are recommended when multiple operators are involved. Nesting of parentheses is also allowed.

- Actual values that are assigned to a selection variable. For example, if you assign a value of 6840 to a `ModelName` variable by building the string, `ModelName = 6840`, then you will restrict packages or profiles to model 6840 mobile devices.

Building a Selection Criteria String

This section provides instructions about creating selection criteria in the Selection Criteria Builder.

To build a selection criteria string

1 Click the Wizard button to open the Selection Criteria Builder.

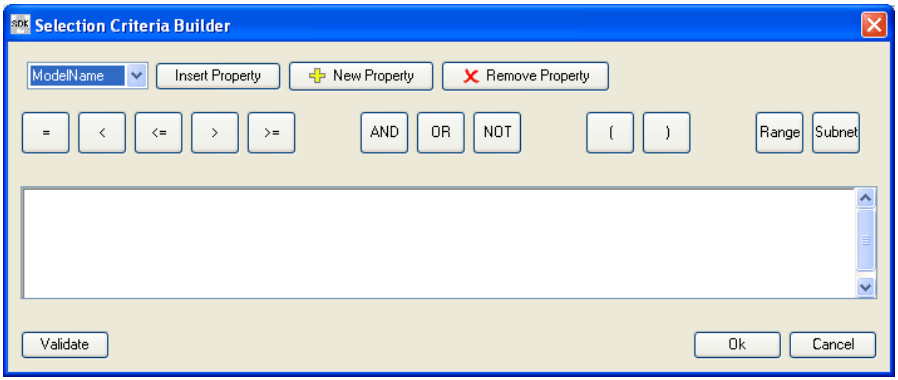

Figure 7-1. *Selection Criteria Builder*

2 Select a source property from the drop-down list.

3 Click `Insert Property` to add the property to the text box.

NOTE For information about source properties, see *Selection Variables* on page 102.

---

4 Select one of the operator buttons.

When you select an operator, it will be added to the text box and will be placed next to the last source property entered.

NOTE For information about operators, see *Operators* on page 106.

---

5 In the text box, type a value for the source property that you selected.

6 For each additional element you want to add to the selection criteria string, repeat the preceding steps.

NOTE Due to the potential complexity of long selection criteria strings, it is recommended that you limit the selection criteria to 20 selection variables or less.

---

7 Click `Validate`.

The Selection Criteria Builder will indicate whether the selection criteria expression contains any errors.

If there are no errors, click `OK` to return the previous tab.

Optional Criteria Build Methods

- **New Property**. You can create new properties to add to the drop-down list. To create a new property click the `New Property button` and enter the name of the new property.

- **Remove Property**. You can remove properties that you created, but you can not remove the default properties in the list. To remove properties, select the property from the drop-down list and click the `Remove Property` button.

Selection Variables

The selection criteria is based on the use of selection variables. You can place numbers and strings directly in the selection criteria string, with or without quotes.

NOTE Selection variables and values are case sensitive.

For example, the following selection criteria strings are valid:

```
ModelName=6840
ModelName = 6840
ModelName="6840"
```

The following Palm emulation selection criteria strings is valid:

```
Series = S
```

while the following is not:

```
series = s
Series = s
series = S
```

Long strings are also supported as selection criteria. For example, the following string is valid:

```
Series = 3 | (MAC = 00-A0-F8-27-B5-7F | MAC = 00-A0-F8-80-3D-
4B | MAC = 00-A0-F8-76-B3-D8 | MAC = 00-A0-F8-38-11-83 | MAC
= 00-A0-F8-10-24-FF | MAC = 00-A0-F8-10-10-10)
```

Selection variables for use in the selection criteria string as follows:

Variable	Description
`ModelName`	The standard model name for a mobile device. This name is often a number but it can be alphanumeric as well. Examples include 6840, 3940, 4040. If the model number is unknown, it might appear in one of the views when the mobile device is selected A few of the supported values include: `1040, 1740, 1746, 1840, 1846, 2740, 2840, 3140, 3143, 3540, 3840, 3843, 3940, 4040, 5040, 6140, 6143, 6840, 6843, 6940, 7240, 7540, 7940, 8140, 8940, PTC960, TR1200, VT2400, WinPC, WT2200, 7000CE,` HHP7400, MX1, MX2, MX3, VX1, iPAQ, iPAD, Falcon, ITCCK30, ITC700 Example: `ModelName = 6840`
`ModelCode`	A number set by the device manufacturer and used internally by the BIOS to identify the hardware. Supported values include: `1`= LRT 38xx/LDT `2` = VRC39xx/69xx `3` = PDT 31xx/35xx `4` = WSS1000 `5` = PDT 6800 `6` = PDT 6100 Example: `ModelCode <= 2` This matches all 38xx, 39xx, and 69xx devices.

Table 7-1: *Selection Variables*

Variable	Description
`Series`	The general series of a device. This is a single character: '3' for Symbol '3000' series mobile devices, '7' for Symbol '7000' series mobile devices, etc. Supported values include: `3` = DOS 3000 Series `P` = DOS 4000 and 5000 Series `7` = DOS 7000 Series `T` = Telxon devices `C` = CE devices `S` = Palm devices `W` = Windows machines D = PSC and LXE DOS devices Example: `Series = 3`

`KeyboardName`	A string depicting which style of keyboard the mobile device is using (46key, 35key etc.). This selection variable is not valid for CE devices.
	
	Supported values include:
	
	`35KEY`
	`46KEY`
	`101KEY`
	`TnKeys`
	
	Example:
	
	`KeyboardName = 35KEY`

`KeyboardCode`	A number set by the device manufacturer and used internally by the BIOS to identify the keyboard type. Supported values include: `0` = 35-Key `1` = More than 35 keys and WSS1000 `2` = Other devices with less than 35 keys Example: `KeyboardCode = 0`
`Rows`	The number of display rows the mobile device supports. The possible value range is 1 to 25. Example: `!(KeyboardName=35Key)&(Rows=20)` This example matches all mobile devices with 20 rows, except those with 35-key keyboards.

Table 7-1: *Selection Variables*

Variable	Description
`Columns`	The number of display columns the mobile device supports. The possible value range is 1 to 80. Example: `Columns > 20`
`IP`	IP address of the mobile device. Enter all IP addresses using dotted notation. IP addresses can be compared in three ways: Direct comparison with a single IP address. For example, IP = 10.1.1.1. Comparison with an arbitrary address range. For example, IP = 10.1.1.5 – 10.1.1.15 (This can also be written as IP = 10.1.1.5 – 15.) Comparison with a subnet number. This is done by supplying the network number along with the netmask or CIDR value. For example, IP = 10.1.1.0/255.255.255.0. Using CIDR notation, this can also be written as IP = 10.1.1.0/24.
`MAC`	MAC address of the mobile device. Enter any MAC Addresses as a string of hexadecimal digits. Dashes or colons between octets are optional. For example: `MAC = 00:A0:F8:85:E8:E3`

Table 7-1: *Selection Variables*

Variable	Description
`Group`	Mobile device groups can be created and designed within Avalanche. These groups can then be used as selection criteria. Example: Group = CE would represent all devices that are listed within the Mobile Device Group named “CE.”
`LastContact`	The LastContact proper allows specifying absolute time stamps and relative time stamps. This forces constant re-evaluation as the time base changes. The following is a list of absolute time stamp formats. Because of their syntax, they must always be quoted. mm/dd/yyyy LastContact =“12/20/2009” (All day long) HH:MM mm/dd/yyyy LastContact=“23:15 12/20/2009” (All minute long) hh:mm AP mm/dd/yyyy LastContact=“11:15 PM 12/20/2009 (All minute long) Plus range- forms of the above. The following is a list of relative time stamps. Relative format uses an offset from the current time. <offset>M LastContact=60M (60 minutes in the past) <offset>H LastContact=1H (One hour in the past) <offset>D LastContact: 1D (One day in the past) Range forms of the above. Special syntax allows inverted ranges for the range form to reduce the amount of confusion. Example: Last Contact 7D-1M (The past seven days to the present)

Table 7-1: *Selection Variables*

Operators

All selection criteria strings are evaluated from left to right, without operator precedence. When more than one operator is involved, you must include parentheses in order for the selection criteria string to be evaluated properly.

For example:

```
(ModelName=3840) or ((ModelName=6840) and (KeyboardName=
46Key))
```

NOTE Spaces around operators are optional.

The preceding selection criteria string states that either 3840 mobile devices regardless of keyboard type or 46Key 6840 mobile devices will receive the software package.

You may use the symbol of the operator (!, &,. |, etc.) in a selection criteria, or you may use the letter abbreviation (NOT, AND, OR, etc.). If you use the letter abbreviation for the operator, then you must format the letter abbreviation in all upper-case letters.

The following operators (Table 7-2) can be used along with any number of parentheses to combine multiple variables.

Symbol	Description
`NOT (!)`	Unary operator that negates the boolean value that follows it. In the following example, all mobile devices with 20 rows receive the software packages within the collection except for those with 35Key keyboards. `! (KeyboardName = 35Key) & (Rows = 20)`
`AND (&)`	Binary operator that results in TRUE if and only if the expressions before and after it are also both TRUE. Example: `(ModelName=3840) | ((ModelName=6840) & (KeyboardName= 46Key))`
`OR (|)`	Binary operator that results in TRUE if either of the expressions before and after it are also TRUE. In this example, either 6840 or 3840 mobile devices can receive the software packages. `(ModelName=6840) | (ModelName=3840)`
`RANGE (or -)`	Binary operator allows for a range of values such as IP addresses. To avoid possible syntax issues, always surround the range operator with spaces. If you have a value that contains the range operator, surround it with double quotes. Example: `IP =10.20.10.15 - 200`

Table 7-2: *Operators*

Symbol	Description
`SUBNET (or \)`	Binary operator allows for a comparison with subnets or CIDR notation. Example: IP = 10.20.10.0/255.255.255.0 or IP = 10.20.10.0/24
`(=)`	Binary operator that results in TRUE if the two expressions on either side of it are equivalent. Example: `ModelName = 6840`
`>`	Binary operator that results in TRUE if the expression on the left is greater than the expression on the right. Example: `Rows > 20`
`<`	Binary operator that results in TRUE if the expression on the left is less than the expression on the right. Example: `Rows < 21`
`>=`	Binary operator that results in TRUE if the expression on the left is greater than or equal to the expression on the right. Example: `Rows >= 21`
`<=`	Binary operator that result in TRUE if the expression on the left is less than or equal to the expression on the right. Example: `Rows <= 20`

Table 7-2: *Operators*

Operators use the following precedence:

1 Parenthesis

2 OR operator

3 AND operator

4 NOT operator

5 All other operators

Operator spelling can either be all upper-case letters or all lower-case letters. You can not mix upper case and lower case.

Sample Strings

You can place numbers and strings directly in the selection criteria string, with or without quotes. Selection variable names and values are case sensitive.

Enter all IP addresses using dotted notation. IP addresses can be compared in three ways:

- Direct comparison with a single IP address. For example, `IP = 10.1.1.1`.
- Comparison with an arbitrary address range. For example, `IP = 10.1.1.5 - 10.1.1.15` (This can also be written `IP = 10.1.1.5 - 15`.)
- Comparison with a subnet number. This is accomplished by supplying the network number along with the netmask or CIDR (Classless Inter-Domain Routing) value. For example, `IP = 10.1.1.0/255.255.255.0`. Using CIDR notation, this can also be written as `IP = 10.1.1.0/24`.

Additional Examples:

```
(ModelName =6840) or (ModelName = 3840)
```

In the following example, all mobile devices with 20 rows receive the software packages except for those with 35Key keyboards.

```
Not (KeyboardName = 35Key) and (Rows = 20)
```

Appendix A: Assigning Storage Location for Retrievable Files

Retrievable files are device files that you want to retrieve from the mobile device and upload to a specific directory on the Mobile Device Server. For example purposes, this section will refer to the "HOME" directory. These files could be log files indicating activity of a program or any other type of data you want to retrieve.

The handling of Retrievable files is configured in the Avalanche Package Builder where you define which files should be retrieved and where to store the files. Currently, the **Store Drive** selection list in Package Builder does not include an option to select the "HOME" drive (or other custom drives), which is the default storage location in Avalanche. Therefore, to assign Retrievable files to upload to the "HOME" directory, you must manually edit your Avalanche package.

The following steps are required:

- Defining the "Home" Path in Avalanche
- Building your Package
- Editing Your Software Package
- Deploy the package to the Mobile Device Server.

Defining the "Home" Path in Avalanche

The first step in the process is to define the path where you want your Retrievable files stored. The Mobile Device Server automatically creates a directory for Retrievable files. However, by defining your own path for the files, you will not have to navigate through multiple folders and directories on the Mobile Device Server.

To define the HOME Path:

1. Launch your Avalanche Console and select the Mobile Device Server Profile you want to configure.
2. Click **Edit**.

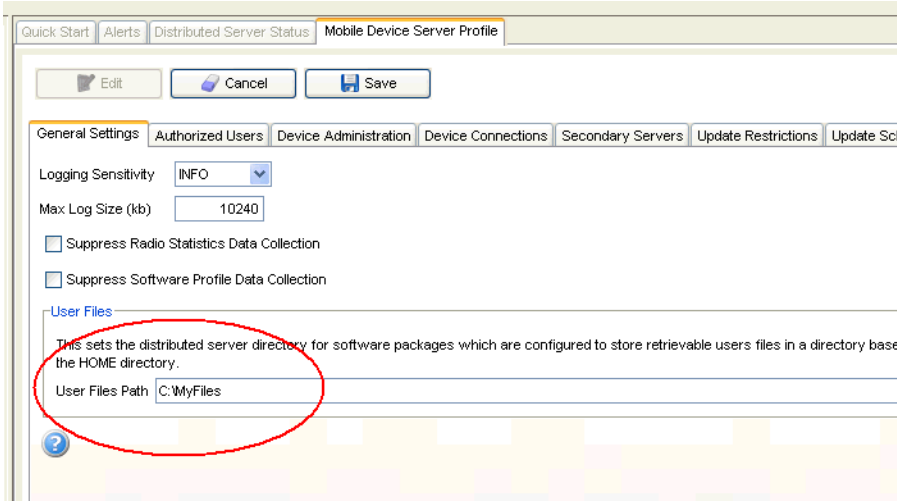

Figure 0-1. *Defining the Home Directory*

3 In the **General Settings** tab, under **User Files**, enter the path where you want Retrievable file to be stored. (Example C:\MyFiles.)

4 Click **Save**.

Building your Package

The next step is to build your Avalanche package using Package Builder. If you are creating a regular package with files you want sent to the device, create the package as you usually would. If you are creating a package to send down only the Retrievable Files configuration, you must create the package as a **Support** package and add information in the **Post Install Script** tab for the package to be valid.

To build your package:

1 Launch Avalanche Package Builder.

2 Enter the required information to build a software package in the **Package Information** tab:

- Name
- Title

- Type
- Revision

3 Select the **Retrievable Files** tab.

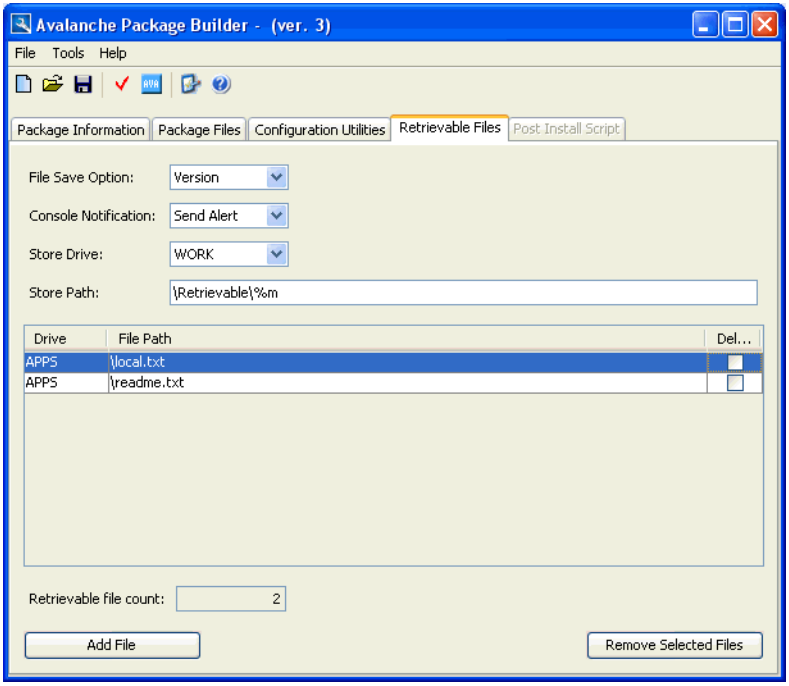

Figure 0-2. *Retrievable Files*

4 Set the **Store Drive** to **WORK.** (Technically you can set it to any option, but setting to WORK makes it easier to edit later.)

NOTE The example of the Store Path used (**\Retrievable\%m**) uses a variable that indicates the files will be stored in a subdirectory based on the MAC address of the device. For more information about Package Builder variables, refer to *Table 4-9: Retrievable Files* on page 46 (*Store Path*).

---

5 In the **File Path** text box, enter the file names you want retrieved from the mobile devices. You can also use **Add File** to add files to the list.

6 Finish building your package by adding additional package files in the **Package Files** tab or enter information into the **Post Install Script** tab.

   If you are not adding any package files, you must do one of the following to prevent the package from failing:

   - Create the package as a **Support** package and add information in the **Post Install Script** tab. You can add any script information you want, but there must be some information included.
   - Enter the Retrievable files information in the **Packages Files** tab.

7 Validate your package by selecting **Tools >Validat**e. If your package is configured correctly, you will receive a message telling you the package is valid.

8 When you are ready to build your package, select **Tools > Make Package**. Your package will generate and you can edit the package to change the **Store Drive**.

Editing Your Software Package

You must edit your Avalanche package to set the Retrievable files to store in your "HOME" directory.

To edit the software package:

1 Navigate to the location of the package you built and change the extension of the package name to `.zip` extension.

2 Extract the `.zip` file into a blank folder.

3 Navigate through the package until you find the `.PPF` file. This is the file that contains the information about your package.

4 Open the `.PPF` file in a text editor (such as Notepad).

5 Locate the **user.file.agent.dir** line and change **WORK** to "**HOME"**. (If you specified the **Store Drive** as something other than WORK, change that name to "HOME". The drive name must be capitalized.)

6 Save the file and navigate out of the directory until you are at the **TempInstall.PRF** folder.

7 Right-click on the folder and zip it using a filename with an `.ava` extension.

8 Install the new `.ava` package in Avalanche and then deploy the changes. The Retrievable files for the package will be configured to store in the specified directory.

NOTE For the example used in this document, the file destination of the Retrievable files will be `C:\MyFiles\Retrievable\00a0f8123456\local.txt` (or readme.txt)

Index

R

S

T

U

V