

PalmSource® Installer Developer Guide

Introduction.....	1
Terminology.....	2
Palm OS Package Builder Architecture.....	3
Creating a PSI Package.....	5
A Complex Installation Scenario.....	7
More on the OTA Scenario.....	7
Creating a PSML Definition File.....	7
PSI File Format Details.....	12
Sample HTML and JavaScript for Your Application Download Page.....	12
More on PalmCall.....	13

Introduction

PalmSource Installer is a Palm OS® software solution that simplifies the downloading and installing of add-on applications over the air (OTA) and via Palm OS Desktop HotSync® technology. Users can install applications with a single click. This Developer Guide is for third-party developers and distributors of software, via web sites.

The PalmSource Installer software solution will accelerate the adoption of third-party applications, thereby leveraging the power and flexibility of the Palm OS platform. Users no longer need to know their screen resolution, OS version, or how to uncompress a file.

PalmSource Installer includes three components:

- **Palm OS Package Builder** is a developer tool for creating packages.
 - It reads in PSML files (a type of XML file) and resource files referenced by the PSML definition file and creates .PSI files, or PalmSource Install packages.
 - It can automatically create packages for OTA delivery of specific screen sizes or languages, as well as create a full package for download to the user's desktop. The full package includes all languages and all screen sizes, plus any desktop components or other files too big for OTA delivery.
- **PalmSource Package Installer Desktop** is a Full Package file extractor/installer for desktop deliveries and System Info reporting application to the PalmSource Installer's desktop conduit.
 - It takes the full packages and determines which components are appropriate for the target device.
 - If a desktop application (EXE) needs to be executed to install any desktop components, the EXE file is run to install the device-specific software.
- **PalmSource Package Installer Client** is an OTA package extractor/installer for smaller file deliveries via OTA.
 - When PSI packages are downloaded directly to the device, PalmSource Package Installer Client unpacks OTA packages and installs the files appropriately. It can also include a simple notification to pass to the desktop at the next HotSync operation. The desktop will then prompt the user to download the full package. This is useful if the application has desktop components to install.

- PalmSource Package Installer Client reports to the conduit the system capabilities of the device so the desktop Package Installer can determine the proper files to install.

Here's a general guide to getting started using PalmSource Installer. There are just a few steps:

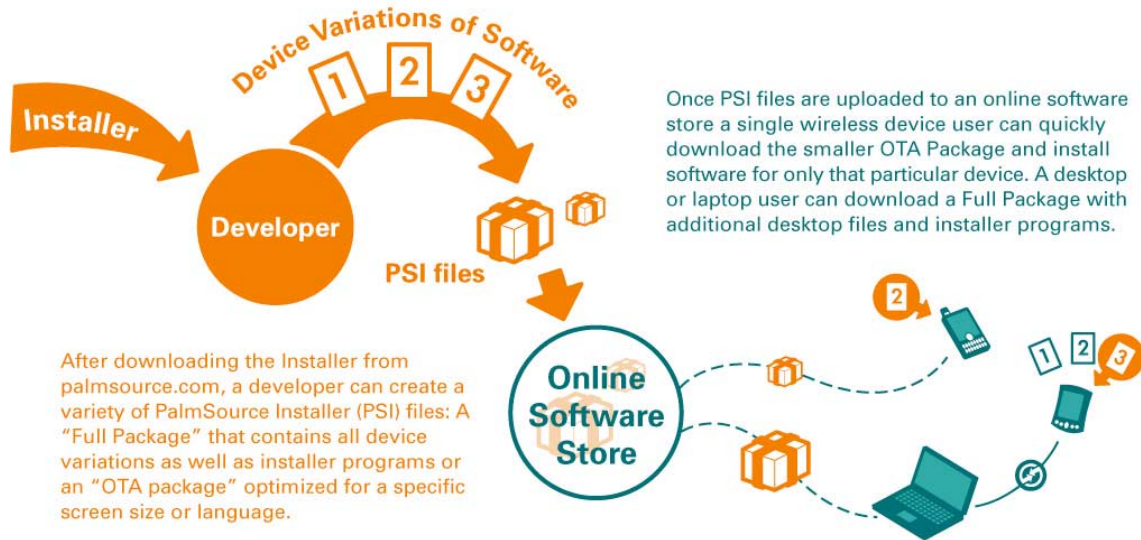
- 1) Download and install Palm OS Package Builder.
- 2) Create a PSML definition file by modifying the provided samples, or use the **-s** option from the command line to create a starter file. See "Creating a PSML Definition File" on page 7.
- 3) Place all your files (PRCs, PDBs, readme files) into the appropriate directory. All file references are assumed to be relative to the PSML file unless they contain a full path.
- 4) Run the Package Builder application to create a PSI file. See "Creating a PSI Package" on page 5 and "PSI File Format Details" on page 12.
- 5) Place the resulting PSI file on your distribution web site. Be sure to include the sample HTML file on your distribution web site for downloading the PSI file. The proper HTML file will enable users to download the PalmSource Package Installer Desktop or Client if the user does not already have it. For examples, see "Sample HTML and JavaScript for Your Application Download Page" on page 12.

Terminology

OTA Package – A small, specialized package created for a particular screen size or language. OTA packages optimize the over-the-air (OTA) download experience by including only the files, overlays or language files necessary for a particular device's capabilities.

Full Package – A complete package that includes all files, overlays, language files, etc. The Full Package may also include desktop files such as installer programs for synchronization conduits or desktop applications. The full package is normally downloaded to the desktop and not over the air (OTA).

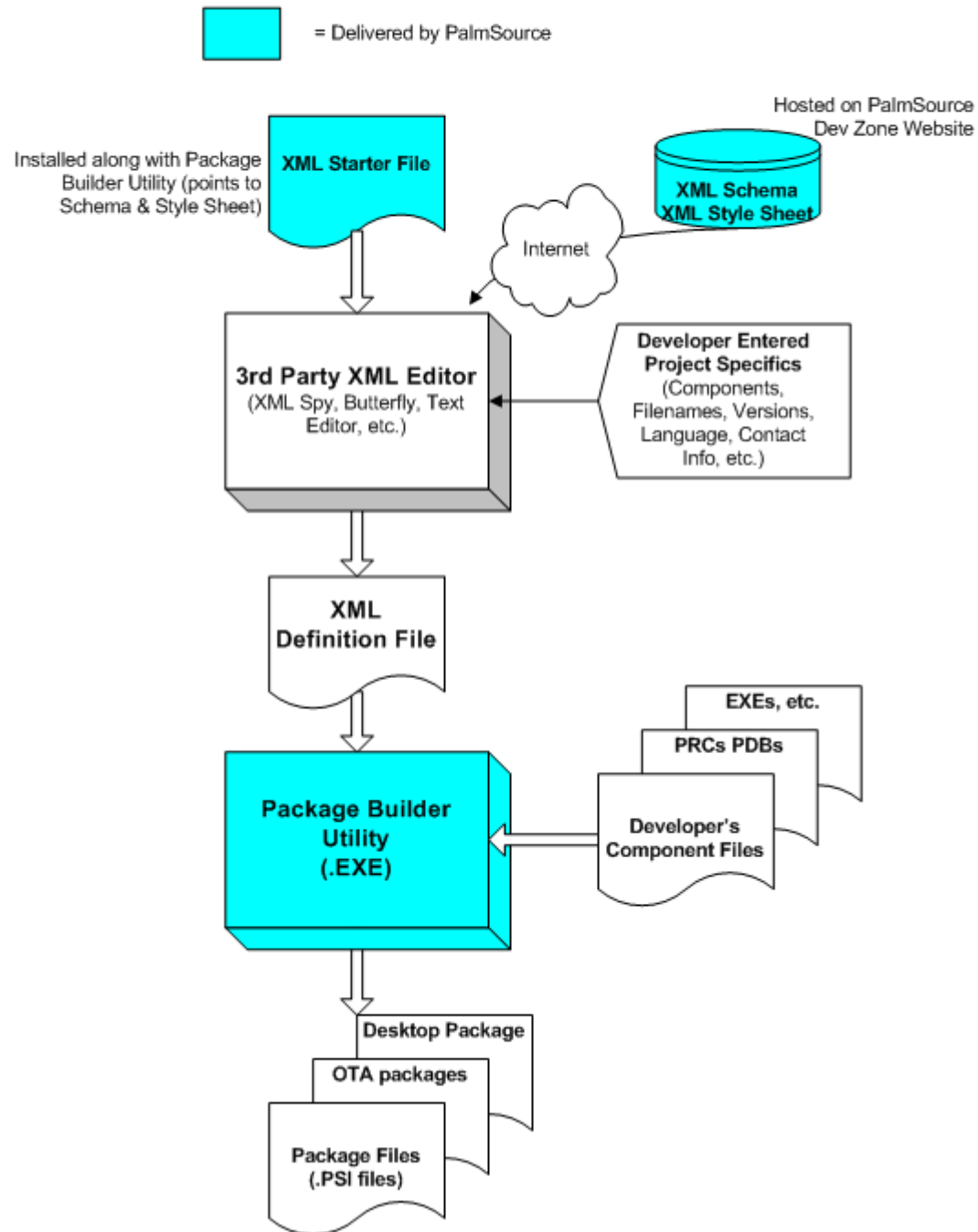
How the PalmSource Installer Works



Palm OS Package Builder Architecture

PalmSource Installer provides a single package file that can be installed to any desktop and device with the barest minimum of queries to the end user. Here is a brief summary of the components involved:

Palm OS Package Builder Architecture



Creating a PSI Package

To define and create a package for the first time, follow these instructions:

- 1) Take the Package Builder application and PSML Samples that you downloaded to your hard drive and put them in a location you can get to easily via a command line. If you do not already have the application, go to <http://www.palmsource.com/developers/>.
- 2) Install an XML editor such as Altova XMLSpy. The Home Edition of the XMLSpy tool is free and can be obtained at <http://www.altova.com/>.
- 3) Look at some of the PSML samples to get an idea how you might define a package definition. The WordSmith directory contains the actual sample applications. The structure of a PSML definition file is described in detail under “Creating a PSML Definition File” on page 7.
- 4) Using a typical application like WordSmith as an example, create a WordSmith set of packages by executing from a Windows command line:

```
> PackageBuilder -d WordSmith.psml
```

The **-d** option is for debug, to display more information about what is happening. Alternatively, you can right-click on WordSmith.psml and choose “Build PalmSource Install Package(s)” using Windows Explorer. In order to enable this shell integration, you must run PackageBuilder at least one time.

NOTE: Developers can turn on “Verbose” log mode in the PackageInstaller conduit configuration. This gives much more information about the install and about the device that is syncing.

- 5) Notice how Package Builder produces different packages depending on screen size and language, as entered in the PSML definition file. Also, because this particular sample has variants depending on whether the packages are destined for Handango or PalmGear, packages are created specifically with those variants as well.
- 6) To install a package, set up the desktop by running the setup application for the desktop conduit. Simply double-click the PSISetup.EXE file. To uninstall the conduit, use the Add/Remove Programs control panel.

This installer installs the conduit (PackInst.dll) in the HotSync Manager directory, and adjusts the Registry to register the conduit to HotSync Manager. You should now see the conduit under the Custom option of HotSync Manager.

Note that the Package Installer Desktop is an install conduit, and registers with the Install Tool. If you run the Install Tool from Palm Desktop and click **Add**, you will see a new file type supported by the Install Tool: PalmSource Install (.psi).

Each time you synchronize, the conduit checks to see if you have the Package Installer for Palm OS client on your device. If it is not there or is an older version, the conduit automatically generates and installs a copy of PackageInstaller.prc. If you are unable to synchronize, you can grab the PackageInstaller.prc file from the original zip file and install the file manually to your device.

Also note that at HotSync time, the Package Installer runs on the device in order to report the capabilities of the device (such as screen size and language). These capabilities are

captured and stored on the desktop for later use. When the conduit installs a full package, the capabilities and system information gathered by the PRC and conduit are used to select the appropriate files for installation to the device.

- 7) To see how packages are installed from your desktop, you can either install a full PSI package by installing one of the WordSmith packages you built earlier, or use your desktop web browser and download the file from:
<http://www.palmsource.com/installer/demo/WordSmith.psi>

This sample file should be equivalent to the WordSmith.psi file created by Package Builder in the earlier steps.

Note that as part of PalmSource Installer 1.0, an ActiveX control simplifies the desktop downloading experience to a single click. Additionally, if the desktop component is not installed, the ActiveX control will retrieve and install the Package Installer Desktop software, similar to Flash or Shockwave Player. This design reduces the steps to download a PSI file and get the proper software ready for install at the next HotSync operation.

- 8) Now that you have the PSI file staged to be installed, press the HotSync button. This will initiate the Package Installer conduit to extract the PSI file and install the proper files to your device. You might want to try different devices—Low Res, Hi Res, German and Spanish devices—to be sure the proper files are being extracted.
- 9) Check the HotSync Log to see what happened during the HotSync operation.

Here's a typical example:

```
HotSync operation started 7/25/2004 5:02:14 PM
Install - checking device capabilities
Extracting files from WORDSM~1.PSI
OK Address Book
OK To Do List
OK Memo Pad
OK Expense
PocketMirror®      3.1.6.0
OK Outlook Calendar
  - Installed file: C:\Program
Files\Palm\Treo60R\Install\SpellDataSmall.pdb
  - Installed file: C:\Program
Files\Palm\Treo60R\Install\SpellSmith.prc
  - Installed file: C:\Program Files\Palm\Treo60R\Install\Thesdata.pdb
  - Installed file: C:\Program Files\Palm\Treo60R\Install\WordSmith.prc
OK Install
  -- Backing up db Saved Preferences to file C:\Program
Files\Palm\Treo60R\Backup\Saved_Preferences.PRC
OK System
HotSync operation complete 7/25/2004 5:03:05 PM
```

You can see the files Package Builder extracted early in the HotSync operation and then installed to the device at the end, just before backing up files.

- 10) Inspect the device. You should have WordSmith on the device. Also, the Package Installer hidden application should be installed on the device as well. Look under **Info** for details.
- 11) You can now download OTA packages via the web browser. Go to:
<http://www.palmsource.com/installer/demo/ota/>
- 12) It is recommended that you tell users where the application will appear once it's installed.

A Complex Installation Scenario

Developer X creates a multiplayer strategy game called Aggression. Users can play against the computer or against other players. When playing against other players, the user's moves are communicated either OTA or via a conduit on your next HotSync operation.

Aggression supports English, French, Spanish, and German. It includes a conduit for Windows. It has separate graphics overlays for 160x160, 320x320 and 480x320 resolution.

Here's how to use PalmSource Installer to make it easy for users to install the Aggression application:

The developer must create an installer for the conduit.

The developer must create a PSML file describing the various capabilities and the location of the required files. Package Builder provides an XML Schema and stylesheet for the XMLSpy tool, as well as links to some excellent free tools to facilitate the quick and correct generation of the PSML files. (Examples of PSML file creation appear a bit later in this document.)

Package Builder will use this to create a single PSI file containing all of the elements, including the conduit installer. It will optionally create OTA files. In this example, it will create 12 OTA files since the four languages can be combined in any combination with the three screen resolutions. If OTA packages are built, Package Builder will also generate an HTML file containing the HTML link that needs to be pasted into a web page so that the Package Installer Client can select the correct OTA file.

The developer puts all of the files on a web site. Ideally that site will display the full PSI file to a desktop browser, and either the OTA files or the PalmCall link to a handheld web browser. PalmSource provides sample web code to developers to show how to do this.

Now a customer on a Windows computer decides to try Aggression. If your web site includes the proper HTML file to implement the ActiveX control, the customer simply clicks on the link, Aggression is downloaded, and installation begins. Without the ActiveX control, the browser asks the customer whether they would like to install PalmSource Installer.

Once Package Installer Desktop opens the PSI file, the user is asked to select a device username. The capabilities of the device will be detected when they sync to install. In this example, the conduit will detect that the user has a French Sony UX-50, so the desktop conduit merges the 320x480 graphics overlay and the French language overlay into the standard English game, and then installs the required files to the device.

More on the OTA Scenario

Now suppose that rather than installing from the desktop, the user discovers Aggression while browsing with a Treo 600. The user sees a link to click on to install. The link includes the proper PalmCall: string. When the user clicks the link, the Package Installer Client is sent the list of available versions via the PalmCall string. Since the user's device can only support 160x160 and the locale is set to French, Package Installer tells the browser to download the 160x160 French PSI file via a specific URL. Once the browser completes the download, Package Installer is passed the file via Exchange Manager and expands the game into main RAM.

Inside the package is a Desktop Component Request, which the conduit recognizes. Next time the user syncs, the following message appears: "Aggression has desktop components available that are not yet installed. Would you like me to launch your browser and install them?" If the user says yes, the Full Package of Aggression will be downloaded and installed, including any desktop software.

Creating a PSML Definition File

Package Builder is implemented as a Win32 console application so that it can be used in MAKE files.

The Package Builder reads a PSML definition file and creates one or more PSI files. Each PSI file consists of an install script and a series of files compressed using zlib compression. This is the same compression scheme that is used for ZIP files.

The PSML file contains a product description and a list of components. Components can be nested in the PSML file. Here is a simple example:

```
<?xml version="1.0" encoding="UTF-8"?>
<PalmOSInstaller xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="psi.xsd">
  <Publisher Name="Tax Sense Software"/>
  <Product DisplayVersion="1.3" ID="TaxCalc" Version="1.3.0.0">
    <Component>
      <File Filename="TaxCalc.prc"/>
    </Component>
  </Product>
</PalmOSInstaller>
```

As you can see, this product is “TaxCalc 1.3”, and it has a single PRC file to install. For a script as simple as this, the Package Builder will produce a single PSI file that can be used on the desktop or on the handheld.

Here is a more complex example. It describes a game with multiple graphics resolutions and multiple languages.

```
<?xml version="1.0" encoding="UTF-8"?>
<PalmOSInstaller xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="psi.xsd">
  <Notes>This is an example of an installer for a video game with multiple screens.</Notes>
  <Publisher Name="Super Games">
    <ContactInfo Type="Support">
      <URL>http://supergames.com/support</URL>
      <Email>support@supergames.com</Email>
    </ContactInfo>
    <ContactInfo Type="Sales">
      <Address1>123 Main Street</Address1>
      <City>Escondido</City>
      <State>CA</State>
      <Country>US United States</Country>
      <Zip>92027</Zip>
      <Email>sales@supergames.com</Email>
    </ContactInfo>
  </Publisher>
  <Product PrimaryLanguage="English" ID="Aggression" Version="1.0.9.2" DisplayVersion="1.09b">
    <Output Variant="PalmSource" Filename="AggressionPS"/>
    <Output Filename="AggressionPG" Variant="PalmGear"/>
    <Prefer Orientation="Landscape"/>
    <Component Description="Aggression for Palm OS">
      <File Filename="aggression.prc" InstallToCard="OK">
        <Select Type="Resolution">
          <When Resolution="160x160">
            <File Filename="con160x160.pdb">
              <Select Type="Language">
                <When Language="Spanish">
                  <File Filename="con160x160sp.pdb"/>
                </When>
                <When Language="Swahili">
                  <File Filename="con160x160sw.pdb"/>
                </When>
              </Select>
            </File>
          </When>
        </Select>
      </File>
    </Component>
  </Product>
</PalmOSInstaller>
```



```

    </When>
    <When Resolution="320x320">
      <File Filename="con320x320.pdb">
        <Select Type="Language">
          <When Language="Spanish">
            <File Filename="con320x320sp.pdb"/>
          </When>
          <When Language="Swahili">
            <File Filename="con320x320sw.pdb"/>
          </When>
        </Select>
      </File>
    </When>
    <When Resolution="320x480">
      <File Filename="con320x480.pdb"/>
    </When>
    <When Resolution="480x320">
      <File Filename="con480x320.pdb"/>
    </When>
  </Select>
  <Select Type="Variant">
    <When Variant="PalmGear">
      <Resource Num="9900" Type="tSTR" Text="http://PalmGear.com/?12345"></Resource>
      <Resource Num="9910" Type="tSTR" Text="Buy at PalmGear"></Resource>
    </When>
  </Select>
</File>
<Select Type="Language">
  <When Language="English">
    <File Filename="conhelpENUS.pdb" InstallToCard="No" OTAExclude="true"/>
  </When>
  <When Language="Spanish">
    <File Filename="conhelpSP.pdb" InstallToCard="No" OTAExclude="true"/>
  </When>
  <When Language="Swahili">
    <File Filename="conhelpSW.pdb" InstallToCard="No" OTAExclude="true"/>
  </When>
</Select>
</Component>
<Component Platform="Windows" Description="Install conduit to your desktop computer."
Action="Execute">
  <File Filename="condsetup.exe"/>
</Component>
</Product>
</PalmOSInstaller>

```

The description file shows the power of this format. First, notice that directly under the <Component> layer, there are only two files: Aggression.prc and a help file. Inside the tags for Aggression.prc there are additional files to support various graphics resolutions. These are inside the <File> tags for Aggression.prc, which means that these are to be merged into Aggression.prc, resulting in a single file to be installed to the device. You can see that the game supports 160x160, 320x320, 320x480 and 480x320. In 160x160 and 320x320, the game has Spanish and Swahili merge files.

The help file also has multiple languages. If a user installs Aggression to a Tungsten T3 and chooses English, the installer will determine that the best combination for the device is to merge agg480x320.pdb into Aggression.prc and then to install Aggression.prc and conhelpENUS.pdb to the device. The landscape graphics were chosen due to the <Prefer> tag that indicated horizontal.

If a user installs to a Tungsten T3 and chooses Spanish, the installer will merge con320x320sp.pdb into con320x320.pdb, merge the result into Aggression.prc, then install Aggression.prc and conhelpSP.pdb onto the device.

The Full Package PSI file would contain all options and is meant for downloading to, and installing from, a desktop machine. Since bandwidth and download times are critical considerations on a handheld device, eight additional files would be generated for OTA install. These files would be pre-merged and contain only the final deliverable needed by the user such as Spanish at 160x160 resolution. The web site would use the generated PalmCall to ensure that the correct file is automatically downloaded by the user.

Here is another example of a simple PRC with a conduit:

```
<?xml version="1.0" encoding="UTF-8"?>
<PalmOSInstaller xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.palmsource.com/installer/psi.xsd">
  <Product Description="SplashID provides a safe-haven for all of your data." DisplayVersion="3.01"
ID="SplashID" Version="3.0.1.0">
    <Component Description="SplashID for Palm OS">
      <File InstallToCard="NotPreferred" Filename="SplashID.prc"/>
    </Component>
    <Component Platform="Windows">
      <File Action="Execute" Filename="SplashIDDesktopSetup.exe"/>
    </Component>
  </Product>
</PalmOSInstaller>
```

The PSI file will contain the PRC and the EXE file. It will install the PRC and run the EXE file. The EXE file can do whatever it needs to do, such as installing conduits and desktop components.

NOTE: A Mac conduit is not yet available, but if this PSI file were installed on a Mac, the Windows portion would simply be ignored. A developer could include a Mac platform section as well. The download will be larger, but there is only a single file to maintain and user confusion will be reduced.

For complete details of the XML Schema, it is best to refer to the schema that Package Builder references by default on the PalmSource website (currently www.palmsource.com/installer/psi.xsd).

Here is a high-level overview of what is in the schema and what will be supported, unsupported or ignored:

- Notes – Ignored. These are purely for the developer's sake.
- Publisher – Includes publisher name and contact information. Package Builder displays the publisher name during the install.
- Include – Allows you to include other PSML files. For instance, you may wish to put your Publisher in a separate PSML file and have all of your PSML files include that file.
- Global Components – You can specify a "Component" outside of a Product specification and then include it in a product specification by ID. This is useful if you want to specify multiple products within a single PSML file that share some common components.
- Product – Developers may specify one or more products in a single definition file. All products must have a unique ID, which can be the product title but must not contain spaces. By default, PSI files for all products will be built, but the developer can specify a single target product ID using the **-p** command-line switch.
 - Version – Required. The format is like that of an IP address; for example, 2.1.0.0 (major release, minor release, major bug level, minor bug level)

- DisplayVersion – Developers may specify a “DisplayVersion” if they prefer another format such as 2.1b. Displayed during device install time.
- Description – Stored in the PSI file but not currently displayed.
- PrimaryLanguage – All files are assumed to be in the specified language unless they are language overlays in Select/When Language blocks. Default is English.
- OTA – OTA must equal *true* for Package Builder to generate OTA files. Default is *false*.
- NoImplicitOutput – By default, Package Builder will create a default (Full Package) output file (PSI) based on the <project name>.PSI, which includes an output file for each variant found in Select/When Variant blocks. Developers can disable this by specifying NoImplicitOutput=”true”. Then only those output files explicitly defined will be created.
- Output definitions – Each product may declare one or more output files. There can be no more than one per variant. For example, a variant might be an online store, an OEM partner, or an affiliate. Anything for which you want a separate version of your product can be a variant. Each output definition can contain:
 - Filename – Synthesized if not specified.
 - Variant – The variant ID must not contain spaces. Leave Variant out for the default version.
 - Title – If specified, this title will be used instead of the default title, such as “Handmark WordSmith Pro”.
 - Description – If present, this will be stored in the PSI file instead of the standard product description.
 - OTA – If not specified, this is the same as the OTA state of the product. Having this here means that you could, for instance, not generate OTA for anyone except PalmSource.
 - DownloadURL – If specified, this DownloadURL will be used instead of the default DownloadURL for the product.
- EULA – Ignored. Not yet implemented.
- Prefer – Currently only specifies an orientation preference. This is only needed if the product provides separate graphics files for portrait and landscape modes. If installed on a device such as the Tungsten T3 or Tapwave Zodiac, the installer will use this attribute to choose the preferred graphics.
 - Orientation – Must be either “Portrait” or “Landscape”.
- Component – A product may have one or more components.
 - ID – This may be used to include the component within multiple Product definitions.
 - Ref – Includes a Component previously declared elsewhere in the file or in an included file. The Ref value should be the Component ID.
 - Title – Ignored. Useful as a note by the developer.
 - Description – Ignored. Useful for describing an optional component.
 - Platform – Options are PalmOS, Windows, Mac and Desktop. Desktop is useful for installing HTML or PDF files to any desktop platform. Default is PalmOS.
 - File –
 - Filename – The filename is actually optional since File can contain other files. If a filename is specified, contained files are merged into this one. Otherwise, only one of the contained files is installed. The remaining options set the state for all contained Files.
 - InstallToCard – Can be No, NotPreferred, OK, Preferred or Required. If the user’s device supports VFS and the product contains InstallToCard=”OK” components, Package Builder desktop will allow the user to select to install as much of the product as possible to their memory card.

InstallToCard="Required" files will not be installed at all if the device does not support VFS.

- Action – Can be Install or Execute. Default is Install. Currently Palm OS targets must be Install and Desktop targets must be Execute.
- OTAExclude – If true, this file will not be included in any OTA output files.
- Resources – Developers may specify one or more text resources to merge into the base file. These are specified by Type/ID.
- Select – Select allows the developer to specify conditional inclusion. Select types are Resolution, Language and Variant. A developer can use a Select statement to either overlay the primary file or install additional files with graphics appropriate to the user's device screen resolution, alternate language resources or custom resources for a variant, such as specifying an alternate purchase URL for a web store. Both files and resources may be overlaid based on a Select statement.
- Select – Besides including Select statements within File blocks, one or more Select statements may also be directly included in Components.

PSI File Format Details

A PSI file is essentially a ZIP file with a description file. The description file tells the Package Installer how the files are meant to be used or merged.

Sample HTML and JavaScript for Your Application Download Page

When you create a web page where users can download your application, you will need to use properly formatted HTML files and scripting language. Following is sample code to help you.

When a customer clicks a link to install your application, this is what happens:

1. Your Web browser checks for Package Installer on your device or desktop (depending on your situation). If Package Installer is not present, your web browser displays a message saying you need the Package Installer, and provides a button to press to get the Package Installer. If the Package Installer is present but is an old version, an updated version will be installed.

Sample code for checking for Package Installer for a desktop customer:

```
<OBJECT id=mng1 classid="CLSID:9B8D3E79-A732-4ec0-AEEE-8AF8CDF10D8A"
CODEBASE="http://www.palmsource.com/installer/PSIWebStub.dll#Version=1,0,0,0" height="40" width="300"><param name="src"
value="Aggression.psi">
<a href="Aggression.psi">Click here to download Aggression</a>
</object>
```

In this example, the OBJECT tag checks for version 1.0.0.0, and if it is outdated or doesn't exist, it goes to the location for downloading the Package Installer Desktop software (PSIWebStub.dll). If you do not specify height and width, a reasonable default will be used. In the HTML file above, you can see that the name of the file is passed to the control using the "<param>". Just in case the user is not using Internet Explorer, a

standard link is included as an alternate. It will not be shown to users of the ActiveX control.

Sample code for checking for Package Installer for an OTA customer:

```
palmcall:psix.appl?v=1.0.0.0(http://www.web_store.com/installer/d
emo/ota/apps_page.html)(http://www.web_store.com/installer/demo/o
ta/get_newer_installer.html)
```

In this example, the PalmCall string checks whether Package Installer version 1.0.0.0 or newer is present on the customer's device. If Package Installer is already present, the PalmCall string says to jump to the next web page for application installation (in this example, the page is apps_page.html). If Package Installer is outdated, the PalmCall string says to jump to the get_newer_installer.html page, which in this example is the page where the customer can download an up-to-date version of Package Installer.

Have a look at the sample pages to see how to use a JavaScript "setTimeout" function and document.location to have the browser automatically try the PalmCall and redirect the user to the PackageInstaller.prc file if it is not detected.

2. Once the customer has an up-to-date version of Package Installer, they can then download the requested application.

Here is sample code for downloading an application OTA. In this example, PalmCall specifies that PalmSource Package Installer should automatically select between English, German, and Spanish, and between low resolution and high resolution. The correct PalmCall is always generated by Package Builder when OTA packages are built. You should never try to create the capability parameters by hand but you can freely change the final portion of the PalmCall if the base URL should happen to change.

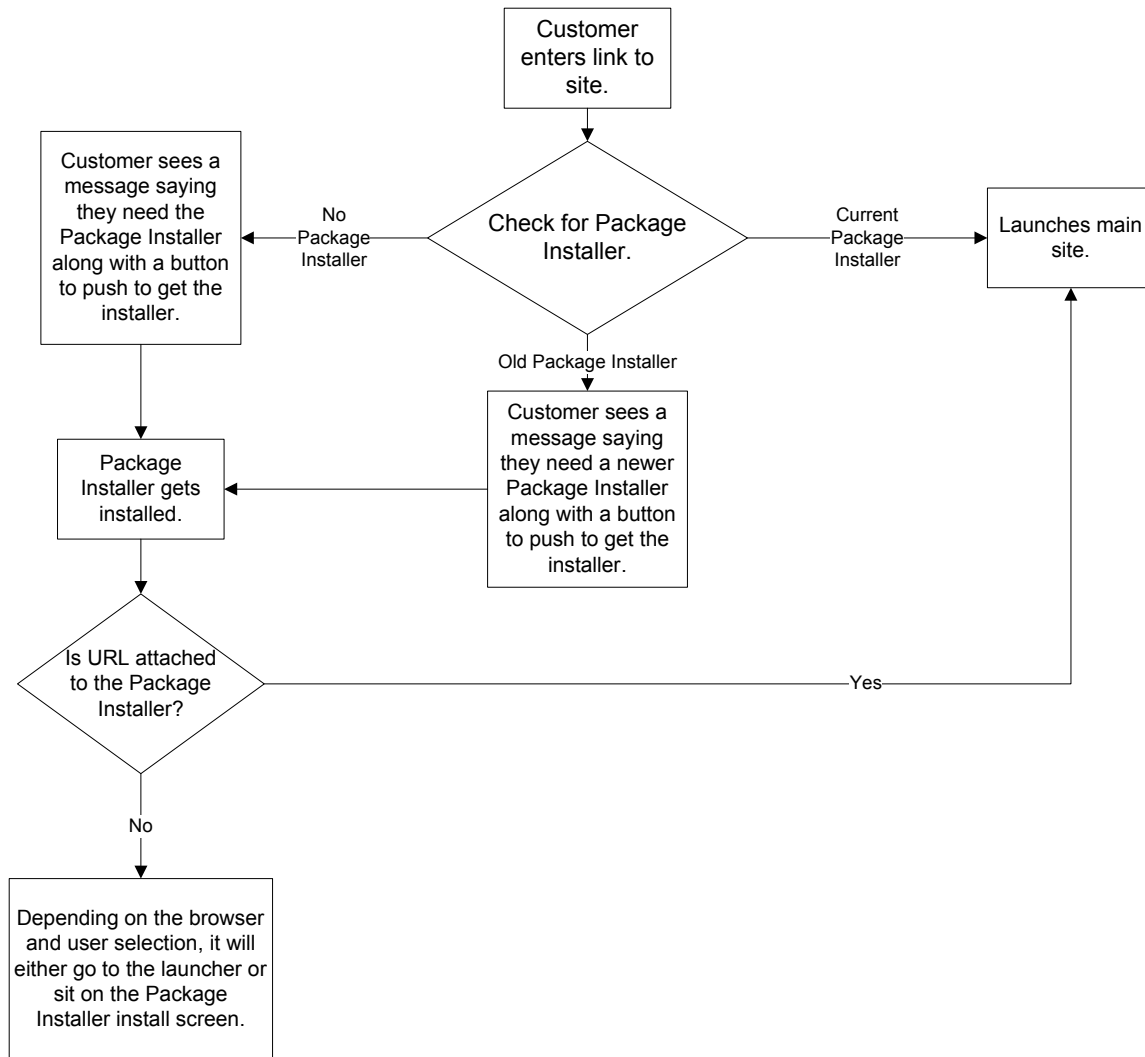
```
<a
href="PalmCall:psix.appl?p=a:en,de,es,b:1414,2828&n=MyApp&u=http:
//www.web_store.com/your_app/demo/ota">Click here to download
MyApp</a>
```

The URL specifies the host name and base directory where the My-App OTA files are located.

More on PalmCall

In order for customers to be able to download PSI files directly to their devices, they first need to install the Package Installer Client. This is called "bootstrapping." Through the use of JavaScript and a PDA-side mechanism called "PalmCall", this process can be automated.

The following diagram illustrates the over the air (OTA) page flow.



What is PalmCall

In HTML, everything has a URL that begins with a tag that indicates where the address lives. For example, "http://" indicates a web page, and "ftp://" indicates a file that can be accessed by "File Transfer Protocol". Similarly, "PalmCall:" indicates a Palm OS application to execute.

For the purpose of this description, all that is really important to understand about PalmCall is that it allows us to run the Package Installer Client (PIC) on the customer's device directly from a web page. If you would like more information about PalmCall, refer to the Web Clipping Tutorial.

Automating the user experience

PSI is a great new PalmSource Installer file format that automatically determines the best combination of features for the user's device and downloads and installs only those features. But before you can take advantage of the PSI format, you need to ensure that the Package Installer Client (PIC) has been installed to the customer's device. The goal is something approximating an ActiveX install experience. Here is the strategy:

1. Display a web page that shows a "Checking to see if your handheld is set up properly" message and try to run PIC with a special command that checks the version.

2. If the customer has a current version of PIC installed, PIC will redirect the browser to the catalog where PSI files can be downloaded.
3. If the customer has an older version of PIC installed, they will be redirected to an "Install Newer PIC" page.
4. If the customer does not have PIC installed, the command will fail. After 10 seconds, the JavaScript on the page will automatically redirect the user to the "Install PIC" page.
5. In both case 3 and 4, the user will install PIC. After it is installed, PIC will redirect the customer to the catalog to download PSI files.

In summary, a user with a current version of PIC will be quickly redirected to the catalog. A user with an old copy of PIC will get the newer copy and then be redirected to the catalog. A user without PIC will wait about 10-15 seconds for the auto-detect to fail, install PIC, then be redirected to the catalog.

PIC bootstrapping details

This section is for experienced web developers. Here is the exact code you need to add to your site to enable the PIC bootstrap mechanism. In this example, we have 4 pages:

1. index.html - The auto-detect is done here.
2. get_installer.html - This is where the customer ends up if the auto-detect failed, indicating that PIC is not yet installed.
3. get_newer_installer.html - The customer gets redirected to here if their copy of PIC is out of date.
4. apps.html - The customer ends up here once they have PIC installed.

index.html

Here is the page:

```
<html>
<head>
  <title>
    Detecting PackageInstaller
  </title>
  <noscript>
    <meta http-equiv="REFRESH" content=
      "0;
URL=PalmCall:psix.appl?v=1.0.2.3(http://mysite.com/apps.html)(http://mysite.com/get_newer_installer.html)">
    </noscript>
    <script language="javascript">
      function SetupTimer()
      {
        document.location =
"PalmCall:psix.appl?v=1.0.2.3(http://mysite.com/apps.html)(http://mysite.com/get_newer_installer.html)";
        setTimeout("GetPackageInstaller()",10000)
      }
      function GetPackageInstaller()
      {
        document.GetInstaller.submit();
      }
    </script>
  </head>
  <body onLoad="SetupTimer()">
    <script language="javascript">
```

```

        document.write("<font size='+1'><table align='center' height='100%' width='100%'><tr><td align='center'
valign='middle'><form name='GetInstaller' action='get_installer.html' method='post'>Detecting the PalmSource
Package Installer. This may take up to 30 seconds...</form></td></tr></table></font>");
    </script>
    <noscript>
        <font size='+2'>
            <p>
                We need to make sure your handheld is setup properly.
            </p>
            <p>
                If your download does not begin within 10 seconds, please <a href=
                "PackageInstaller.prc">tap here</a> to install the PackageInstaller.
            </p>
        </font>
    </noscript>
</body>
</html>

```

Let's go through this a bit at a time:

```

<noscript>
    <meta http-equiv="REFRESH" content=
    "0; URL=PalmCall:psix.appl?v=1.0.2.3(http://mysite.com/apps.html)(http://mysite.com/get_newer_installer.html)">
</noscript>

```

Hopefully the customer has JavaScript enabled since this results in a much smoother install process. If they do not, a refresh is used to execute the PalmCall to PIC. The above PalmCall checks to see if PIC version 1.0.2.3 or later is installed. If so, the browser is redirected to apps.html. If not, it is redirected to get_newer_installer.html.

```

<script language="javascript">
    function SetupTimer()
    {
        document.location =
        "PalmCall:psix.appl?v=1.0.2.3(http://mysite.com/apps.html)(http://mysite.com/get_newer_installer.html)";
        setTimeout("GetPackageInstaller()",10000)
    }
    function GetPackageInstaller()
    {
        document.GetInstaller.submit();
    }
</script>

```

Here, a couple of JavaScript functions are declared that will be needed if the customer's browser has JavaScript enabled. The SetupTimer function first tries to redirect the browser to the PalmCall, then sets a timer for 10 seconds. If the user is still on this page 10 seconds from now, the second function will be called and the GetInstaller form will be submitted, taking the browser to get_installer.html.

```

    <body onLoad="SetupTimer()">
        <script language="javascript">
            document.write("<font size='+1'><table align='center' height='100%' width='100%'><tr><td align='center'
valign='middle'><form name='GetInstaller' action='get_installer.html' method='post'>Detecting the PalmSource
Package Installer. This may take up to 30 seconds...</form></td></tr></table></font>");
        </script>

```

If JavaScript is enabled, the SetupTimer function is run as soon as the page is displayed. Next, document.write is used to output the form that the GetPackageInstaller function will use to redirect the browser if PIC is not installed.


```

<noscript>
  <font size='+2'>
    <p>
      We need to make sure your handheld is setup properly.
    </p>
    <p>
      If your download does not begin within 10 seconds, please <a href=
        "PackageInstaller.prc">tap here</a> to install the PackageInstaller.
    </p>
  </font>
</noscript>

```

If JavaScript is not enabled, this mechanism is used instead. Hopefully the redirect used above will work, but insert a manual link just in case.

get_installer.html

Here is the page:

```

<html>
<body>
  <script>
    if (document.referrer=="")
      document.location = 'index.html';
    else {
      document.write("<table align='center' height='100%' width='100%'><tr><td align='center' valign='middle'>");
      document.write("<p>This site is designed for use only by wireless handheld devices. Before you can
proceed, you need to install the PalmSource Package Installer.</p>");
      document.write("<a href='PackageInstaller.prc'>Install the Package Installer</a>");
      document.write("</td></tr></table>");
    }
  </script>
</body>
</html>

```

If the user is on this page, it confirms that JavaScript is enabled, since the JavaScript path on index.html is the only path that leads here.

The first thing to do is make sure that the customer is not reloading this page. If the customer didn't just come here from some other page, the user gets bounced back to the index. That way, if the customer installs the PackageInstaller.prc and then reloads the browser, they will be taken back to the auto-detect page instead of being stuck here. If you have PHP, ASP, or something similar, you may want to do this on the server side. The easiest way is to check to see if the customer got here via a "post" or not. The index page posts to here also if the server sees that this is a GET instead of a POST, so the browser can be sent to the index immediately.

Then put a link on this page so that the customer can install PIC.

get_newer_installer.html

Here is the page:

```
<html>
<body>
  <script>
    if (document.referrer!="")
      document.location = 'index.html';
    else {
      document.write("<table align='center' height='100%' width='100%'><tr><td align='center' valign='middle'>");
      document.write("<p>Your copy of the PalmSource Package Installer is too old.</p>");
      document.write("<a href='PackageInstaller.prc'>Install the latest Package Installer</a>");
      document.write("</td></tr></table>");
    }
  </script>
  <noscript>
    <table align='center' height='100%' width='100%'><tr><td align='center' valign='middle'>
      <p>Your copy of the PalmSource Package Installer is too old.</p>
      <a href='PackageInstaller.prc'>Install the latest Package Installer</a>
    </td></tr></table>
  </noscript>
</body>
</html>
```

Unlike the previous page, there is no guarantee that JavaScript is available here, so the no-JavaScript case must be handled as well. The code is straightforward. Like the previous page, the code returns to the index if the customer has come here directly. Otherwise, a link is displayed so that they can install PIC.

apps.html

The apps page is up to you. The fact that the customer is here guarantees that PIC has been properly installed.

Embedding the target page in PIC

There is an additional issue to consider. After the customer installs PIC, they are likely to be staring at the "please install" screen or at the application launcher, both of which are undesirable. To prevent this, use the '-u' command in Package Builder, to generate a slightly customized copy of PIC to put on your site. For instance, you might do this:

```
PackageBuilder -u http://mysite.com/apps.html
```

The output is a copy of PackageInstaller.prc (PIC) with your target URL embedded in a resource. When the customer installs this PIC to their device, it will detect that it has just been installed from a browser and redirect the browser to your target.