



InstallAnywhere 2011

User Guide

Legal Information

Book Name: InstallAnywhere 2011 User Guide
Part Number: IA-1200-UG01
Product Release Date: May 17, 2011

Copyright Notice

Copyright © 2006-2011 Flexera Software, Inc. and/or InstallShield Co. Inc. All Rights Reserved.

This product contains proprietary and confidential technology, information and creative works owned by Flexera Software, Inc. and/or InstallShield Co. Inc. and their respective licensors, if any. Any use, copying, publication, distribution, display, modification, or transmission of such technology in whole or in part in any form or by any means without the prior express written permission of Flexera Software, Inc. and/or InstallShield Co. Inc. is strictly prohibited. Except where expressly provided by Flexera Software, Inc. and/or InstallShield Co. Inc. in writing, possession of this technology shall not be construed to confer any license or rights under any Flexera Software, Inc. and/or InstallShield Co. Inc. intellectual property rights, whether by estoppel, implication, or otherwise.

All copies of the technology and related information, if allowed by Flexera Software, Inc. and/or InstallShield Co. Inc., must display this notice of copyright and ownership in full.

Trademarks

Flexera Software, Inc., FlexNet, FlexNet Connect, FlexNet Publisher, InstallAnywhere, InstallShield, LaunchAnywhere, SpeedFolder, and Zero G Software are registered trademarks or trademarks of Flexera Software, Inc. and/or InstallShield Co. Inc. in the United States of America and/or other countries. All other brand and product names mentioned herein are the trademarks and registered trademarks of their respective owners.

Restricted Rights Legend

The software and documentation are “commercial items,” as that term is defined at 48 C.F.R. §2.101, consisting of “commercial computer software” and “commercial computer software documentation,” as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.2702, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §227.2702-1 through 227.7202-4, as applicable, the commercial computer software and commercial computer software documentation are being licensed to U.S. government end users (A) only as commercial items and (B) with only those rights as are granted to all other end users pursuant to the terms and conditions set forth in the Flexera Software, Inc. standard commercial agreement for this software. Unpublished rights reserved under the copyright laws of the United States of America.

Contents

1	InstallAnywhere 2011 Help Library	15
	Using Help	16
	Documentation Conventions	16
	Contacting Us	18
2	Getting Started	19
	InstallAnywhere Interface	19
	Authoring Environments	19
	Project Wizard	20
	Advanced Designer	21
	Setting the Default InstallAnywhere Startup Authoring Environment	23
	Opening an InstallAnywhere Project	23
	Creating a New InstallAnywhere Project	23
	Opening an Existing InstallAnywhere Project	24
	Switching to a Different Project in the Advanced Designer	25
	InstallAnywhere Editions	26
3	Concepts of InstallAnywhere Installer Development	27
	Actions	29
	About Actions	29
	About Action Groups	30
	Action Customizers and the Action Execution Sequence	31
	About Action Customizers	31
	About the Action Execution Sequence	31
	About Get User Input Panels	32
	Input Methods/Component Types	32
	Bidi Orientation and Text Reading Order	33

Contents

<i>Font and Color Settings</i>	33
<i>Results Variables</i>	33
<i>Defaults</i>	35
<i>VAR_BOOLEAN_X Variable</i>	36
Build Tools	36
Custom Code	37
Custom Code Basics	37
About Custom Code and Variables	38
About Plug-ins	38
DIM References	39
Hosts	39
Install Sets, Features, and Components	40
Install Sets	40
Features	40
Components	41
Standalone Components	41
Dependencies	41
Shared Components	42
Installer Modes	42
Graphical User Interface (GUI) Installers	42
GUI Localization	43
Look and Feel	43
Console Installers	44
Silent Installers	45
About Response Files and Silent Installers	45
Installer Types	47
Java Virtual Machines	48
About VM Selection	49
How the Installer's Launcher Selects a VM	51
About the Installer's VM Search	52
About the Launcher's VM Selection Behavior at Run-Time	52
About the Choose Java VM Panel	55
When Are VM Packs Installed?	55
About Java VM Selection Criteria	56
LaunchAnywhere	58
Localization	58
Localization Basics	59
Dynamic and Static Text	59
Best Practices for Localizing	60
External Resource Bundles	60
Magic Folders	61
Merge Modules	62

Project File	65
Rules	65
Source Paths	66
SpeedFolders	67
Templates	67
Uninstaller	68
About Uninstallers and Custom Code	68
About Uninstallers and Variables	68
Feature Uninstall	69
Uninstaller for Merge Modules and Multiple Products	69
Similarities Between Pre and Post Install and Uninstall Tasks	71
Variables	72
Variable Notation	72
Magic Folders and Variables	73
Methods of Setting InstallAnywhere Variables	74
Evaluation of InstallAnywhere Variables	75
Searching for InstallAnywhere Variables	76
Selecting Variables from a List	80
4 InstallAnywhere Tutorials	81
Building an Installer Using the Project Wizard	82
Creating a New Project	82
Defining the Installation Tasks	84
Adding Files	85
Choosing the Main Class	87
Setting the Classpath	88
Building the Installer	89
Testing the Installer	90
Building an Installer Using the Advanced Designer	91
Creating a New Project	93
Adding Pre-Install Actions	96
Defining the Installation Tasks	98
Adding a LaunchAnywhere Executable to the Install Task	101
Adding Post-Install Actions	105
Building the Installer	110
Testing the Installer	112
5 Creating Basic Installers	115
Creating a Basic Installation Project	116
Opening a Project	117
Creating a New Project	117
Opening an Existing Project	117
Customizing the Pre-Install Panels and Actions	118

Contents

Defining the Install Task	119
Starting to Define the Install Task.....	120
Adding Files to a Project.....	121
Adding Install Actions.....	122
Adding LaunchAnywhere Executables to the Install Task.....	123
Recreating the InstallAnywhere Uninstaller and Associated Components.....	125
Customizing the Post-Install Task	126
Organizing Features and Components	128
Best Practices for Components	129
Adding Components	129
Assigning Files to Components	132
Removing Empty Components	132
Integrating Components Already Installed on Target System.....	133
Adding Features	134
Assigning Files to Features	134
Assigning Components to Features.....	135
Defining Rules	136
Assigning a Rule to the Installer.....	137
Assigning a Rule to an Action	137
Assigning a Rule to a Group of Actions	138
Building Complex Rule Expressions	139
Customizing Built-in Rules	141
Customizing a Check File/Folder Attributes Rule.....	142
Customizing a Check If File/Folder Exists Rule	143
Customizing a Check Platform Rule.....	143
Customizing a Check Running Mode Rule	144
Customizing Evaluate Custom Rule Rules.....	146
Customizing a Compare InstallAnywhere Variable Numerically Rule	147
Rules Are Evaluated Multiple Times.....	148
Implementing Maintenance Mode	149
Enabling and Configuring Maintenance Mode	150
Enabling Maintenance Mode	151
About Maintenance Mode Action Groups	153
Customizing Maintenance Mode Text and Images	157
About Check Running Mode Rules	159
Specifying Instance Management Options	159
About the Uninstaller / Maintenance Mode Launcher	164
Maintenance Mode End User Experience	165
Initial Experience	166
Adding a Feature User Experience.....	167
Removing a Feature User Experience	168
Repairing a Product User Experience	169
Uninstalling a Product User Experience.....	170
Maintenance Mode User Experience on a Machine With Multiple Installed Instances.....	171

Customizing the Uninstaller	171
About Uninstaller Customization	172
Adding Uninstall Categories and Actions to the Uninstall Task	174
Preventing an Uninstall Category From Being Uninstalled	177
Reordering Uninstall Categories in the Uninstall Task	178
Building Installers	178
Creating and Editing Build Configurations	179
About Build Configurations	180
Creating a New Build Configuration	184
Creating Migrated Build Configurations	184
Renaming a Build Configuration	185
Copying a Build Configuration	186
Removing a Build Configuration	187
Adding a Build Configuration to the Project Build	187
Using Tags to Customize Build Configurations	188
<i>Determining Whether a Project Element is Included in a Build Configuration</i>	189
<i>Creating New Tags</i>	191
<i>Assigning Tags to Project Elements</i>	191
<i>Associating Tags to Build Configurations</i>	192
<i>Searching for Tags</i>	193
<i>Importing Tags from Merge Modules</i>	194
<i>Specifying Locale Settings</i>	194
Defining Build Targets	194
Setting Build Distribution Options	199
Creating Web Installers	199
Creating CD-ROM/DVD Installers	199
Creating Merge Modules	200
<i>Advertising Merge Module Installer Variables</i>	201
Building Installers Using Build Configurations	203
<i>Building a Selected Build Configuration</i>	203
<i>Building Build Configurations Enabled for Project Build</i>	203
<i>Building All Build Configurations</i>	204
<i>Running the Command-Line Build</i>	204
Testing Installers	205
Working with VM Packs.....	206
Downloading VM Packs	206
Installing VM Packs	207
Bundling a VM Pack	207
Creating Launchers for Java Applications.....	208
Improving Installation Performance.....	210
Launching Additional Installers	211
Installing Fonts	211
Team Development	211

Contents

Enabling Source Paths	212
Adding and Removing Source Paths	212
Updating the Location of Files and Resources	213
Working with Source Control Management Software	214
Referencing Developer Installation Manifests (DIMs)	214
Adding a DIM Reference	214
Customizing a DIM Reference	216
Removing a DIM Reference	217
Creating a Shortcut to a File within a DIM Reference	218
Using FlexNet Connect	218
Checking for InstallAnywhere Update Notifications	219
Installing InstallAnywhere Updates	219
6 Adding Advanced Procedures to Installers	221
Working with Variables	223
Setting Variables in the Advanced Designer	223
Checking the Value of a Variable	225
Encrypting Variable Values	225
Creating VM Packs	226
Using the Create VM Pack Utility	227
Creating a VM Pack Manually	228
VM Pack Structure	229
VM Pack Properties	230
Making a VM Pack FIPS-Compliant	231
Controlling the VM Used to Run the Installer	231
Controlling the Install of Bundled VMs	232
Preventing the Install of the Bundled VM	233
Installing the Bundled VM When the Project Includes a Launcher	234
Installing the Bundled VM Only When No Valid VM Exists	234
Preventing the Uninstall of the Bundled VM	234
Changing the Bundled VM Install Folder	235
Changing the Bundled VM Install Subfolder	235
Controlling the VM Your Launchers Use	236
Customizing the VM Search Settings for Your Launchers	236
Customizing the VM Search Paths and Patterns	238
Customizing Windows Search Paths	239
Customizing Java Executable Patterns for Windows Installs	241
Customizing Unix Search Paths	242
Customizing Java Executable Patterns for Unix Installs	245
Customizing Individual Launcher Settings	246
Causing a Launcher to Use the VM That Ran the Installer	246
Causing a Launcher to Use the VM Selected During the Installer's VM Search	247
Causing a Launcher to Perform an Independent VM Search	247

Using the Choose Java VM Panel.....	248
Adding a Choose Java VM Panel.....	249
Allowing Users to Choose Only the VMs the Installer's Search Finds	249
Allowing Users to Choose the Bundled VM	250
Allowing Users to Search for VMs in Non-Standard Locations	251
Allowing Users to Browse for a VM on the Local System	251
JVM Spec Files.....	252
Generating Response Files.....	255
Generating Log Files	257
Getting User Input	258
Using the Get User Input - Simple Panel	258
Using the Get User Input - Advanced Panel	260
Using Command-Line Arguments with Installers and Uninstallers.....	261
Setting Project Version at Build Time	263
Checking Disk Space During Installation	265
Deploying Your Product with FlexNet Connect.....	267
Adding an Enable Update Notifications Action	268
Customizing an Enable Update Notifications Action.....	268
Connecting Your Product with the Publisher Site	270
Removing an Enable Update Notifications Action.....	271
Localizing Projects and Installers	272
Generating Multi-Language Installers	273
Localizing Resources	273
Localizing Custom Installer Labels.....	274
Localizing the Splash Screen.....	274
Localizing the Get User Input Panels.....	277
Adding an External Resource Bundle	279
Referencing an External Resource Key	280
Packaging and Executing Custom Code	280
Support for Signed JARs as Dependencies.....	282
Calling InstallShield MultiPlatform APIs in InstallAnywhere	283
Packaging Custom Code as a Plug-in	285
Internationalizing Custom Code.....	286
Digitally Signing Installers	287
Setting Installation Rollback Options.....	287
Determining a Successful Installation	288
Troubleshooting.....	289
Debugging During Installer Development	289
Directing Installer Debug Output	290
Using the Output Debug Information Action	292
Debugging Using Display Message Panel.....	293
Debugging LaunchAnywhere Launched Executables.....	293

Contents

Debugging During Post Development	294
Debugging a Win32 Installer	294
Debugging a Unix/Linux or Pure Java Installer	295
Debugging a Mac OS X Installer	296
Other Debugging Issues	296
Mac OS X Magic Folder Behavior	296
EBCDIC Encoding Issues	297
Reviewing Debug Information	299
Exit Codes	302
Installer Exit Codes	302
Build Exit Codes	304
Application Server Support	305
Database Server Support	306

7 Reference 307

Advanced Designer Interface	308
Project	312
Info	313
Description	316
File Settings	318
Platforms	320
Mac OS X	321
Windows	323
UNIX	326
System i (i5/OS)	329
Pure Java	331
Locales	333
Rules	335
JVM Settings	337
General Settings Tab	338
Installer Settings Tab	339
Search Panel Settings Tab	343
Variables	353
Log Settings	356
Advanced	359
Maintenance Mode	360
Instance Management	362
Manage Tags	369
Rollback Settings	370
Windows WOW64 Emulator Settings	371
Installer UI	373
Look & Feel	373
General UI Settings	374
Installer Steps	378

<i>Install Progress Panel</i>	381
<i>Maintenance Mode Labels</i>	384
<i>Installer Icon</i>	386
Billboards	387
Help	390
Organization	391
Install Sets	392
Features	393
Components	394
Modules	399
DIM References	403
Hosts	405
<i>Application Server Hosts</i>	406
<i>Database Server Hosts</i>	409
Pre-Install	411
Install	412
Post-Install	417
Pre-Uninstall	419
Uninstall	420
Post-Uninstall	423
Build	424
Build Configurations Tab	424
<i>Build Targets Subtab</i>	428
<i>Distribution Subtab</i>	435
<i>Tags Subtab</i>	438
<i>Locales Subtab</i>	440
Build Log Tab	441
Project Wizard Interface	442
New Project	443
Open Project	445
Project Info	446
Add Files	447
Choose Main	448
Classpath	449
Build Installer	450
Try Installer	451
Dialog Boxes	452
About Dialog Box	454
Add Files to Project Dialog Box	455
Advertise Variables Dialog Box	457
Configure Input Items Dialog Box	459
Change Disk Space and Name Dialog Box	460
Choose an Action Dialog Box	461
Choose Icon Dialog Box	462

Contents

Choose Variable Dialog Box	464
Configure Search and Replace Strings Dialog Box	465
Create New Project Dialog Box	466
Create VM Pack Dialog Box	467
Custom Rules Dependencies Dialog Box	470
Edit Advertised Variables Dialog Box	470
Filter File Dialog Box	472
InstallAnywhere Merge Module Import Assistant Dialog Box	474
InstallAnywhere Preferences Dialog Box	475
General Settings	475
Resources	477
Source Paths	478
Updates	479
Designer Colors	480
LaunchAnywhere Properties Dialog Box	480
Manage Instances Dialog Box	481
New Project Dialog Box	481
Open Project Dialog Box	483
Open Project File Dialog Box	483
Read/Modify XML File Dialog Box	484
RPM Specification Settings Dialog Box	485
Save New Project As Dialog Box	486
Search Results Dialog Box	487
SWVPD Registry Settings Dialog Box	491
Uninstaller Properties Dialog Box	492
Menus	492
File	493
Edit	494
Wizard	495
Help	495
Actions	496
Install Actions	497
Create Alias, Link, Shortcut	499
Create Alias, Link, Shortcut to DIM File	500
Create DIM Reference	501
Create Folder	503
Create LaunchAnywhere for Java Application	505
Create Uninstaller	507
Deploy WAR/EAR Archive	509
Enable Update Notifications	510
Expand Archive	514
Expand Archive (7-zip)	515
Expand Archive (TAR)	516
Install Archive	517

Install File	517
Install Merge Module	518
Install SpeedFolder	520
Run SQL Script	521
Trigger Rollback	521
Uninstall Actions	522
General Actions	523
Delete Folder	530
Execute Ant Script	531
Execute Command	532
Execute Custom Code	533
Execute Script/Batch File	534
Find Component in Registry	535
Modify Text File - In Archive	537
Modify Text File - Multiple Files	538
Modify Text File - Single File	539
Read/Modify XML File	541
Show Message Dialog	544
Panel Actions	545
Choose Install Sets	549
Choose Java VM	550
Display HTML	552
Pre-Install Summary	553
Console Actions	555
System i (i5/OS) Actions	556
Choose Remote System i (i5/OS) Install Folder	558
Get System i (i5/OS) Login Credentials	559
System i (i5/OS) Command	560
System i (i5/OS) Find Component in RAIR	561
System i (i5/OS) Install File	562
System i (i5/OS) Integrated File System (IFS)	563
System i (i5/OS) Library	564
System i (i5/OS) Licensed Program	565
System i (i5/OS) Object	567
System i (i5/OS) Process Remote Install (Merge Modules)	568
System i (i5/OS) Program	569
System i (i5/OS) Program Temporary Fix (PTF)	570
Plug-In Actions	571
ExecuteAsRoot	572
ExtractToFile	573
PropertiesFileReader	574
Common Action Properties	575
Common Customizer Settings	575
Panel Action Settings	576

Contents

Get User Input Panels	579
Get User Input - Simple	579
Get User Input - Advanced	581
Rules Reference	583
Check File/Folder Attributes Rule	585
Check If File/Folder Exists Rule	587
Check Platform Rule	587
Check Running Mode Rule	588
Evaluate Custom Rule Rule	590
Compare InstallAnywhere Variables Numerically Rule	592
Variables	593
Standard InstallAnywhere Variables	593
\$IA_BROWSE_FOLDERS\$ Variable	604
LAX Properties	606
Magic Folders	611
Localization Reference	615
Language Codes	615
Common Localizable Elements	617
Files and File Formats	622
Product Registry	623
Install Log File Format	624
Manifest Files	625
Response Files	627
BuildProperties.xml File	629
BuildProperties.xml File Settings in the Advanced Designer	629
buildproperties.properties File	636
buildproperties.properties File Settings in the Advanced Designer	636
Command-Line Reference	641
Installer and Uninstaller Command-Line Arguments	641
Maintenance Mode Command Line Options	643
Build Command-Line Arguments	644
Launcher Command-Line Arguments	650
InstallAnywhere Ant Task Reference	650
Custom Code APIs	660
Project Automation API for Maintenance Mode and Instance Management	661
Index	667

InstallAnywhere 2011 Help Library

Welcome to InstallAnywhere—the most powerful multi-platform software deployment solution available. It deploys any application to any platform. InstallAnywhere 2011 combines amazing control, performance, and an easy to use graphic interface to create software installers that install perfectly every time, lowering development and support costs.

Help Library Organization

Information about using InstallAnywhere is presented in the following sections:

Table 1-1 • InstallAnywhere Help Library

Section	Description
Getting Started	Describes the InstallAnywhere interface authoring environments and explains the InstallAnywhere editions.
Concepts of InstallAnywhere Installer Development	Introduces you to the basic and advanced concepts of InstallAnywhere installer development.
InstallAnywhere Tutorials	Provides guided exercises, using both the Project Wizard and Advanced Designer interfaces, to explore using InstallAnywhere.
Creating Basic Installers	Provides procedures and suggestions about common situations when using InstallAnywhere to build a basic installer.
Adding Advanced Procedures to Installers	Explains how to add more advanced procedures to an InstallAnywhere installer.
Reference	Reference information for using InstallAnywhere, including the Project Wizard and Advanced Designer interfaces.

Using Help

When you have questions about this product, first consult the InstallAnywhere Help Library, which is the complete user's guide for using InstallAnywhere. You can open the InstallAnywhere Help Library by selecting **InstallAnywhere Help** from the Advanced Designer **Help** menu or by clicking the **Show Help** button in the lower left corner of the Advanced Designer.

Web-Based Online Help

Web-based online help is available to you 24 hours a day, seven days a week, on our Web site at:

<http://helpnet.flexerasoftware.com>

Documentation as PDF

InstallAnywhere documentation is also available as a PDF. Visit our Documentation Center site at:

<http://support.flexerasoftware.com/document/Default.aspx>

Documentation Conventions

In this documentation, reader alert and style conventions are used to bring your attention to specific information or help you identify information.

Reader Alert Conventions

Reader alerts are used throughout this documentation to notify you of both supplementary and essential information. The following table explains the meaning of each alert.

Table 1-2 • Reader Alert Conventions

Image	Alert Name	Description
	Note	Notes are used to draw attention to pieces of information that should stand out.
	Important Note	Important notes are used for information that is essential for users to read.
	Caution	Cautions indicate that this information is critical to the success of the desired feature or product functionality.
	Tip	Tips are used to indicate helpful information that could assist you in better utilizing the desired function or feature.
	Edition-Specific Note	Edition-specific notes indicate that the information applies to a specific edition of a product (such as Professional or Premier edition).

Table 1-2 • Reader Alert Conventions (cont.)

Image	Alert Name	Description
	Task	The Task graphic indicates that procedural instructions follow.

Style Conventions

The following style conventions are used throughout this documentation.

Table 1-3 • Style Conventions

Style	Example	Description
User Interface Elements	On the File menu, click Open .	User interface elements appear in bold when referenced in tasks.
Variables	<i>fileName</i>	Variables appear in italics.
Code	#define HWND_BROADCAST 0xffff	Code snippets appear in a monospace typeface.
User Inputted Text	Type \$D(install) .	Text that is to be entered as a literal value is displayed in a monospace typeface, in bold, and in blue.
File Name and Directory Paths	My files are located in the C:\MyDocuments\SampleCode directory.	File names and directory paths are presented in a monospace typeface.
.INI File Text	Insert the line LimitedUI=Y into the file to display only the Welcome dialog box when the Windows Installer package is run.	Text in .INI files is presented in a monospace typeface.
Command-Line Statements	To run the installation silently, enter: <code>install.exe -i silent</code>	Command-line statements and parameters are presented in a monospace typeface.
Environment Variables	Set the value of the <code>windir</code> environment variable to your	Environment variables are presented in a monospace typeface.
Examples	Create two groups, one called Admins and the other called General .	Examples are presented in bold.
Functions	FeatureAddItem adds a new feature to a script-created feature set.	Functions are presented in bold.

Table 1-3 • Style Conventions (cont.)

Style	Example	Description
Properties	In the Name property, enter a name for this custom control that is unique among all of the controls in your project.	Properties are presented in bold.
Screen Output	If you type an incorrect parameter, the message The system cannot find the path specified. is displayed.	Screen output (from a log file or from the console) is displayed in a monospace typeface, and in blue.
Links	Obtain the latest modules, white papers, project samples, and more from: http://www.yourcompany.com	Links appear in blue.

Contacting Us

You may contact us from anywhere in the world by visiting our Web site at:

<http://www.flexerasoftware.com>

Getting Started

This section describes the InstallAnywhere interface authoring environments and explains the InstallAnywhere editions.

- [InstallAnywhere Interface](#)
- [InstallAnywhere Editions](#)

InstallAnywhere Interface

This section describes the InstallAnywhere interface authoring environments, how to set interface preferences, and how to open an InstallAnywhere project.

- [Authoring Environments](#)
- [Setting the Default InstallAnywhere Startup Authoring Environment](#)
- [Opening an InstallAnywhere Project](#)

Authoring Environments

InstallAnywhere offers two authoring environments: the Project Wizard and the Advanced Designer.

- **Project Wizard**—The Project Wizard guides you through creating a new installer quickly. Using the six-step Project Wizard, you can build your first installer in less than five minutes. See [Project Wizard](#).
- **Advanced Designer**—The Advanced Designer provides much finer control over installer functionality. It enables you to define multiple Install Sets, add customized splash screen graphics, execute commands from within the installation process, set up Build Configurations, and use many other advanced features. See [Advanced Designer](#).

Chapter 2: Getting Started

InstallAnywhere Interface



Note • You can build an installer in the Project Wizard and then switch to the Advanced Designer to add functionality. See [Switching to the Advanced Designer Environment](#).

Project Wizard

Developers can build their first installer in less than five minutes with the six-step InstallAnywhere Project Wizard. This intuitive design tool also sets the **Classpath** and finds the **Main Class** for a Java application automatically.

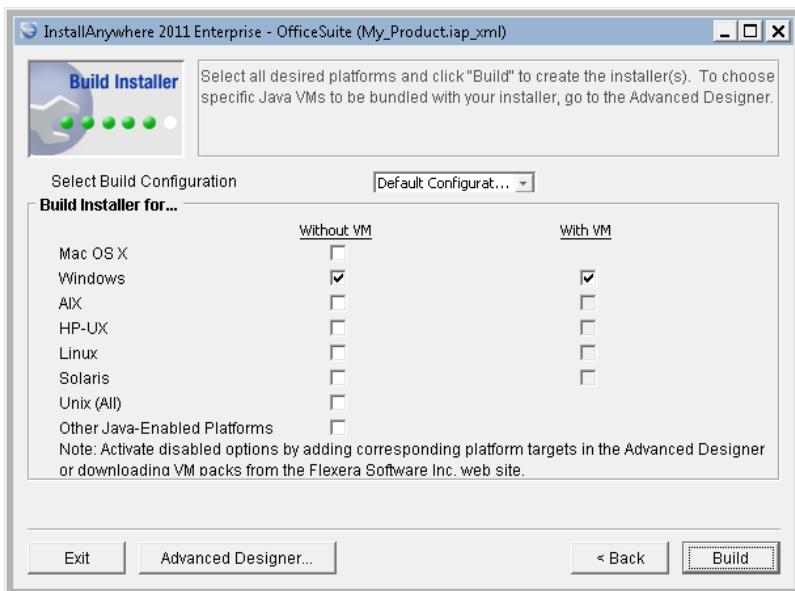


Figure 2-1: Build Installer Frame of the InstallAnywhere Project Wizard



Note • The Project Wizard is the default InstallAnywhere interface. To change the default interface or to switch interfaces when editing a project, see:

- [Setting the Default InstallAnywhere Startup Authoring Environment](#)
- [Switching to the Advanced Designer Environment](#)

More Information

For more information on using the Project Wizard, see the following:

- To become familiar with using the Project Wizard interface, perform the [Building an Installer Using the Project Wizard](#) tutorial.
- For detailed information on the Project Wizard interface, see [Project Wizard Interface](#).

Advanced Designer

The InstallAnywhere Advanced Designer has an intuitive, graphical interface, which allows developers to manage all aspects of their installer project. All the features of InstallAnywhere are available in this easy to use integrated development environment.

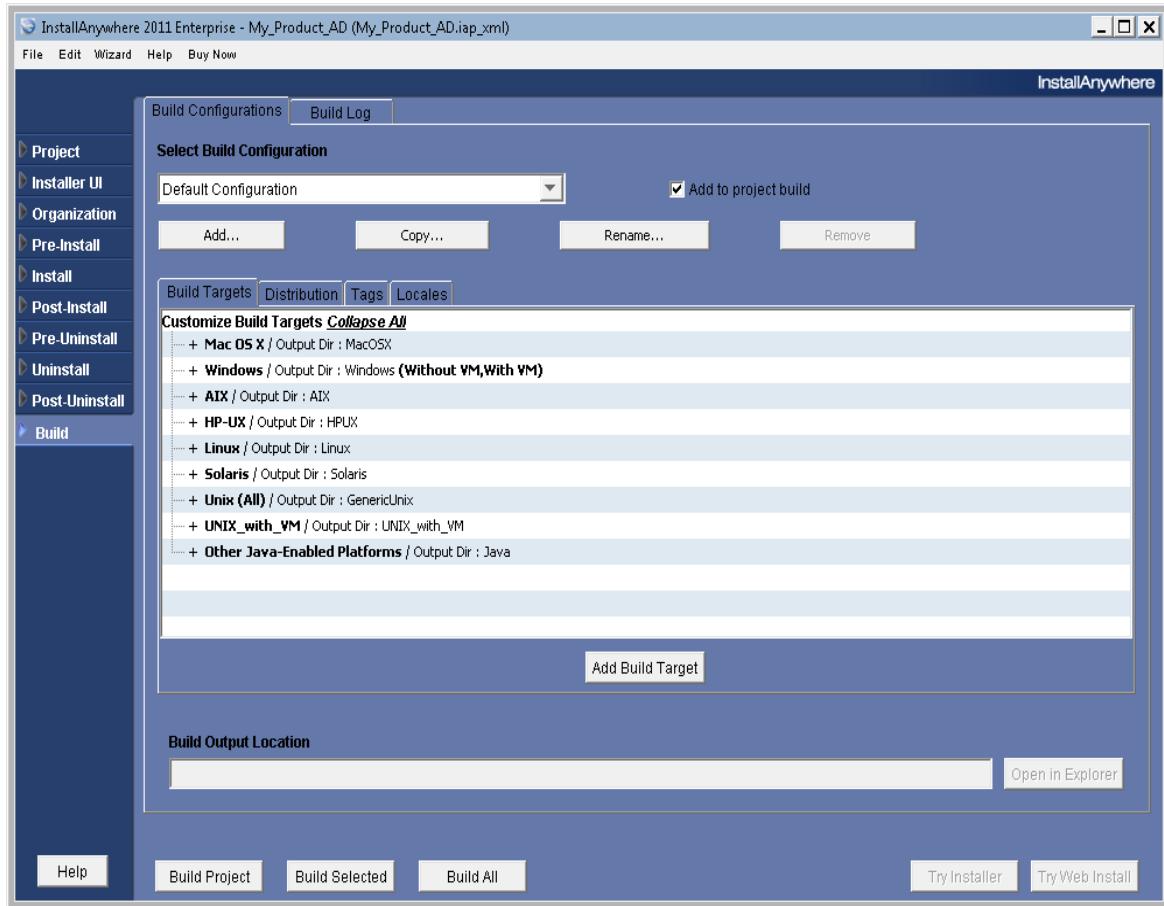


Figure 2-2: Build Task of the InstallAnywhere Advanced Designer

While the Project Wizard leads developers through a few simple steps to create an installer, the Advanced Designer gives greater precision, enabling developers to access all the powerful features of InstallAnywhere.

The Advanced Designer gives developers precise control over all the design options of an installation project. Developers can use this interface to assign files and actions to feature sets, which enables the end user to define which files are installed, and to add rules to selectively install different files or different installation locations depending on the target platform.

Chapter 2: Getting Started

InstallAnywhere Interface

Switching to the Advanced Designer Environment

While using the Project Wizard, you can switch to the Advanced Designer by clicking the **Advanced Designer** button.



Task: *To switch from the Project Wizard to the Advanced Designer:*

1. Open an InstallAnywhere project in the Project Wizard.
2. Click the **Advanced Designer** button at the bottom of the current Project Wizard frame.

Advanced Designer Tasks

The Advanced Designer is divided into “Tasks” represented by tabs, found along the left-hand side of the window, that contain settings specific to each installation project. These tasks are listed in the following table. Each of these task tabs contains sub-tabs that offer greater fine-tuning of InstallAnywhere’s features.

Table 2-1 • Advanced Designer Tasks

Task	Description
Project	Settings related to your specific project. These include general settings, file settings, and localization settings.
Installer UI	Set the look and feel for the installer by adding background images, billboards, and other graphical components.
Organization	Manages Install Sets, Features, Components, and Merge Modules.
Pre-Install	An ordered sequence of panels and actions that occur before file installation.
Install	Manage File installation tree and install time actions.
Post-Install	An ordered sequence of panels and actions that occur after file installation.
Pre-Uninstall	An ordered sequence of panels and actions that occur before file uninstallation.
Post-Uninstall	An ordered sequence of panels and actions that occur after file uninstallation.
Build	Manage build settings, including bundling of a Java Virtual Machine.

More Information

For more information on using the Advanced Designer, see the following:

- To become familiar with using the Advanced Designer interface, perform the [Building an Installer Using the Advanced Designer](#) tutorial.
- For detailed information on the Advanced Designer interface, see [Advanced Designer Interface](#).

Setting the Default InstallAnywhere Startup Authoring Environment

Initially, the Project Wizard is the default InstallAnywhere authoring environment. You can set the default authoring environment on the **InstallAnywhere Preferences** dialog box.



Task: *To set the default InstallAnywhere authoring environment:*

1. Launch InstallAnywhere and open an existing project or create a new project.
2. Switch to the **Advanced Designer** interface.
3. On the **Edit** menu, click **Preferences**. The InstallAnywhere Preferences dialog box opens.
4. On the **General Settings** tab, set the **InstallAnywhere Startup** option to **Project Wizard** or **Advanced Designer**.

Opening an InstallAnywhere Project

This section explains how to create a new InstallAnywhere project or open an existing project. The steps you take to perform these tasks depends upon whether you are using the Project Wizard or the Advanced Designer interface.

- [Creating a New InstallAnywhere Project](#)
- [Opening an Existing InstallAnywhere Project](#)

Creating a New InstallAnywhere Project

The steps you take to create a new project depends upon whether you are using the Project Wizard or the Advanced Designer interface.

Project Wizard

To create a new InstallAnywhere project when using the Project Wizard, perform the following steps:



Task: *To create a new InstallAnywhere project:*

1. Launch InstallAnywhere. The **New Project** frame of the Project Wizard opens.
2. Select the **Create New Project** option.
3. Select a template to use.
4. Click **Save As** and enter a name for this new project.
5. Click **Next** and proceed with the steps listed in [Creating Basic Installers](#).

Chapter 2: Getting Started

InstallAnywhere Interface

Advanced Designer

To create a new InstallAnywhere project when using the Advanced Designer, perform the following steps:



Task: *To create a new InstallAnywhere project:*

1. Launch InstallAnywhere. The InstallAnywhere About dialog box opens.
2. Click **New Installer**. The **Create a New Project** dialog box opens.
3. Select a template to use.
4. Click **Save As** and enter a name for this new project.
5. Click **OK**. The Advanced Designer interface opens.
6. Proceed with the steps listed in [Creating Basic Installers](#).

Opening an Existing InstallAnywhere Project

In the Project Wizard, the first dialog box allows you to either create a new project or open an existing project.

Project Wizard

To open an existing project from the Project Wizard, perform the following steps.



Task: *To open an existing project from the Project Wizard:*

1. Launch InstallAnywhere. The **New Project** frame of the Project Wizard opens.
2. Select the **Open Existing Project** option.
3. Select the project from the **Recent Projects** list or click **Other Project** to specify a different project.
4. Click **Next**.
5. If you want to open this project in the Advanced Designer, click **Advanced Designer**.
6. Proceed with the steps in listed in [Creating Basic Installers](#).

Advanced Designer

To open an existing project from the Advanced Designer, perform the following steps.



Task: *To open an existing project from the Advanced Designer:*

1. Launch InstallAnywhere. The InstallAnywhere About dialog box opens.
2. Click **Open Existing**. The **Open Project File** dialog box opens.
3. Locate and select the project file you want to open.
4. Click **Open**.
5. Proceed with the steps in listed in [Creating Basic Installers](#).

Switching to a Different Project in the Advanced Designer

To switch from one project to another in the Advanced Designer, perform the following steps.



Task: *To switch to a different project in the Advanced Designer:*

1. Open an existing project in the Advanced Designer.
2. On the **File** menu, click **Open**. The **Open Project File** dialog box opens.



Note • If you have made edits to the existing project, you are prompted to save the current project before proceeding.

3. Locate and select the project file you want to open.
4. Click **Open**.
5. Proceed with the steps in listed in [Creating Basic Installers](#).

InstallAnywhere Editions

There are two editions of InstallAnywhere. Each edition is designed to meet product deployment needs for the different types of customers. This manual describes the features available in the Enterprise Edition. Although some specific differences are called out throughout the manual, for a detailed chart of which features are available in each edition, check the InstallAnywhere Web site.

Enterprise Edition

Enterprise Edition provides the powerful configuration options, user interaction, and client/server features. It simplifies complex installations and provides comprehensive developer customization. The Enterprise Edition is available in English and Japanese. Each Enterprise Edition has full international support to create installers in 31 different languages.

Standard Edition

Standard Edition offers more features and customization than any other product in its class. It is ideal for desktop application deployment and has international support for 9 languages.

Concepts of InstallAnywhere Installer Development

InstallAnywhere provides many complex options for creating installer functionality. This section introduces you to the basic and advanced concepts of InstallAnywhere installer development.

Table 3-1 • InstallAnywhere Installer Development Concepts

Topics	Description
Actions	Explains InstallAnywhere's support for an extensible action architecture that gives developers the ability to perform operations during installation. Also includes information on Action Groups, Action Customizers, and the Action Execution Sequence.
Build Tools	Explains InstallAnywhere's command-line build tool.
Custom Code	Discusses custom code concepts including custom code actions, panels, consoles, and rules, as well as the use of InstallAnywhere variables in custom code and the creation of custom code plug-ins.
DIM References	Explains the purpose of Developer Installation Manifest (DIM) references.
Hosts	Explains the three different types of hosts that InstallAnywhere can apply rules, deploy files, and execute actions on.
Install Sets, Features, and Components	Explains the purpose of install sets, features, and components in InstallAnywhere installer development.
Installer Modes	Describes the available installer modes: GUI, Console, and Silent.
Installer Types	Describes the two types of installers that you can create with InstallAnywhere: Web and CD.

Table 3-1 • InstallAnywhere Installer Development Concepts (cont.)

Topics	Description
Java Virtual Machines	Explains the Java VM selection options available to setup authors, and the logic behind the Java VM resolution performed by InstallAnywhere launchers and installers.
LaunchAnywhere	Describes the purpose and function of LaunchAnywhere and how to specify a particular VM on the command line.
Localization	Addresses the differences between dynamic and static text, built-in locale files and external resource bundles, and provides tips on how to best work with InstallAnywhere's localization resources.
Magic Folders	Describes how InstallAnywhere uses Magic Folders to define installation locations.
Merge Modules	Covers the different types of Merge Modules and how they can be integrated with your project and installers.
Project File	Describes the InstallAnywhere project file.
Rules	Describes the purpose of InstallAnywhere rules.
Source Paths	Discusses how InstallAnywhere implements source paths in the project file.
SpeedFolders	Explains the benefits of using SpeedFolders to increase installation speed and memory efficiency.
Templates	Explains the purpose of using InstallAnywhere templates to create new installer projects.
Uninstaller	Provides additional discussion of concepts related to InstallAnywhere uninstallers such as how uninstallers relate to features, variables, custom code, and Merge Modules.
Variables	Explains the purpose of and benefits of using variables in an InstallAnywhere installation.

Actions

InstallAnywhere supports an extensible action architecture that gives developers the ability to perform operations during installation. Information about actions is presented in the following sections:

Table 3-2 • Topics About Actions

Topic	Description
About Actions	Explains InstallAnywhere's support for an extensible action architecture.
About Action Groups	Explains how you can use Action Groups to apply rules to a collection of actions to make InstallAnywhere projects more manageable and easier to understand.
Action Customizers and the Action Execution Sequence	Discusses the use of action customizers to define the detailed behavior of InstallAnywhere actions, how InstallAnywhere determines the sequence in which actions are run, and how Get User Input panels work.

About Actions



Important • Actions may only be added while working in the Advanced Designer.

InstallAnywhere supports an extensible action architecture that gives developers the ability to perform operations during installation. Some of these actions are as simple as installing files and folders and as complex as creating modifying text files, executing custom code during the installation process, or extracting contents from a compressed file.

Actions may occur in the background, not requiring any user input, or may require user input:

- **General/Install Actions** do not require any user input.
- **Panel Actions and Console Actions** request user input. Panel Actions display a graphic element that requests user input.
- **Console Actions** display a command line request.

Actions may be added to the Advanced Designer tasks that represent real-time operations being accomplished by the installer: **Pre-Install**, **Install**, **Post-Install**, **Pre-Uninstall**, and **Post-Uninstall**.

Chapter 3: Concepts of InstallAnywhere Installer Development

Actions

The following actions are available:

Table 3-3 • Actions

Action	Description
Install Actions	Install actions are used to deploy the installer payload to the target machine.
Uninstall Actions	Uninstall actions are used to customize the flow of the InstallAnywhere Uninstaller.
General Actions	Most General actions occur transparently to the end user. They do not require any user input.
Panel Actions	Panel actions are requests for user input that appear inside the graphical installer wizard.
Console Actions	Console actions are command-line requests for user input and are used during command-line installation.
System i (i5/OS) Actions	System i actions are general, install, panel, and console actions specifically for installers that run on the i5/OS operating system on System i.
Plug-In Actions	Custom code can be integrated with the InstallAnywhere Advanced Designer and appear as a Plug-In action in the Choose an Action dialog box.

About Action Groups

Action Groups allow you to apply rules to a collection of actions and to make InstallAnywhere projects more manageable and easier to understand. They allow the developer ability to logically group a set of actions or panels in the Install task as well as the Pre-Install, Post-Install, Pre-Uninstall and Post-Uninstall. Rules applied to the Action Group affect all the actions or panels contained inside it. They can be extremely helpful with large complex installations that contain numerous actions and panels.

Action Customizers and the Action Execution Sequence

The topics in this section provide more information about action customizers and the action execution sequence.

Topic	Description
About Action Customizers	Discusses the use of action customizers to define the detailed behavior of InstallAnywhere actions and apply rules to an action.
About the Action Execution Sequence	Explains how InstallAnywhere determines the sequence in which actions are run.
About Get User Input Panels	Describes how Get User Input panels work with a focus on how InstallAnywhere installers manage the results variables that store the end users responses/input.

About Action Customizers

Once an action has been added, its customizer appears in the bottom half of the Advanced Designer. Developers may customize this action by modifying the customizer.

- **Properties**—The **Properties** tab enables developers to add file locations and variables to the action as well as details about how developers would like the action to run.
- **Rules**—The **Rules** tab enables developers to add specific rules to an action. Once an action has a rule added to it, its icon is modified to display a small R on it, so developers can see that there is a rule associated with it. This icon badge may help test or modify an installer if there are a long list of actions, some with rules and some without. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use the **Tags** tab to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use the **Rollback** tab to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

About the Action Execution Sequence

Actions are executed in the order that they appear in the task from top to bottom. Actions may be reordered either by use of the up and down arrows. The left and right arrows move actions out of or into folders.

Pre-Install/Uninstall actions are executed before their corresponding Install/Uninstall actions, which are executed before Post-Install/Uninstall actions. Pre-Install actions are generally used to determine what to install on the target system or even if the installation should occur. Actions added from the Install task define what will occur on the target system, like creating folders, expanding archives, or moving files. Post-Install actions are often actions, like showing the ReadMe file or launching the application that was just installed, that require the existence of recently installed files. Actions that occur within Pre-Install or Post-Install will not be automatically uninstalled.



Note • When an action is included in Pre-Install or Post-Install task, it may be executed more than once if an end user repeatedly clicks Previous and Next. This could cause several errors for actions that modify files, such as Modify Text File - Single File or Register Windows Service. To prevent such errors, execute such actions in the Install task only.

About Get User Input Panels



Edition • The Get User Input panels are available only in InstallAnywhere Enterprise edition.

The **Get User Input - Simple** and **Get User Input - Advanced** actions provide a non-programmatic way to show a simple Java dialog box that allows your installer to interact with the end user.

A simple prompt guides the end user through the selection or completion of various Java elements (text boxes, check boxes, buttons, pop up menus, or lists). The end-user-supplied input is returned and stored in an InstallAnywhere variable that you are free to name.

The values stored in the InstallAnywhere variable can subsequently be used to customize the installation process, to apply rules to the installer (enabling only certain files in the installer to be installed or certain actions performed during installation), and so on.

The following information about **Get User Input** panels is provided:

- [Input Methods/Component Types](#)
- [Bidi Orientation and Text Reading Order](#)
- [Font and Color Settings](#)
- [Results Variables](#)
- [Defaults](#)
- [VAR_BOOLEAN_X Variable](#)

Input Methods/Component Types



Edition • The Get User Input panels are available only in InstallAnywhere Enterprise edition.

Input methods or components types refers to the type of control on a **Get User Input** panel—the manner in which end user input will be accepted and the manner in which the labels will be displayed to the end user.

- **Input methods** available on the **Get User Input - Simple** panel include Text Fields, Checkboxes, Radio Buttons, Pop-up Menus or Lists.
- **Components** available on the **Get User Input - Advanced** panel include these input methods plus Choice Groups, File Choosers, and Labels.

Bidi Orientation and Text Reading Order



Edition • The Get User Input panels are available only in InstallAnywhere Enterprise edition.

Text (Bidi) orientation and text reading order controls pertain to the left-to-right or right-to-left orientation of the components and text on a **Get User Input** panel. There are three possible values for the text orientation controls: **Based on Locale** (default), **Left-to-Right**, and **Right-to-Left**.



Note • InstallAnywhere generally manages text direction based on the installer's locale. Therefore, panels and controls that do not have a specific setting take their text orientation from the natural flow of the language selected. You can override the text orientation for particular controls by setting their bidi orientation or text orientation settings.

Font and Color Settings



Edition • The Get User Input panels are available only in InstallAnywhere Enterprise edition.

Most controls/input methods that you can add to a **Get User Input** panel can have their font and color settings customized. By clicking **Configure** on the **Get User Input - Simple** panel or **Configure Selection** on the **Get User Input - Advanced** panel, you open a configuration dialog box, where you can specify font and color settings for the controls you select for your panel.

The default for all these options is **Use Default**. To customize a font or color setting, click to deselect the **Use Default** option and then click the corresponding **Choose** button. This opens a **Choose Font** or **Choose Color** dialog in which you can specify the typeface, font size, and color to use.

Results Variables



Edition • The Get User Input panels are available only in InstallAnywhere Enterprise edition.

Results variables store the results of the end user's input. The default value for the Results Variable name is `USER_INPUT_RESULTS`.

The variable name of the Results Variable also forms the base name of additional InstallAnywhere variables that will be used to store individual end-user input choices from the end-user input panel. The convention for the naming of the additional variables is to append an underscore and an integer number to the end of the base name. The number, starting at one (1), increases by one (1) for each additional variable that is needed to store data.

The number of additional variables derived from the Results Variable base name depends upon the Input Method/Component Type that you have selected for the panel, the choices made by the end user at install time, and the number of values listed in Defaults.

The following table describe how user input is recorded in variables for each input type:

Table 3-4 • User Input Results By Input Type

Input Type	Description
Textfields	<p>The Results Variable is the base name for the values returned by the Get User Input panel. Entries in the textfields will be returned in a comma-separated list surrounded by quotes and stored in the Results Variable.</p> <p>In addition, additional variables derived from the Results Variable will store the values from the individual textfields. For example, if the Results Variable is \$X\$, and there are three textfields populated with the values Peter, Paul, and Mary, when the end user clicks Next during the install, the following InstallAnywhere variables and values exist:</p> <pre>\$X\$="Peter", "Paul", "Mary" \$X_1\$=Peter \$X_2\$=Paul \$X_3\$=Mary</pre> <p>However, if only two values were inserted (Peter and Mary), the values are:</p> <pre>\$X\$="Peter", "", "Mary" \$X_1\$=Peter \$X_2\$= \$X_3\$=Mary</pre>
Check Boxes	<p>The Results Variable is the base name for the values returned by the Get User Input panel. A comma-separated list containing the Value for choices from selected checkboxes (a blank entry in the comma-separated list will be included for check boxes that were not selected) will be returned and stored in the Results Variable. Additional variables derived from the Results Variable will store the values for each check box that was selected.</p> <p>For example, if the Results Variable is \$X\$, and there are three check boxes with the labels Paul, John, and George (from the Values for choices list) and Paul and George are selected when the end user clicks Next during the install, the following InstallAnywhere variables and values exist:</p> <pre>\$X\$="Paul", "", "George" \$X_1\$=Paul \$X_2\$= \$X_3\$=George</pre>

Table 3-4 • User Input Results By Input Type

Input Type	Description
List	<p>The Results Variable is the base name for the values returned by the End-user Input panel. A comma-separated list containing the Value for choices from selected list entries (a blank entry in the comma-separated list will be included for checkboxes that were not selected) will be returned and stored in the Results Variable.</p> <p>Additional variables derived from the Results Variable will store the values for each list entry that was selected.</p> <p>Lists behave somewhat differently than checkboxes. For example, if the Results Variable is \$X\$, and there are three list entries with the labels Paul, John, and George (from the Values for choices list) and Paul and George are selected when the end user clicks Next during the install, the following InstallAnywhere variables and values will exist:</p> <pre>\$X\$="Paul","","George" \$X_1\$=Paul \$X_2\$= \$X_3\$=George</pre>
Radio Buttons and Popup Menus	<p>The Results Variable is the base name for the values returned by the Get User Input panel. A single value based both on the Value for choices value and what was selected is returned and stored in the Results Variable.</p> <p>Additional variables are not derived from the Results Variable unless Defaults are specified.</p> <p>For example, if the Results Variable is \$X\$, and there are three radio buttons with the labels Michael, Matthew, and David (from the Values for choices: list) and Michael is selected when the end user clicks Next during the install, the following InstallAnywhere variables and values exist:</p> <pre>\$X\$="Michael","",""</pre>

Defaults



Edition • The Get User Input panels are available only in InstallAnywhere Enterprise edition.

The effect of **Defaults** on **Get User Input** panels also differ by input method.

- **If the Input Method is Textfields**, the values listed in **Defaults** will be set as the defaults for the textfields. The first default item will be used for the first textfield, the second item for the second textfield, etc.
- **If the Input Method is any other option** (Checkboxes, Radio Buttons, a Pop-up menu, or a List), you may choose the default to be either **Selected** or **Not Selected**, and **Set Variable**. Choosing **Set Variable** opens a dialog box requesting the name of an InstallAnywhere variable. If the variable resolves to true at the time of installation, the component will be selected.

If there are more defaults listed than there are entries in Values for choices, these extra defaults will be stored in additional variables derived from the Results Variable base name. For example, if there are three items in Values for choices and there are four defaults, the fourth default would be stored in a variable with a _4 derivation. Extra default values do not affect the String returned in the base Results Variable.

VAR_BOOLEAN_X Variable



Edition • The Get User Input panels are available only in InstallAnywhere Enterprise edition.

There is also a variable to correspond with the *USER_INPUT_RESULTS_1*, *USER_INPUT_RESULTS_2*, and *USER_INPUT_RESULTS_3*. If your installer requires localization, the values and characters that correspond with this user input panel may vary from language to language. The *VAR_BOOLEAN* variable deals with this by providing a true (1) or false (0) value if choices are selected. To use the example above:

```
$X$="Paul","","George"  
$X_1$=Paul  
$X_2$=  
$X_3$=George  
$X_BOOLEAN_1$=1  
$X_BOOLEAN_2$=0  
$X_BOOLEAN_3$=1
```

Build Tools

InstallAnywhere includes a command-line build tool. This tool (*build.exe* in the InstallAnywhere application folder) accepts command-line options that can customize the build settings at the command line or specify stored settings in a build properties file.

InstallAnywhere also includes the ability to integrate with Ant via the InstallAnywhere Ant task.



Note • The use of *iaant.jar* requires Java 1.4 or newer.

More information on Ant can be found on the Apache Foundation's Ant Project Web site: <http://ant.apache.org>.

Custom Code

The majority of tasks needed to deploy the application can be handled using InstallAnywhere's built-in actions and panels. If there is functionality not covered by InstallAnywhere's built-in actions and panels, developers can create their own custom components using the Custom Code API.

Topic	Description
Custom Code Basics	Discusses custom code actions, panels, consoles, and rules.
About Custom Code and Variables	Explains the interaction between custom code and InstallAnywhere variables.
About Plug-ins	Provides an overview of custom code plug-ins and the advantages of packaging custom code as a plug-in.

Custom Code Basics

InstallAnywhere offers an open Application Programming Interface (API) which allows developers to write Java code that can run within InstallAnywhere's architecture. Using the API also gives access to additional functionality in InstallAnywhere, such as its unique Variables and resource loading features. Developers can use the API to create custom actions and GUI elements that seamlessly interact with and extend the InstallAnywhere framework.

All custom code that is going to run within the InstallAnywhere framework must be written in Java. The major forms of custom code are actions, panels, consoles, and rules:

Table 3-5 • Major Forms of Custom Code

Custom Code	Description
Custom Code Actions	These actions run within InstallAnywhere's action framework, alongside the default InstallAnywhere actions.
Custom Code Panels	These panels run within InstallAnywhere's graphical interface during the installation process. The developer can use this mechanism to add panels to the Installer not provided in InstallAnywhere's default panels.
Custom Code Consoles	These actions run within InstallAnywhere's console interface during the installation process. Developers can use this mechanism to add custom console elements to the Installer.
Custom Code Rules	These rules are evaluated when the action they are associated with is about to be executed. Custom Code rules need to return a Boolean value defining if the action is to be run.

InstallAnywhere provides sample custom code actions, panels, and console actions. These can be found in the CustomCode folder in the root installation directory of InstallAnywhere. These samples can be used as examples of how to implement InstallAnywhere custom code using the API. Additionally, there is a wide variety of custom code actions and panels located on the InstallAnywhere web site.

About Custom Code and Variables

Custom Code can both get and set InstallAnywhere variables. This ability can be useful in two cases:

- **A project can set InstallAnywhere variables, which can be used as parameters for Custom Code.**
For example, if a Custom Code Action is designed to know the directory to which the end user was installing, use the Set InstallAnywhere variable action to set PARAM_1 to \$USER_INSTALL_DIR\$. Then, in the Custom Code Action, call `InstallerProxy.substitute("$PARAM_1$")` to determine its value.
- **InstallAnywhere variables may also be used to store return values from Custom Code Actions.**
These variables can be queried by other actions or rules. To do this, call the method `InstallerProxy.setVariable("PARAM_1", <OBJECT>)` to set the InstallAnywhere variable's value. Remember that these variables are treated as strings, and will have `java.lang.Object.toString()` called upon them to determine their value in InstallAnywhere. Once PARAM_1 has been set it can be accessed in the normal way, using `$PARAM_1$` in a later InstallAnywhere rule or action.

A common problem when accessing the values of InstallAnywhere variables from within custom code is the improper use of the `getVariable()` and `substitute()` methods. The most common problem is the use of `getVariable()` on a Magic Folder object (for instance, `$USER_INSTALL_DIR$`) and then trying to cast this object to a string. Instead, use `substitute()`, which will resolve the contents of the variable to a string, instead of casting the object.



Note • When each method in a custom action or plug-in is invoked, it will not have access to any argument (InstallAnywhere) variables specified if they are implemented as a plug-in (those set in the plug-in customizer). If the variables are implemented as a custom code action, any InstallAnywhere variables the method uses need to be set and finalized in the previous phase of the installation (or externally for a dynamic InstallAnywhere Merge Module).



Note • For information on variables, see [Variables](#).

About Plug-ins

Developers can register custom code as plug-ins with the InstallAnywhere Advanced Designer. This feature allows properly packaged custom code to be integrated into the design environment, where it will appear on the **Choose an Action** dialog box under the **Plug-Ins** tab. Plug-ins are stored within the `<InstallAnywhere>/plugins` folder. The advantages of packaging custom code are:

- **Can be added as regular actions**—The developer can add them as regular actions without having to specify the JAR and class.

- **Utilizes InstallAnywhere variables defined in the plug-in action**—While custom code actions typically require a separate **Set Install Anywhere Variable** action to define variable values used by the custom code action, a plug-in allows the setting of variables at the action level, thus simplifying the install by reducing the number of actions required.
- **Easily portable**—Plug-ins are easily portable across development teams.



Note • *InstallAnywhere Support is interested in distributing plug-ins developed by our users. If you have written a plug-in that you think would be useful to other developers, and you would like to share it, please contact the InstallAnywhere Support team.*

DIM References

Developer Installation Manifest (DIM) references allow InstallAnywhere developers to include the output of Flexera Software's installation collaboration products in their installers. DIMs are XML files that describe specific product subsystems and their requirements. Application developers create DIMs using InstallAnywhere Collaboration (a plug-in module for Eclipse) or InstallShield Collaboration (a plug-in module for Microsoft Visual Studio .NET).

Hosts



Edition • Hosts options are only available in the Enterprise Edition. Standard Edition uses only the Operating System host, which requires no additional configuration in the Organization and Install tasks.

InstallAnywhere can apply rules, deploy files, and execute actions on three different types of hosts:

- **Operating System host** is the default host, transparent to InstallAnywhere Standard Edition users, that contains files, launchers, and actions that your installers deliver to the target system.
- **Application Server hosts** target an application server such as Geronimo, JBoss, Resin, Sun Application Server, TomCat, WebLogic, and WebSphere.
- **Database Server hosts** target a database server such as MySQL, Oracle, MS SQL Server, DB2, and several others.

Install Sets, Features, and Components

Install Sets, Features, and Components are some of the most important concepts to understand when using the InstallAnywhere installer development environment.

InstallAnywhere provides levels of granularity for end user installation options. Install Sets and Features may be selected by the end user. Install sets are groupings of features such as Typical Install or Minimal Install. Features are meant to identify specific units of functionality of your product. While both Install Sets and Features are made up of files, there is not necessarily a direct correlation between these larger organizational groupings and the files.

Components are groupings of the specific files and actions of your product, and are invisible to the end user. A component may also include a group of registry changes or other elements needed to make a feature work properly. Components are used for the organized sharing of resources, for versioning, are uniquely identified, and are the organization tool of the installer developer, not the installer user. A developer could create a component for each feature in the application, a component for shared libraries used by all the features, and a component for the help system.

Though developers may assign files, folders and actions directly to Product Features, it is best to think of features as groupings of components. Install sets such as Typical Install or Minimal Install are groupings of Features. The interaction between the three levels should be addressed when planning the options to present to the end user.

Install Sets

Install Sets are the simplest and broadest organizational concept within InstallAnywhere. Install Sets represent high-level, easily selectable, installation options. End Users can choose only one install set. These sets are generally options such as: Typical and Minimal, or Client only and Client and Server. End Users select their desired Install Set using the Choose Install Set panel or console. Install Sets are made up of Features.

Features

Features are a logical grouping of capabilities in a product. Features are visible to the end user only when the user chooses to customize an Install Set. Features are effective if developers want to give their end users fine-grained choices about what to install.

Features may be hierarchical, and there can be as many as desired, but there needs to be at least one. For an example of Features in a hierarchy, a Documentation sub-feature and a Samples sub-feature could be added beneath the main Help feature.

Several Features make up each Install Set, and each Feature can belong to one or more Install Sets. End Users select Features as an option from the Choose Install Set panel or console.

Files can be assigned directly to Features, or you can assign Components to Features. InstallAnywhere actually creates Components for you if you assign Files directly to Features.

End users can uninstall specific Features if desired.

Components

Components are the smallest piece of an installation handled by the installer. From a developers perspective they are the building blocks of applications or features. End Users never see or interact with Components.

There are many advantages to having fine-grained control over components. Common components may be shared between multiple installers, multiple versions of a product, or multiple products. For example, two products in a suite could have several shared components.

Components are versioned and each has a unique ID, so that a particular version of a component on a system can be searched for to see if the latest version has been installed at a particular location.

Several Components make up each Feature, and each Component can belong to one or more Feature. Components are made up of files and actions. Although you can assign files and actions to Components, you can also assign Files and actions directly to features and let InstallAnywhere automatically create the Components.

A file or action may belong to one component only. All installers must have at least one component, and can have as many as needed.

There are 3 types of components: standalone components, dependencies, and shared components. Each component type has different install/uninstall behavior.

- [Standalone Components](#)
- [Dependencies](#)
- [Shared Components](#)

Standalone Components

Standalone components are always installed. It does not matter if the component is already installed on the system. At uninstall time the application will uninstall the component. It is the default and most commonly used component.

Dependencies

Dependencies are components that are needed by the application that is being installed, but are not actually installed by the application's installer. This is useful if your application relies on other components like an application server or a database that may be installed by other applications. At uninstall time the application will uninstall the component even though it did not install it unless another installed application needs it.

The installer will search the product registry to check if a component with the component ID of the dependency specified is installed. You can also optionally specify a version number or key file location to make the search more specific. You can control when the installer searches for dependencies explicitly by using the Evaluate Dependencies action.

The search will set a series of InstallAnywhere variables based on its findings. The Matched Key File location variable will contain where the dependency is installed. You can use the Dependency State Variable to see if the search was successful. If the dependency check passes the Dependency State Variable will be an empty string. If the dependency check fails the Dependency State Variable will contain the Dependency Failed Message. For more information about the variables check the Standard Variables section in the User Guide.

For more information about dependencies check out the Sample Dependencies Template that comes with InstallAnywhere. It is sample project that helps illustrate how dependencies work, and what variables are set.

Shared Components

Shared components are components that can either be a standalone component or a dependency. If the component is not already installed the component will act as if it was a standalone component. If the component is already installed the component will act as if it was a dependency.

Installer Modes

InstallAnywhere installers can run in three different interactive modes:

- **Graphical User Interface (GUI)**—Graphical User Interface mode renders an installer with Wizard Panels and Dialog Boxes.
- **Console**—Console mode installers can perform remote installations over telnet or on systems without a graphical windowing environment. Also known as Command Line Interface, console mode is only available in installers created with InstallAnywhere Enterprise Edition.
- **Silent**—Silent installers do not interact with the user at all, and are suitable for distribution when all of the settings are already known or provided in a response file. Like console mode, silent mode is only available in installers created with InstallAnywhere Enterprise Edition.



Note • See [Silent Installers](#) for more information on using response files.

Graphical User Interface (GUI) Installers

Most aspects of an InstallAnywhere installer's user interface can be localized and customized. As a developer of an InstallAnywhere installer, you can alter text strings as well as graphics, such as the splash screen, installer screens, panel additions, background images, and billboards. Developers may even add their own animated GIF files to provide a multimedia experience.

GUI Localization

Nearly every text string in an InstallAnywhere project can be localized. Translations of the text of built-in InstallAnywhere screens and dialog boxes are already provided. The Enterprise Edition supports 31 different locales and the Standard Edition supports 9.

If developers want to further modify text strings by locale, the string files are output each time an installer is built, in a folder called <project_name>locales. This folder is in the same directory as the build output folder, and the files contained there are named by locale code. For example, the default English (locale code, en) locale file has the name custom_en.

These locale files contain the text string grouped by the name of the action to which it belongs. Developers may alter the text string and, upon the next build of the installer, the new localized text will be displayed with the action.



Note • For Enterprise edition users, InstallAnywhere also supports an alternative localization scheme based on the use of custom resource bundles. See [External Resource Bundles](#) for more information.

Look and Feel

InstallAnywhere provides many options for altering the look and feel of the installer.

Splash Screen

InstallAnywhere installers present a splash screen at the initial launch of the installer. This screen is displayed for a few seconds while the installer prepares the wizard, and allows end users to set the installer locale. The splash screen is an ideal introduction to your product, and is an opportunity to set the mood and image for your product. The splash screen will also appear on the HTML page generated for the InstallAnywhere Web Install Applet. It can be either a GIF, a PNG, or a JPEG file of any size, although the preferred size is 470x265 pixels.

GUI Panel Additions

The Additions to GUI Installer Panels option allows developers to display a list of steps or an image along the left side of the installer's panel.

Background Images

This Background image feature allows you to create a truly unique installer. The Background image is the graphical background for every panel in your installer. Background images are only supported in GUI installers.

Billboards

Billboards are graphics that the installer will display during the installation of files. Billboards generally convey a marketing message, a description of the product, or simply something fun for the end user to see as the file installation is occurring. Each billboard added will be displayed for an even amount of time, based on actions within the installation. If an installation has very few files, and many billboards, each billboard will only be displayed for a short time.



Note • When multiple billboards are specified, you can control how much time each billboard will be displayed while the installation is in progress by using the **Display billboards for every nn seconds** option. For more information, see [Billboard Timers](#)

Billboards can be GIF, PNG, or JPEG files. Billboards can even use animated GIF files, providing the end user with a richer media experience during their installation.

The maximum and preferred size for billboards is 380 x 270 pixels. If the preferred GUI mode is AWT and you have chosen not to display a side panel during the Install Progress step, the maximum size is 587 x 312 pixels.

Console Installers



Note • Only InstallAnywhere Enterprise Edition can create installers that operate in console mode. Console mode is unavailable in installers built with InstallAnywhere Standard Edition.

Console mode mimics the default GUI steps provided by InstallAnywhere, and uses standard input and output. The biggest advantage to console mode is that UNIX developers no longer need X-windows (X11) to run their installers.

Console mode allows for text to be output to the console line-by-line. It does not allow for any formatting, clearing of the screen, or positioning of the cursor.

```
+-----+
| CHOOSE ALIAS, LINK, SHORTCUT FOLDER      |
|                                           |
| Where would you like to create application shortcuts? |
|                                           |
| 1) In the Start Menu                      |
| 2) On the Desktop                         |
| 3) Don't create shortcuts                 |
+-----+
| Please make a selection [1, 2, or 3], and then   |
| press ENTER.                                |
+-----+
```

To trigger a console installer from the command line, type the following command:

```
installername -i console
```

Silent Installers



Note • Only InstallAnywhere Enterprise Edition can create installers that operate in silent mode. Silent mode is unavailable in installers built with InstallAnywhere Standard Edition.

Silent mode, which enables an installer to run without any user interaction, is fully supported on all UNIX platforms. A near-silent mode is possible on Windows and Mac OS X. InstallAnywhere and end-user-defined variables may be set through command-line parameters or a properties file.

To trigger a silent installer from the command line, type the following command:

```
installername -i silent
```

You may also call a properties file from the command line:

```
installername -f properties file
```

You may use the direct or the relative path to the properties file.



Note • InstallAnywhere variables may be incorporated in these values, and they will be resolved at install time.

You can also use response files and silent installers.

About Response Files and Silent Installers

InstallAnywhere's silent UI mode is useful for silently deploying to enterprise desktops. You can specify that you want to run a silent installation by using the `-i` command line switch (to set the installer interface mode) with the `silent` argument, such as:

```
install.exe -i silent
```

In silent mode, InstallAnywhere has no end user interaction, and runs either by using:

- the defaults provided by the developer, or
- a *response file* from which the installer retrieves the values for various InstallAnywhere variables used to control the install.

A response file contains a record of a specific installation session. InstallAnywhere creates the file when the installation is complete, storing the values of the applicable properties in the file.

Generating a Response File

You can choose to generate a response file by selecting an option in the Advanced Designer or by using the `-r` command line switch.

- **Selecting an option in the Advanced Designer**—You can specify that a response file is generated each time an installation is run by selecting the **Always Generate Response File** option on the **Project > Info** subtask of the Advanced Designer.

If you generate a response file by selecting this option, the response file is always named `installer.properties` or `[installername].properties` and it will be created in the same directory as the installer.

- **Using the -r command line switch**—You can also generate a response file when running an installation via command line by using the `-r` command line switch followed by the path and file name of the response file you want to generate, such as:

```
install.exe -r "/Users/MyName/myresponse.properties"
```

If you do not enter a path and file name for the response file, the file will be named `installer.properties` or `[installername].properties` and it will be created in the same directory as the installer.

Specifying the Response File to Use

When performing a silent installation, the installer automatically checks the directory in which it resides for a file named `installer.properties` or `[installername].properties`.

If you want to use a response file that has a different name or is in a different location, you can use the `-f` command line switch to set the name and location of a response file for the installer to use, such as:

```
myinstall.exe -f c:\tmp\installer.properties
```



Note • This path can be absolute or relative. Relative paths are relative to the location of the installer.

Contents of a Response File

Response files utilize a simple key=value format.

For example, if a console installer that you have previously built has effectively one real option, the installation directory, the properties file might look like this:

```
INSTALLER_UI=silent  
USER_INSTALL_DIR=C:\\Program Files\\OfficeSuite\\
```

`INSTALLER_UI` allows you to specify the installer mode in the properties file, negating the need to use the `-i silent` command-line switch.

If you would like to specify the install set you would like to install you can pass the `CHOSEN_INSTALL_SET` variable to a silent installer. You can also use the `CHOSEN_INSTALL_FEATURES_LIST` to specify the features you would like to install.



Note • To view a sample response file, see [Response Files](#).

Installer Types

InstallAnywhere offers Web and CD-ROM installers.

Web

Web Installers are packaged into a single executable file by platform, and are appropriate for distribution over the Web or via e-mail. They use a self-extractor to prepare the source files at the start of the installation, and therefore require more temporary disk space available.

CD

A CD-ROM Installer already has most of the files extracted and ready to install. There is still a self-extractor, but it only extracts the installer engine. Therefore, CD-ROM installers take up less temporary space and start up faster, but they are not appropriate for Web or electronic distribution. CD-ROM installers are also good for distribution on DVD, and can span multiple CDs or DVDs.

About Self-Extractors

The behavior of self-extractors varies by platform:

- **Non-Windows platforms**—On non-Windows platforms, the self extractor first attempts to extract the installer files to /tmp. If there is not sufficient space in /tmp, the self-extractor will then attempt to use the user's \$HOME directory.
- **Windows platforms**—On Windows, files are extracted to the following directory:

C:\Document and Settings\<user name>\Local Settings\Temp\<dirname>

where dirname is I<number> (a capital letter I followed by a random number). For example:

C:\Documents and Settings\JohnSmith\Local Settings\Temp\I1259850268

Java Virtual Machines

InstallAnywhere installers are Java based. They require a Java Virtual Machine (VM) to run.

Setup authors can bundle a virtual machine with the installer, thus eliminating the possibility that the installer will not find a suitable virtual machine for the product or for the installer. Setup authors can also exercise control over the VM search the installer's launcher performs and customize the search the installer does on behalf of the launchers it deploys.

Finally, developers can use the **Choose Java VM** panel in the **Pre-Install** task to allow their end users to choose from a set of VMs to use for the products they are installing.

The following topics explain the VM selection options available to setup authors, and the logic behind the VM resolution performed by InstallAnywhere launchers and installers.

Table 3-6 • Java Virtual Machines Topics

Topic	Description
About VM Selection	Provides an overview of the Java virtual machine selection processes in InstallAnywhere.
How the Installer's Launcher Selects a VM	Discusses how the installer's launcher determines which Java virtual machine will run the installer.
About the Installer's VM Search	Discusses how the installer searches for virtual machines suitable to run the launchers it deploys and applies the results of that search to the launchers.
About the Launcher's VM Selection Behavior at Run-Time	Explains how the options on the Create LaunchAnywhere for Java Application action customizer's Advanced Settings tab control the VM the launcher invokes.
About the Choose Java VM Panel	Describes the purpose the Choose Java VM panel and the options it provides to end users.
When Are VM Packs Installed?	Provides examples of cases in which a bundled VM will be installed on the target system.
About Java VM Selection Criteria	Describes the criteria and operators you can use to create simple or strict VM selection expressions for your project's installers and launchers. These expressions are used on the Project > JVM Settings task as well as for the LAX property, <code>lax.nl.valid.vm.list</code> .

About VM Selection

The Java virtual machine chosen to run an InstallAnywhere installer, or any InstallAnywhere-installed Java application, depends upon the settings in your InstallAnywhere project and the Java executables available on the end user's system.

The core of the VM selection process is performed by InstallAnywhere's LaunchAnywhere technology. LaunchAnywhere launchers are capable of performing their own VM search each time they run. However, it is also possible to

- **Bundle a VM with the installer** to be used by the installer and, optionally, to be used by the launchers your installer deploys. See [Bundling a VM Pack](#) and [Customizing Individual Launcher Settings](#).
- **Employ the results of the installer's VM search.** The installer's VM search is a customizable scan of the end users system. The results of this search can then be used to configure the launchers the installer deploys.
- **Include a Choose Java VM panel** that allows the end user to choose the bundled VM, choose local VMs that match the installer's VM search, start a VM search in a non-standard directory, or browse for a local Java executable.

Basic VM Selection by LaunchAnywhere Launchers

Without any significant changes to an InstallAnywhere project's default settings, Java virtual machine selection is managed exclusively by InstallAnywhere's LaunchAnywhere technology in a fairly straightforward manner. This is true for both the installer's launcher and the launchers for your Java applications provided that:

- No Java VM is bundled with the installer, AND
- No **Choose Java VM** panel is included in the installer.

A launcher first attempts to use the VM specified in its current VM property (`1ax.n1.current.vm`). If the VM specified there is either unavailable or invalid, the launcher searches the system for a valid VM.

When a LaunchAnywhere launcher must search for a VM, it scans common system locations for Java executables and uses the first one it finds that matches the criteria in its valid VM list (`1ax.n1.valid.vm.list`).



Note • The default value for all launcher's valid VM list only requires a VM version 1.4 or newer. For information on changing the valid VM list for the installer's launcher, see [Controlling the VM Used to Run the Installer](#). For information on changing the valid VM list for a launcher deployed by your installer, see [Customizing Individual Launcher Settings](#).

Impact of Bundled VMs, Installer's VM Search, and Choose Java VM Panel

Understanding which VM a launcher will use becomes more complicated when you choose to do one or more of the following:

- **Bundled VM**—Include a bundled VM with your installer. See [Controlling the VM Used to Run the Installer](#).
- **Custom VM search**—Customize the installer's VM search. See [Customizing the VM Search Settings for Your Launchers](#) and [Customizing the VM Search Paths and Patterns](#).
- **Choose Java VM panel**—Include a Choose Java VM panel in your installer. See [About the Choose Java VM Panel](#).

For the launchers the installer deploys, the VM selection process can be much more complex. The VM selected depends on:

- The presence of a bundled VM
- The results of the installer's VM search
- The presence of a Choose Java VM panel and that panel's configuration
- The launcher's Advanced Settings
- The Java VMs present on the end user's system

See [Controlling the VM Your Launchers Use](#) and [Using the Choose Java VM Panel](#) for detailed discussions of these conditions.

When Do VM Searches Take Place?

The installer's launcher, the installer itself, and the installed launchers are all capable of making VM selection decisions and performing VM searches—some during the install process and some after.

- **Controlling the VM used to run the installer**—When a user runs an InstallAnywhere installer, the installer's launcher does a "bootstrap" search of the system for a VM that meets the criteria expressed in your project's **Installer Valid VM List** (which is set in the Advanced Designer in the **Valid VM list** field on the **Installer Settings** tab of the **Project > JVM Settings** task). See [Controlling the VM Used to Run the Installer](#) for more information.
- **Controlling the VM launchers use**—Once the installer is running, it performs a search for a VM to use with any LaunchAnywhere launchers it will install. The criteria the installer uses for this search depends on VM search settings you provide on the **Search Panel Settings** tab of the **Project > JVM Settings** task. See [Controlling the VM Your Launchers Use](#) for details.

These settings can be further customized for Windows and Unix installers. See [Customizing the VM Search Paths and Patterns](#).

- **Customizing launcher settings**—LaunchAnywhere launchers are also capable of performing an independent VM search when they are executed. LaunchAnywhere performs this search when it does not receive a VM setting from the installer or when specifically set to perform its own search. For information on how to require LaunchAnywhere to perform its own VM search, see [Customizing Individual Launcher Settings](#).

Where Do VM Searches Take Place?

The common locations examined during a VM search performed by an InstallAnywhere launcher or a VM search performed by an installer are quite similar.

Table 3-7 • Location of VM Searches

Platform	VM Search Location
Windows	VM searches are done in directories listed in the PATH environment variable as well as the following registry keys: SOFTWARE\JavaSoft\Java Development Kit SOFTWARE\JavaSoft\Java Runtime Environment SOFTWARE\IBM\Java2 Runtime Environment SOFTWARE\IBM\Java Development Kit
UNIX	Launchers search only the PATH environment variable while the installer VM search includes these paths as well: /bin /usr/bin /usr/local/bin /opt /opt/gnu/bin /usr/gnu/bin

How the Installer's Launcher Selects a VM

InstallAnywhere installers are Java-based and require a Java virtual Machine (VM) in order to run. The VM the installer uses depends on a couple of factors:

- The presence of a bundled VM.
- The criteria in the Installer Valid VM List.

The installer's launcher searches the end user's machine for a VM that matches the "bootstrap" VM search criteria set for the installer's launcher (**Valid VM list** on the **Installer Settings** tab of the **Project > JVM Settings** task). If a VM has been bundled with the installer, the installer uses the bundled VM. If no VM is bundled with the installer, the installer uses the first VM on the target system that matches the "bootstrap" search criteria. If no VM matching the search criteria is found, the installer reports the error and exits.



Note • To bundle a virtual machine, it must be packaged as a VM pack. Free VM packs are available on the InstallAnywhere Web site. See [Working with VM Packs](#) for more information.

About the Installer's VM Search

After an InstallAnywhere installer is launched, it searches for VMs on the end user's system. The intent of this search is to find a VM suitable for all the LaunchAnywhere launchers your installer deploys. The criteria used for this search can be one of the following:

- **Criteria used in bootstrap phase**—The criteria the installer's launcher uses in the "bootstrap" phase (the **Valid VM list** field on the **Installer Settings** tab of the **Project > JVM Settings** task) can be used.
- **Criteria in lax.nl.valid.vm.list property**—The criteria expressed in the `lax.nl.valid.vm.list` property of the launchers in your project can be used. If more than one launcher exists in your project, the installer searches for a VM that meets all the VM criteria of these launchers.
- **Custom criteria**—A custom criteria set specifically provided for the project's launchers can be used. You specify this criteria using the **Use a specific valid VM list** field on the **General** subtab of the **Search Panel Settings** tab of the **Project > JVM Settings** task.



Note • *The installer applies the result of this search to your launchers only when the installer includes a bundled VM or a **Choose Java VM** panel. If neither is present, the search results are still available via custom code but are not applied to your project's launchers.*

On Windows and Unix target systems, InstallAnywhere also supports further refinement of the VM search paths and search patterns (Java Executable patterns). These describe the locations in which the installer searches for valid VMs and the patterns it uses to recognize Java executables on those platforms. For more information see [Customizing the VM Search Paths and Patterns](#).

Every LaunchAnywhere launcher is capable of performing an independent VM search. For details on configuring a launcher to always perform an independent search and setting criteria for that search, see [Customizing Individual Launcher Settings](#).

About the Launcher's VM Selection Behavior at Run-Time

Individual InstallAnywhere launchers can be customized to dictate how their VMs are set (or not set) by the installer. Typically, a project's launchers acquire their VM settings from either the installer's VM search (using the criteria set specified on the **General** subtab of the **Search Panel Settings** tab of the **Project > JVM Settings** subtask) or, when provided, the choice the end user makes in the **Choose Java VM** panel. However, you can configure your InstallAnywhere product such that an individual LaunchAnywhere launcher acquires its VM setting in one of a variety of ways. And depending on your project settings and the target system environment, the launcher can:

- Use the VM bundled with the installer.
- Use the VM found by the installer's VM search.
- Use the VM found by the installer's launcher during the "bootstrap" search (the same VM that was used to run the installer when the installer has no bundled VM).
- Use the VM found by an independent LaunchAnywhere VM search.

- Use the VM identified by an end user's search in a non-standard path.
- Use the VM selected by an end user after browsing for a specific Java executable.



Note • The last two options require the inclusion of the **Choose Java VM** panel action in the **Pre-Install** task. For more information on the use of the **Choose Java VM** panel, see [Using the Choose Java VM Panel](#).

Advanced Settings of the Create LaunchAnywhere for Java Application Action

By default, LaunchAnywhere launchers follow the behavior indicated by the first of four Advanced Settings options. You can find the **Advanced Settings** tab on the **Create LaunchAnywhere for Java Application** action customizer. The **Advanced Settings** tab shows the following settings:

- VM Selected by the Installer or by the End User Via Choose VM Panel
- VM Used by the Installer
- First VM Found in the System Matching the VM Search Settings Defined on the Installer Settings Tab of Project > JVM Settings
- No Specific VM

VM Selected by the Installer or by the End User Via Choose VM Panel

This option is the default LaunchAnywhere behavior that provides consistency with prior InstallAnywhere versions. If you choose this option on the **Advanced Settings** tab, the result depends upon whether a **Choose Java VM Panel** was included in the project:

Table 3-8 • Result of Choosing the “VM Selected by the Installer or by the End User”

Scenario	Result
Choose Java VM Panel is NOT Included	If a Choose Java VM panel is NOT included in the project, one of the following will occur: <ul style="list-style-type: none"> • If no VM is bundled with the installer, then the launcher uses the same VM that launched the installer. • If the bundled VM is not installed, the installer does not set a VM for the launcher. In this case, the launcher performs its own VM search when it is invoked (as necessary) and uses the criteria value in its <code>tax.n1.valid.vm.list</code> property to qualify suitable VMs. • If the bundled VM is configured to be installed, the installer sets the launcher to use the bundled VM. • If the bundled VM is configured to be installed only when a compatible VM is not found on the system and a compatible VM is found, the installer configures the launcher to use the first VM found by the search.

Table 3-8 • Result of Choosing the “VM Selected by the Installer or by the End User”

Scenario	Result
Choose Java VM Panel Is Included	<p>If a Choose Java VM panel is included in the project, the installer configures the launcher to use the VM selected by the user.</p> <p>Depending on the settings in the Choose Java VM customizer and whether or not a bundled VM is installed, the user may:</p> <ul style="list-style-type: none"> • Choose the bundled JMV, or • Choose the VM that matched the installer's VM search, or • Browse to locate a specific VM. <p>See Using the Choose Java VM Panel for details.</p>

VM Used by the Installer

This option configures the launcher to use the VM used to run the installer. Hence the VM the launcher will use depends on whether or not a VM is bundled with the installer, whether the bundled VM is installed, and the results of the search that locates a VM to launch the installer (bootstrap):

- **If no VM is bundled with the installer**, the installer configures the launcher to use the first VM the installer found using the criteria in the **Valid VM list** field on the **Installer Settings** tab of the **Project > JVM Settings** task. This is the same VM used to run the installer.
- **If the bundled VM is installed**, the installer configures the launcher to use the bundled VM.



Note • There are two cases in which this setting, **VM Used by the Installer**, combined with certain settings on the **General** subtab of the **Search Panel Settings** tab of the **Project > JVM Settings** task result in a build error:

1. The bundled VM is not configured to be installed.
2. The bundled VM is configured to be installed only when a compatible VM is not found in the system.

First VM Found in the System Matching the VM Search Settings Defined on the Installer Settings Tab of Project > JVM Settings

This option configures the launcher to use the results of the installer's VM search. The installer uses the **VM Search Settings** defined on the **General** subtab of the **Search Panel Settings** tab of the **Project > JVM Settings** task.

- **If a compatible VM is found on the system**, the installer configures the launcher to use the first VM found in the search.
- **If no compatible VM is found on the system**, the launcher receives no VM configuration. As a result, the launcher performs its own VM search when it is invoked.



Note • For details on how the VM Search Settings are applied to find a VM, see [Customizing the VM Search Settings for Your Launchers](#).

No Specific VM

This option provides no VM configuration for the launcher from the installer at run time. When you choose this option, the launcher performs its own VM search when it is invoked and uses its `lax.nl.valid.vm.list` property to establish valid VM criteria.



Note • You can customize the default value (1.4+) of a launcher's `lax.nl.valid.vm.list` property. To do so, click the **Edit Properties** button on the **General Settings** tab of the **Create LaunchAnywhere for Java Application** customizer. This opens the **LaunchAnywhere Properties** dialog box. Locate `lax.nl.valid.vm.list`, double-click its value, and edit the text. (For a detailed discussion of VM selection criteria settings, see [About Java VM Selection Criteria](#).) Click **OK** to close the **LaunchAnywhere Properties** dialog box.

About the Choose Java VM Panel

In addition to the VM selection decisions the installer and its launchers make, you can provide your end users with the option to choose a VM they want your software to use by adding the **Choose Java VM** panel to your project. The **Choose Java VM** panel supports the following range of choices for your end users:

- Choose a valid VM found by the installer's VM search.
- Choose the bundled VM (only shown when a bundled VM is installed).
- Search for a VM in non-standard locations.
- Browse for a VM on the local system.

The **Choose Java VM** panel enables all these options by default. The **Choose Java VM** panel always searches the local system for VMs and presents those VMs that match the selection criteria to the end user. The options for choosing the bundled VM, searching non-standard locations, and browsing for a specific Java executable are represented by checkboxes in the **Choose Java VM** customizer.

When Are VM Packs Installed?

Whether or not a VM Pack is installed along with your installer depends on several conditions.

VM Pack Will Be Installed

A VM will be installed under the following conditions:

- You bundle a VM with the installer and set the **Install Bundled/Downloaded Virtual Machine** option without setting the **Only When Installing a LaunchAnywhere** and **Only When a Compatible VM Is Not Found in the System** options.
- You bundle a VM with the installer, and set the **Install Bundled/Downloaded Virtual Machine** and **Only When Installing a LaunchAnywhere** options, and LaunchAnywhere executables are installed on the target system.

- You bundle a VM with the installer, and set the **Install Bundled/Downloaded Virtual Machine** and **Only When a Compatible VM Is Not Found in the System** options, and no compatible VM is found on the target system.
- You bundle a VM with the installer, set the **Install Bundled/Downloaded Virtual Machine** option, and include a Choose Java VM panel in which the end user chooses the bundled VM at install time.

VM Pack Will Not Be Installed

A VM pack will not be installed under the following conditions:

- The installer is built without a bundled VM.
- You bundle a VM with the installer, but you deselect the **Install Bundled/Downloaded Virtual Machine** option on the **Installer Settings** tab of the **Project > JVM Settings** subtask.
- You bundle a VM with the installer, but you set the **Only When a Compatible VM Is Not Found in the System** option and a compatible VM is found during the installer's VM search.
- You bundle a VM with the installer, but you set the **Only When Installing a LaunchAnywhere** option and LaunchAnywhere executables were not installed (including the Uninstaller's LaunchAnywhere executable file).



Note • For more information about VM packs, see [Working with VM Packs](#).

About Java VM Selection Criteria

InstallAnywhere supports strict virtual machine selection. This feature allows you to specify the vendor, type, and version of the Java virtual machine required by your installer or your installed product. If no VM listed in the Valid VM List is available, the installer fails to run and LaunchAnywhere reports a **Could not find valid Java virtual machine to load** error.

Java VM selection criteria can be used to specify valid VMs on the **Project > JVM Settings** task and for LaunchAnywhere (`lax.n1.valid.vm.list` property).

The values for these properties can be any space-delimited combination of the following general operators.

Table 3-9 • Java VM Selection Criteria

Property	Description
VM Vendor	Vendor values specify the creator or publisher of the virtual machine. Some common Java virtual machine vendors include: <ul style="list-style-type: none"> • IBM • SUN • HP (only available on HP-UX systems) • APPLE (only available on Mac systems)

Table 3-9 • Java VM Selection Criteria

Property	Description
VM Type	<p>VM type values include:</p> <ul style="list-style-type: none"> • ALL (any VM) • JDK (any JDK) • JRE (any JRE)  <p>Note • The optional JDK or JRE specifies which type of VM is valid.</p>
VM Version	<p>Version values can be specific versions, such as 1.4.2_02 or partial versions that include a + or * wildcard character.</p> <p>The version number can have varying degrees of precision; however, we recommend having at least the major and minor version numbers specified:</p> <ul style="list-style-type: none"> • JRE_1.4+ selects any JRE type VM of version 1.4.0_0 or greater. • JDK_1.4.2* selects any JDK type VM of the 1.4.2 series. <p>The + or * operator specifies a version range:</p> <ul style="list-style-type: none"> • The + operator means “at least this version.” • The * operator means “of this version.” <p>When you use these operators, any unspecified version part is assumed to be zero (specifying 1.4 is interpreted as 1.4.0_0). If you do not specify * or +, only versions which exactly match the specified version are valid (1.4 does not validate for 1.4.1_07 VMs).</p>
Strict VM Selection	<p>Alternately, you can use a strict VM expression, such as JRE_1.5.1_03 or JDK 1.4.2_02, by joining a vendor (IBM, SUN, and others), type (ALL, JDK, or JRE), and version number (1.4.2_02, 1.5*, and so on) with an underscore character.</p> <ul style="list-style-type: none"> • IBM_ALL_1.6* allows the installer or application to run only against a version 1.6 VM from IBM. • JRE_1.5.1_03 allows the installer or application to run only against the JRE 1.5.1_03. • JDK_1.4.2_02 allows the installer or application to run only against a JDK 1.4.2_02.  <p>Note • If more than one of these expressions is present, consider them ORed together. In other words, a VM is valid if it matches any of the given expressions.</p>

LaunchAnywhere

LaunchAnywhere is a Java application launcher technology. A LaunchAnywhere Executable is an executable file that is used to launch a Java application on any LaunchAnywhere-compatible platform (all Windows, UNIX platforms, and Mac OS X).

LaunchAnywhere enables end users to launch a Java application by double-clicking on an icon (Windows or Mac OS X) or by typing a single command (UNIX).

A LaunchAnywhere Java application launcher automatically locates an appropriate Java Virtual Machine (JVM), either bundled with the application or already installed on the system, and also configures the Java application environment by setting the classpath, redirecting standard out and standard error, passing in system properties, environment variables, and command-line parameters, and many other options.

LaunchAnywhere hides the console window by default for GUI applications, but it can be set to display the console for text-based applications. All LaunchAnywhere settings are configured within InstallAnywhere and automatically set when the installer installs the application.

LaunchAnywhere accepts a command-line parameter, LAX_VM, to force selection of a certain Java Virtual Machine. To use this property:

```
<LaunchAnywhere> LAX_VM <fully qualified path to java executable>
```

The launcher looks at a configuration file <MyLauncherName>.lax to determine how the launcher runs. This LAX file is created during the installation, and is placed in the same location as the launcher.

Localization

InstallAnywhere provides many tools and resources to help you deliver installers in the language your users are most comfortable using. InstallAnywhere Enterprise edition supports localizations to 31 different languages. InstallAnywhere Standard edition supports localizations to 9 different languages.

The following topics describe some key localization concepts for InstallAnywhere users.

Table 3-10 • Localization

Topics	Description
Localization Basics	Describes the location and content of the locale files InstallAnywhere generates for the locales you enable in your project.
Dynamic and Static Text	Discusses the differences between dynamic and static text.
Best Practices for Localizing	Provides a set of recommendations for efficient, effective localization.
External Resource Bundles	Describes the purpose and content of external resource bundles—an alternative to the standard locale files InstallAnywhere generates.

Localization Basics

When an installer project is first built, a folder called <projectname>locales is created in the same directory as the project file. For each locale selected on the **Locales** subtab of the **Build > Build Configurations** tab in the Advanced Designer there will be a file in this <projectname>locales folder. The locale files are generated as custom_<localecode>, so for English which has a locale code of en, the name of the locale file will be custom_en. These files contain keys and values for all of the dynamic strings in the project. The keys are generated by the name of the action, with a unique value to represent the unique instance of the action, and an additional parameter to signify which dynamic value of the action is being referenced. For example:

```
InstallSet.9733839b90f6.installSetName=Typical  
InstallSet.9733839b90f6.description= The most common application features will be installed.
```

This option is recommended for most users. The ProjectLocalizationInfo.txt file contains the mapping between the actions in the project and their keys in the locale files. Review the ProjectLocalizationInfo.txt file for any questions as to which action the key refers to.

Dynamic and Static Text

Text which can be entered into the InstallAnywhere Advanced Designer is dynamic text. Dynamic text means that the developer can change its value. Text that cannot be edited by the Advanced Designer is referred to as static text. Standard items such as file choosers as well as text for actions and panels the developer is unlikely to wish to change have static text. Dynamic text is written to the locale files (located next to the project file). Translations of static text can be found in

```
<InstallAnywhere>\resource\i18nresources
```

While developers are unlikely to need to change the static text, InstallAnywhere provides the developer with the option to change the static text.

Almost all dynamic text has default values in the Advanced Designer. These dynamic values also have default translations. Every string that is modified in the Advanced Designer needs to be localized by the installer developer if they want the translation to match. There are also actions that do not have defaults such as custom code and **Get User Input** panels. It is the developer's job to localize these strings as well.



Note • *InstallAnywhere writes all changes of dynamic values into the locale file of the same language the Advanced Designer is running in. When using the Japanese language variation dynamic changes are written into the custom_ja file. Using the Advanced Designer is the correct way to modify this locale file. Changes to the locale files made outside of the Advanced Designer may be overwritten.*

Best Practices for Localizing

When localizing your installers, observe the following best practices:

- **Complete the installer design before translating the locale files.** Changes to the installer design can affect the layout and contents of the locale files. If these files change, it may require costly re-translation work.
- **Stick with the default text whenever possible.** All default text is already translated, saving the team time and effort.
- **Test the installers on systems running in the foreign locale.** This helps identify any errors where the proper strings are not translated in the locale files.
- **Make sure every resource referenced in the locale files is included in the installer.** This is especially true for license agreements, readme files, release notes, and other commonly translated documents.

External Resource Bundles



Edition • External resource bundles are available only in InstallAnywhere Enterprise edition.

As an alternative to the built-in, dynamic locale files, InstallAnywhere allows you to create custom resource bundles. Because these external resource bundles use custom keys and values, they are well suited to values that remain consistent across products and versions.

These resources are created and referenced during installer design, bundled at build time, and resolved at install time.



Note • In the Advanced Designer, when you replace text elements with references to a key in an external resource bundle, InstallAnywhere uses the locale files from the external resource bundle for all localizations of that element.

The installer resolves the localized value based up on the value of \$INSTALLER_LOCALE\$. So, for example, when a user chooses German on your installer's splash screen, InstallAnywhere draws the value of the key your installer references from the German locale file in subsequent panels.

- [Naming Conventions](#)
- [File Format](#)
- [External Resource Bundle Support for Merge Modules in Install and Uninstall Phases](#)

Naming Conventions

The properties files in the external resource bundle should use the following syntax:

<bundle>_<language_code>.properties

- <bundle> can be any string of legal file name characters but must be consistent between the various locale properties files in the same external resource bundle.
- <language_code> must be the language code for one of InstallAnywhere's supported locales.

For example: custom_en.properties, custom_ja.properties

File Format

The properties files in the external resource bundle are standard Java resource bundles. The properties files must be text files (UTF-8) with a list of key/value pairs for each localized string.

InstallSet.installSetName=Custom

InstallSet.description=Choose this option to customize the features to be installed.



Note • To use external resource bundles, you must create a properties file for each locale you plan to support. Installers built for locales without a properties file default to the English values for each localized string that references the external resource bundle.

External Resource Bundle Support for Merge Modules in Install and Uninstall Phases

Starting with InstallAnywhere 2011, subinstallers (Merge Modules) are able to access the External Resource Bundle of their parent installer during the installation and uninstallation phases. This will also be applicable to one-level nested merge modules and to multiple-merge modules which get installed with a single parent.

Magic Folders

InstallAnywhere uses Magic Folders to define installation locations. These Magic Folders are a way of keeping track of installation locations. InstallAnywhere can install to any Magic Folder or sub-folder of a Magic Folder. Magic Folders represent a specific location, such as the user selected installation folder, the desktop, or the location for library files. At install time, the installer determines which operating system it is running on, and sets the Magic Folders to the correct absolute paths. Many Magic Folders are platform specific and many are predefined by InstallAnywhere. You can install to nearly standard location on any supported platform. InstallAnywhere also provides user controlled Magic Folders which can be set as the developer needs them.



Note • In previous releases, you were limited to a maximum of 10 user-defined Magic Folders. Starting with InstallAnywhere 2011, the number of permitted user-defined Magic Folders has increased to 25.



Note • For information on how Magic Folders are associated with variables, see [Magic Folders and Variables](#).



Note • Not all Magic Folders are available in every edition.

Merge Modules

Merge Modules are essentially installer sub-projects that can be created independently of one another and later merged together. Like an installer, a Merge Module is a reusable collection of installation functionality, complete with features, components, panels, actions, and files.

However, a Merge Module cannot be installed on its own; instead, developers use Merge Modules when they want to include the functionality of one installer within another installer.

- [Benefits of Using Merge Modules](#)
- [Using Third Party Merge Modules](#)
- [Methods to Add Merge Modules to an Existing Installer](#)
- [Integrating Merge Modules Into Projects](#)
- [Size Limitations of Merge Modules](#)



Note • For instructions on creating Merge Modules, see [Creating Merge Modules](#).

Benefits of Using Merge Modules

Merge Modules provide many benefits and provide solutions to complex installation requirements. For instance:

Table 3-11 • Benefits of Using Merge Modules

Benefit	Description
Can Use to Create a Suite Installer	Combine several Merge Modules from different products to create a “Suite Installer.”
Can Be Used by Independent Development Teams	Independent development teams in different locations can create Merge Modules for different software components. A release engineer can combine those Merge Modules into a single product installer.
Self-Contained Units Can Be Reused in Future Installer Projects	Create self-contained units of installer functionality for reuse in future installer projects. For instance, if the same software component needs to be in several different installers, build it into a Merge Module and make it available for all of the installer developers.

Table 3-11 • Benefits of Using Merge Modules

Benefit	Description
Enables You to Store Common Installer Functionality	Save common installer functionality, such as License Agreement panels and Custom User Input panels, into Merge Modules to simplify future installer project creation.
Can Combine With Third Party Software Packages	Combine Merge Modules from 3rd party software packages to build complex software “Solutions,” without having to figure out how to install each individual package.

Using Third Party Merge Modules

InstallAnywhere provides many third-party Merge Modules on its Web site.

- **Use as a template**—You can use a Merge Module as the starting point for a new installer project. These Merge Modules are referred to as Templates, and are covered in another section.
- **Combine with installer projects**—Any installer project can be built into a Merge Module. And any Merge Module can be used within any other installer project.
- **Create during installer build process**—Merge Modules are created as an option through the installer build process. Since a Merge Module contains all of the resources for a project, it is just like building an installer. They can be built automatically when the installer is built, or they can be explicitly built from the Advanced Designer (check the **Build Merge Module** option on the **Build** task, under the **Distribution** tab) or from the command line (use the +merge option).

Methods to Add Merge Modules to an Existing Installer

Merge Modules can be merged into an existing installer in one of three ways:

Table 3-12 • Methods to Add Merge Module to an Existing Installer

Method	Description
Import Dynamic Merge Module	<p>In the Organization > Modules task, click Import Dynamic Merge Module to merge a Merge Module into the current installer.</p> <ul style="list-style-type: none"> • All of the Merge Module’s files, actions, and panels (optionally) will be combined into current project. • The actions will appear as in an Action Group in Pre-Install and Post-install. <p>Dynamic Merge Modules are recommended if you have Merge Modules whose contents constantly change. A parent project will automatically refresh dynamic Merge Modules when the parent project is loaded and built.</p>

Table 3-12 • Methods to Add Merge Module to an Existing Installer

Method	Description
Import Merge Module	<p>In the Organization > Modules task, click Import Merge Module to merge a static Merge Module into the current installer.</p> <ul style="list-style-type: none"> • All of the Merge Module's features, components, files, actions, and panels (optionally) will be combined into the current project, allowing developers to further customize any settings. • Static Merge Modules are recommended if you want import features and components and if you want more freedom in the place actions. • Changes made to a static Merge Module will not be noted in the parent project.
Installed as Self-Contained Sub-Installer Units	<p>Merge Modules can also be installed as self-contained sub-installer units, without merging them into the current project.</p> <p>This is useful if developers do not know what will be in a Merge Module, or they will not be modifying any settings. Merge Modules added in this manner are run as silent sub-installers.</p>

Integrating Merge Modules Into Projects

Merge modules can be integrated with a project in one of two ways:

Table 3-13 • Methods to Integrate Merge Modules Into Projects

Method	Description
Bundle Merge Module at Build Time	Use the Install Merge Module action and select Bundle Merge Module at Build Time , if the Merge Module is available when ready to build the installer. These Merge Modules will be included in the actual generated installer.
Locate Merge Module at Install Time	Use the Install Merge Module action and select Locate Merge Module at Install Time to have the installer install a Merge Module that is available at install time, but external to the installer. The Merge Module can be either on the end user's system or stored on a CD. If the location is a folder that contains several Merge Modules, they will all be installed.



Note • Starting with InstallAnywhere 2011, subinstallers (Merge Modules) are able to access the External Resource Bundle of their parent installer during uninstallation.

Size Limitations of Merge Modules

The size at which a Merge Module build will fail depends upon the amount of memory allocated to the build process. If the amount of memory allocated is very low, even smaller Merge Modules may fail.

You can specify the size of memory allocation using the `lax.nl.java.option.java.heap.size.initial` and `lax.nl.java.option.java.heap.size.max` properties in the `build.lax` file. They correspond to `-Xms` and `-Xmx` flags for the Java VM.

Table 3-14 • Memory Allocation of Build Process

Property	Code
Heap Size Initial	<pre># LAX.NL.JAVA.OPTION.JAVA.HEAP.SIZE.INITIAL # ----- # the initial heap size for the Java VM lax.nl.java.option.java.heap.size.initial=25165824</pre>
Heap Size Maximum	<pre># LAX.NL.JAVA.OPTION.JAVA.HEAP.SIZE.MAX # ----- # the maximum heap size for the Java VM lax.nl.java.option.java.heap.size.max=134217728</pre>

Project File

InstallAnywhere stores every project in its own XML file. These XML-based project files can be checked in and out of source control systems, and can be modified with text and XML editors. For added flexibility, project files may also be modified using XSL transformations, providing the ability to modify referenced file paths or other attributes. Several XML and XSL tools to work on the XML project file can be found in the InstallAnywhere application folder inside XML Project File Tools.

Rules

InstallAnywhere Rules can be applied to any action within the InstallAnywhere installer, as well as to organizational units such as Install Sets, Features, and Components.

InstallAnywhere uses variable based Boolean rules to control most aspects of installer behavior. The Rules logic allows developers to create simple and complex logic systems that determine what actions will occur. The rules can be structured based on end user input, or on conditions determined by the installer.

Source Paths

You can use Source Paths to reference file resources using variable paths instead of absolute paths. This allows you to share a project file with other team members, even when the file resources are located at different paths on their development systems.

Using Source Paths also enables you to use the same project file on different types of operating systems, such as between UNIX and Windows.

Source Path Substitution

Source Paths will automatically be substituted for the most complete path possible. For example, suppose you have a project with the following defined Source Paths:

```
$SOURCEPATH1$ = D:\Sources\SampleApp\3000  
$SOURCEPATH2$ = D:\Sources
```

When you add a file such as D:\Sources\SampleApp\3000\readme.txt to that project, because \$SOURCEPATH1\$ has the most complete path match available, this file will be referenced by:

```
$SOURCEPATH1$/readme.txt
```

If a team member opens this project and the Source Path is not defined, InstallAnywhere shows a dialog box that requests the location for the Source Path.

Predefined Source Paths

InstallAnywhere provides some predefined Source Paths that exist in any project, which cannot be changed or edited:

Table 3-15 • Predefined Source Paths

Source Path	Description
\$IA_HOME\$	Location on the system where InstallAnywhere is running. A common location is: C:\Program Files\InstallAnywhere
\$IA_PROJECT\$	Location on the system where the InstallAnywhere project is located.
\$USER_HOME\$	The User Home folder.

SpeedFolders

Using SpeedFolders will dramatically increase installation speed and memory efficiency. Similar to folders that are added to a project using the Add Files method, SpeedFolders represent a container of other folders and files that are to be installed on the destination computer. SpeedFolders are a pointer to a particular folder, as opposed to a traditional folder, in which every item inside of it is a separate action. However, unlike normal folders, SpeedFolders and the contents they represent are treated as a single action, rather than each item representing an individual item. This combining of items lowers memory requirements and speeds up the installation.

SpeedFolders are excellent for use in an automated build environment. The contents of a SpeedFolder are determined at build time. At the time the installer is built, all of the contents of the folder on the build system (excepting items that have been marked to filter out) are added to the installer recursively. SpeedFolders are used to specify that a folder and all of its contents and sub-folders on the development system are to be automatically updated and included in the installer at the time the project is built. Standard folders (non-SpeedFolders) require developers to add or remove any files that are present or absent since the last installer build, or an error will occur.

SpeedFolders have filters that allow inclusion or exclusion of files or folders that meet particular naming criteria.



Note • Individual files or folders in a SpeedFolder cannot be assigned to different components, nor can SpeedFolders be converted into traditional directories, or traditional directories into SpeedFolders. To convert one type to the other, you must delete the one folder and replace with the other type of folder.

Templates

A template is the starting point for every new installer project. A template can be a simple empty project, or it can contain everything a regular project would contain, such as license agreements, custom graphics and billboards, and even files.

A template is simply a Merge Module that has been placed within the `iatemplates` folder inside the InstallAnywhere installation folder. When you create a new project, you have the option of starting from a Template. When you start from a Template, a copy of the template is created and saved.

Templates are great for large installer teams, where you want everyone to have a consistent starting point, or for starting a new project based upon an older one.

Uninstaller

InstallAnywhere automatically creates an uninstaller for the project which can be removed manually. The InstallAnywhere uninstaller removes all files and actions that occur during the Install task of the installation. Actions added in other phases of the installation cannot be removed using the uninstaller, and should be accounted for in the install phase.

The uninstaller is much like the installer. It is a collection of panels, consoles and actions. All Pre-Uninstall panels, actions and consoles run first. Then the uninstall functionality of actions in the Install task are called. Lastly, Post-Uninstall actions are run.

Table 3-16 • Uninstaller Topics

Topics	Description
About Uninstallers and Custom Code	Discusses how custom code can be created to extend the functionality of the uninstaller.
About Uninstallers and Variables	Briefly describes how variables are written out at the end of the install process.
Feature Uninstall	Discusses feature-level uninstallation behavior.
Uninstaller for Merge Modules and Multiple Products	Describes some more complex uninstaller behavior as it pertains to Merge Modules and multiple products.
Similarities Between Pre and Post Install and Uninstall Tasks	Explains the similarities between basic Install and Uninstall tasks.

About Uninstallers and Custom Code

As with the installer custom code can be created to extend the functionality of the uninstaller. There are several ways to include Custom Code in the uninstaller. The first is by including Custom Code Actions, Panels or Consoles in the Pre-Uninstall or Post-Uninstall task. For Custom Code Actions, the `uninstall()` method will be called for these actions. The second is by placing a Custom Code Action in the Install task. In this case the `install()` method of the action will be called at install time and the `uninstall()` method of the same action will be called at uninstall time.

About Uninstallers and Variables

The majority of InstallAnywhere variables are written to the uninstaller at the end of the installer. These variables are available to help custom uninstaller actions by providing the state of the installer at the end of its execution.

Feature Uninstall

Each installer project has one uninstaller. All features are registered with the uninstaller through a local registry. If the Choose Feature panel is included in the uninstaller, the user will be offered the option to uninstall only certain features.

A feature level uninstall, enables end users to choose specific features to uninstall. If an end user opts to uninstall one feature that has a shared component with a feature they were not planning to uninstall, the uninstaller recognizes this conflict and does not uninstall the shared component.

Uninstaller for Merge Modules and Multiple Products

When developing an installer that installs multiple products (or uses Merge Modules), it is important to consider how the uninstallers for these products will integrate. Each uninstaller is tied to a certain product. It does this through its Product ID (found in the **Project > Description** task).

You have three options when developing an uninstaller for a project that installs multiple products (or uses Merge Modules):

- Separate Uninstallers
- One Uninstaller for Multiple Products (or Merge Modules)
- Master Uninstaller Which Executes Sub-Uninstallers
- Uninstalling Merge Modules When Main Project is Uninstalled

Separate Uninstallers

You can decide to have a separate uninstaller for each product. This is the easiest option.

If you choose this option, make sure that every product has a unique product ID, and that the uninstallers install to unique locations.

One Uninstaller for Multiple Products (or Merge Modules)

You can choose to have one uninstaller that includes uninstallation logic for all of products. InstallAnywhere has the ability to merge the uninstallation logic from separate uninstallers. In order to create one merged uninstaller function for a group of separate products, follow these guidelines:

- Every product must have the same Product ID.
- Each separate product should then be “demoted” to a feature of the master product.
- The uninstallers for these separate projects must also share the same uninstaller name and uninstaller location.
- Every product that installs features will register its features with the uninstaller.

The following is an example of creating one uninstaller for multiple products:

Table 3-17 • Example of Creating One Uninstaller for Multiple Products

Item	Description
Product	Acme Office Suite (ID: 97338341-1ec9-11b2-90e2-a43171489d33)
Installer 1	Acme Word Processor and Acme Spreadsheet <ul style="list-style-type: none">• Product ID: 97338341-1ec9-11b2-90e2-a43171489d33• Features: Acme Word Processor and Acme Spreadsheet
Installer 2	Acme Slide Show <ul style="list-style-type: none">• Product ID: 97338341-1ec9-11b2-90e2-a43171489d33• Features: Acme Slide Show
End Result	Acme Office Suite uninstaller for Acme Word Processor, Spreadsheet, and Slide Show.

Master Uninstaller Which Executes Sub-Uninstallers

You can choose to produce a master uninstaller that executes the other sub-uninstallers. This option is most similar to concept of Merge Modules used in the installer.

Developers use the **Execute Uninstaller** action to cause the master uninstaller to execute other uninstallers during uninstallation. The advantage of this technique is that it allows developers to separate their uninstallation logic. Developers who use this option will find the InstallAnywhere Variables \$NEVER_UNINSTALLS_VM\$ and \$REGISTER_UNINSTALLER_WINDOWS\$ useful.

Uninstalling Merge Modules When Main Project is Uninstalled

For Dynamic Merge Modules, you can specify that you want to implement a single uninstaller which will uninstall both your main project as well as the merge module simultaneously. To specify this option, select the **Uninstall Merge Module when parent is uninstalled** option on the **Install** tab of the **Dynamic Merge Module** customizer on the **Organization > Modules** task.



Note • The **Uninstall Merge Module when parent is uninstalled** option is selected by default for new InstallAnywhere 2011 projects. For projects made with previous releases of InstallAnywhere and opened in InstallAnywhere 2011, this option will not be selected.

You can also specify this option for an individual Merge Module by selecting the **Uninstall Merge Module when parent is uninstalled** option on the **Install Merge Module** action customizer in the **Install** task.

Similarities Between Pre and Post Install and Uninstall Tasks

The basic **Install** and **Uninstall** tasks all follow the same organization and have the same options. The differences between the tasks are determined solely by the time of their occurrence in the installation or uninstallation processes as shown below:

Installer

- Pre-Install Task
- File Installation (determined by the Install task)
- Post-Install Task

Uninstaller

- Pre-Uninstall Task
- File Uninstallation (determined by the Install task)
- Post-Uninstall Task

Each task enables developers to add, remove, or order actions. The actions will occur in the order they appear in the action list. Actions may be moved by dragging and dropping, or by selecting and using the ordering arrows.

Use these tasks to add the Panel, Console, and General actions needed by your installer and uninstaller. Some General Actions (such as Modify Text File) used in these tasks have functionality at both install time and uninstall time if used in the Install task. The uninstall functionality (reverting the changes made) will not be activated if the action is not in the Install task.

Advanced Designer Interface

Because these tasks are so similar, the options displayed in the **Advanced Designer** for the **Pre-Install**, **Install**, **Post-Install**, **Pre-Uninstall**, **Uninstall**, and **Post-Uninstall** task panels are also very similar to each other. These options are explained in detail in the [Reference](#) section:

- Pre-Install
- Install
- Post-Install
- Pre-Uninstall
- Uninstall
- Post-Uninstall

Variables

During installation, InstallAnywhere keeps track of dynamic values through the use of variables. Almost every dynamic value in InstallAnywhere, such as the path that a Magic Folder refers to, is represented by an InstallAnywhere Variable. Variables may be modified or accessed in order to affect the functionality or behavior of an installer.

InstallAnywhere variables are the key to any InstallAnywhere based installation. They allow developers to control the flow of information and the flow of the installation. Variables let developers store information from the system and information input by end users and create rules to determine operations based on that information. Developers can even output that information to configuration files or other resources to be used by the application.

Table 3-18 • Variables

Topic	Description
Variable Notation	Describes the notation InstallAnywhere users apply to refer to InstallAnywhere variables.
Magic Folders and Variables	Discusses the associations between magic folders and InstallAnywhere variables.
Methods of Setting InstallAnywhere Variables	Discusses various methods of setting variables.
Evaluation of InstallAnywhere Variables	Explains how InstallAnywhere variable values are resolved.
Searching for InstallAnywhere Variables	Explains how to search a project for a specific variable.
Selecting Variables from a List	Explains how to enter variable names into fields in your installer by selecting them from a list.

Variable Notation

Variables are referenced by a variable name. For instance, USER_INSTALL_DIR is the variable that contains the folder where the user has selected to install files to. Developers can set USER_INSTALL_DIR if they want to change the default location where files are installed. InstallAnywhere variables are of the form \$NAME\$. To get the value of a variable, use the notation \$NAME\$, as in \$USER_INSTALL_DIR\$. Technically, the \$ notation means “substitute the value of the variable here.” For simplicity in this document, the \$ notation is used throughout.

Magic Folders and Variables

Every Magic Folder has an associated InstallAnywhere variable. These variables are first initialized when the installer begins. Changing the value of a Magic Folder variable will change the destination to which the Magic Folders installs. Changing the value of the \$USER_INSTALL_DIR\$ through InstallAnywhere will change where the files will install.

With three exceptions, these variables are initialized at install time and will not change except through using custom code or the **Set InstallAnywhere Variable** action. The exceptions are:

- **Exception #1: \$USER_INSTALL_DIR\$**—This is initialized to the default value determined in the **Platforms** task in the Advanced Designer. Its value can change at the **Choose Install Folder** step if the end user selects a different folder.
- **Exception #2: \$USER_SHORTCUTS\$**—This is initialized to the default value determined by the Platforms task in the Advanced Designer. Its value can change at the **Create Alias, Link, Shortcut** install step if the end user selects a different location.
- **Exception #3: \$JAVA_HOME\$**
 - **Installer without VM**—Defaults to the value of the Java property java.home. Its value can change at the **Choose Java Virtual Machine** step if the end user selects a VM.
 - **Installer with VM**—Defaults to the value specified in the **General** subtab of the **Search Panel Settings** tab of the **Project > JVM Settings** task. It can change when the \$USER_INSTALL_DIR\$ changes or at the **Choose Java Virtual Machine** step if the end user selects a VM already on their machine.



Note • Variables cannot be set to themselves unless they are defined with the **Evaluate at Assignment** option. For variables defined without **Evaluate at Assignment** checked, you cannot set USER_MAGIC_FOLDER_1 = USER_MAGIC_FOLDER_1\$/test to append /test to USER_MAGIC_FOLDER_1. InstallAnywhere only allows direct and indirect recursion with InstallAnywhere variables that use **Evaluate at Assignment**. Otherwise, this condition causes an error.

Methods of Setting InstallAnywhere Variables

InstallAnywhere variables can be set in several ways:

Table 3-19 • Methods of Setting InstallAnywhere Variables

Method	Description
Provided by the Java Virtual Machine	InstallAnywhere installers have access to all properties of the Java Virtual Machine. These are stored in the variable set \$prop.PROPERTY_NAME\$ where PROPERTY_NAME is the name of the Java property.
Imported From the Environment	InstallAnywhere's LaunchAnywhere technology imports all environment variables on Windows and Unix based systems. These values are stored in the variable set \$1ax.n1.env.VARIABLE_NAME\$ (For example, on Unix systems, \$1ax.n1.env.PATH\$ would take the value of the PATH environment variable). These variables are imported when the installer is launched and are read-only. To set environment variables, use the Set Environment Variable action.
Set by Specific Actions	The Set InstallAnywhere Variable - Single Variable and Set InstallAnywhere Variable - Multiple Variables actions set InstallAnywhere variables to specific values.
Set via User Input	Most Panel and Console actions set InstallAnywhere variables.
Set by Installer/Uninstaller Progress	As the installer and uninstaller progress certain values are set or updated (such as \$INSTALL_SUCCESS\$).
Set via Custom Code	InstallAnywhere variables can be set using custom code.

Evaluation of InstallAnywhere Variables

InstallAnywhere variables can be set by reference or by value (evaluated at assignment). The default behavior for InstallAnywhere variables is to be set by reference—evaluated at execution time. However, variables defined by the Advanced Designer actions, **Set InstallAnywhere Variable - Multiple Variables** and **Set InstallAnywhere Variable - Single Variable**, can be expressly set to **Evaluate at Assignment**. Hence, these variables keep the value they have when they are set.

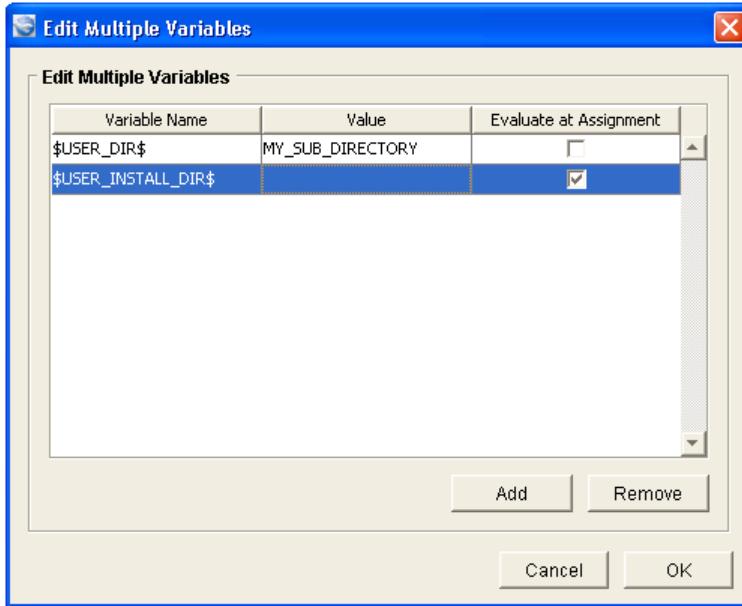


Figure 3-1: Evaluate at Assignment Option on the Edit Multiple Variables Dialog Box

Using **Evaluate at Assignment** allows you to define variables that are self-referencing. For example, a construction like `$USER_DIR$=$USER_DIR$+"MY_SUB_DIRECTORY"` can be used to redefine the value for `$USER_DIR$`. The same expression, without the **Evaluate at Assignment** setting, creates an infinite loop as it attempts to resolve `$USER_DIR$`.

Evaluate at Assignment Checked

The following is an example of evaluating an InstallAnywhere variable at assignment:

```
A=$USER_DIR$  
B=A          //Set using Evaluate at Assignment (set by value).  
  
A=A+"/mydir/" //Set using Evaluate at Assignment.  
print B
```

In the code above, B resolves to `$USER_DIR$`.

Evaluate at Assignment Unchecked

The following is an example of evaluating an InstallAnywhere variable by reference:

```
A=$USER_DIR$  
B=A          //Set without using Evaluate at Assignment (set by reference).  
  
A=A+"/mydir/" //Set using Evaluate at Assignment.  
print B
```

In the code above, B resolves to \$USER_DIR\$/mydir/.

Results of Evaluate at Assignment

The following table compares the two methods of evaluating variables:

Table 3-20 • Results of Evaluate at Assignment

Expression	Evaluate at Assignment	Result
\$MY_VAR\$=\$MY_VAR\$+".txt"	Checked	Appends .txt to the value of \$MY_VAR\$.
\$MY_VAR\$=\$MY_VAR\$+".txt"	Unchecked	Code error. Infinite loop.

Searching for InstallAnywhere Variables



Edition • This feature is included with InstallAnywhere Enterprise Edition.

You can search a project to locate all references to a specific variable. You can choose to search for a variable in any of the installation tasks/phases (**Pre-Install**, **Install**, **Post-Install**, **Pre-Uninstall**, **Uninstall**, **Post-Uninstall**) and can choose to search **Features** and/or **Components**. Search results are displayed on the **Search Results** dialog box. You can also perform global replacements of variables.

- Performing a Search for a Variable
- Performing a Search and Replace of a Variable

Performing a Search for a Variable

To search your installation project to locate all references to a specific variable, perform the following steps:

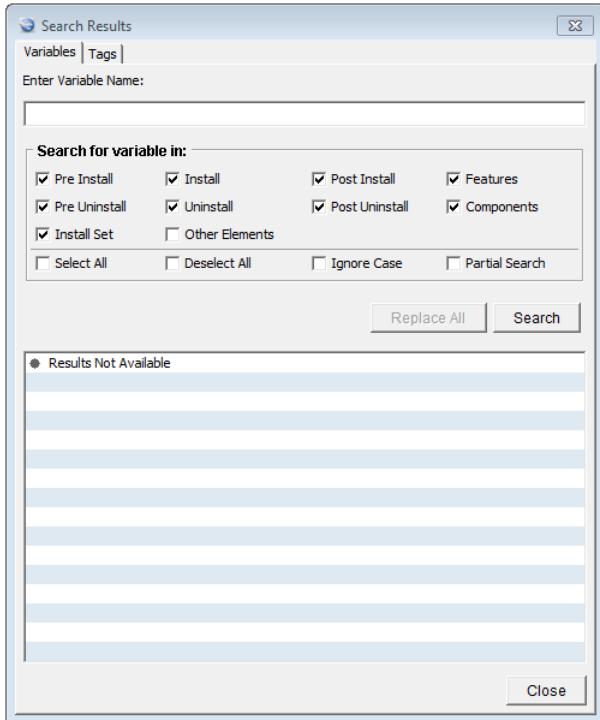


Task:

To search for a variable:

1. Open a project in the Advanced Designer.
2. On the **File** menu, click **Search** (or press Ctrl + F). The **Search Results** dialog box opens.

3. Open the **Variables** tab.



4. In the **Enter Variable Name** field, enter the name of the variable you want to search for.

- **Exact search**—If you want to search for an exact match of a variable, enter the complete variable name in this field and leave the **Partial Search** option unselected.



Note • It is always recommended that you enter the \$ symbols before and after the variable name.

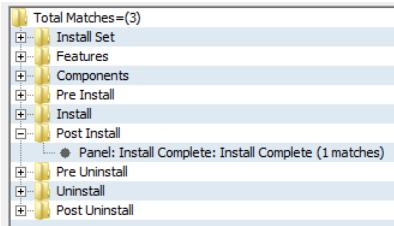
- **Partial search**—If you want to search for a specific text string in a variable, enter the text string in this field and also select the **Partial Search** option.



Note • After performing a partial search, the Replace feature is not available.

5. If you want to perform a case insensitive search, select the **Ignore Case** option.
6. Under **Search for variable in**, select the installation phases (**Pre-Install**, **Install**, **Post-Install**, **Pre-Uninstall**, **Uninstall**, and/or **Post-Uninstall**) and product elements (**Features** and/or **Components**) that you want to search.
7. Click **Search** to initiate a search for the specified variable (or text string in variable) in the selected installation phases and project elements.

8. After a search is performed, results are listed in a tree structure with the total number of variables found listed at the top, and items associated with the selected variable listed below, grouped by category.



Performing a Search and Replace of a Variable

To perform a search and replace of a specific variable, perform the following steps:

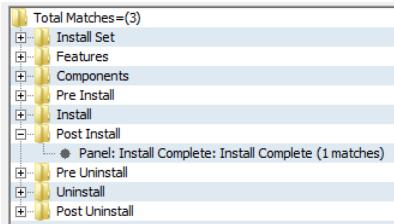
**Task:****To perform a search and replace of a specific variable:**

1. Search for a variable, as described in [Performing a Search for a Variable](#).

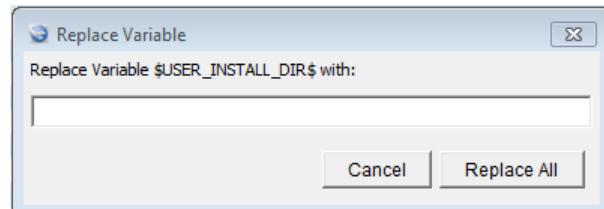


Note • Enter the full variable name in the **Enter Variable Name** text box, and do not select the **Partial Search** option.

After a search is performed, results are listed in a tree structure with the total number of variables found listed at the top, and items associated with the selected variable listed below, grouped by category.



2. Click **Replace All** to replace all instances of the found variable with a replacement value. The **Replace Variable** dialog box opens, prompting you to enter the replacement value.



3. Enter the replacement value in the **Replace Variable VARIABLE_NAME with** field.



Note • It is always recommended that you enter the \$ symbols before and after the replacement variable name. For example instead of searching for \$NAME\$ and replacing it with NEWNAME, you should replace it with \$NEWNAME\$.

4. Click **Replace All**. All found instances of the variable will be replaced with the replacement variable.

About Replacing Magic Folder Variables

If you choose to replace a magic folder variable with another variable, InstallAnywhere will simply copy the children of the source magic folder to the destination magic folder variable.



Note • There are known issues regarding replacing the \$USER_INSTALL_DIR\$ magic folder. Therefore, searching for and replacing the \$USER_INSTALL_DIR\$ magic folder is not recommended.

Selecting Variables from a List

Instead of having to manually type in the name of variables in text boxes or text areas in your installation project, you can open the **Choose Variable** dialog box and select a variable from a list.

While you are editing the contents of a text box or a text area, you can open the **Choose Variable** dialog box by pressing Alt + V.

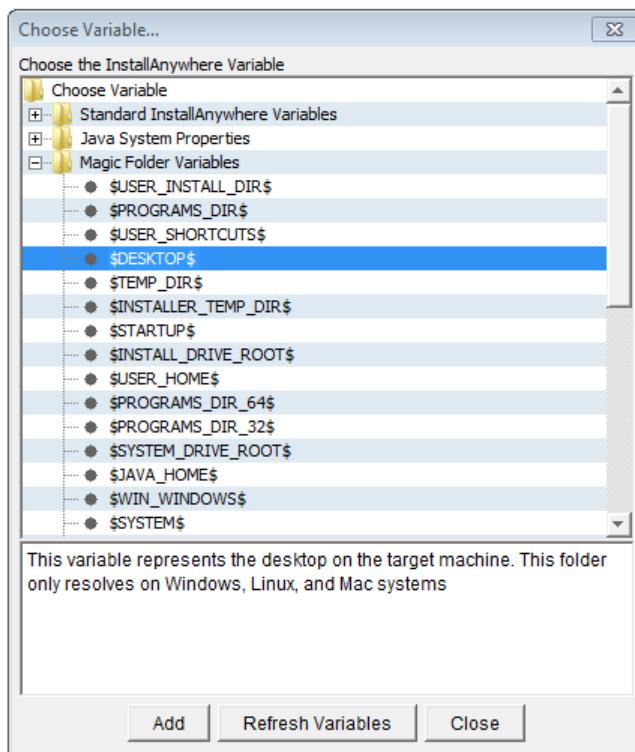


Figure 3-2: Choose Variable Dialog Box

The **Choose Variable** dialog box lists all of the variables available in your project, grouped by category, with embedded help for the selected variable. Whenever you add a new variable to your installation project, the **Choose Variable** dialog box is refreshed to include it. Variables are listed in the following categories:

- Standard InstallAnywhere Variables
- Java System Properties
- Magic Folder Variables
- User Defined Magic Folders
- User Defined Variables

Click **Add** to add the selected variable to the currently active text box. Click **Refresh Variables** to manually refresh the variable list.

The **Choose Variable** dialog box remains open until you click the **Close** button.

InstallAnywhere Tutorials

InstallAnywhere has two authoring modes. The **Project Wizard** makes the process easier by making choices for developers. The **Advanced Designer** gives developers greater control of an installer project.

To explore using InstallAnywhere in a guided exercise using both the Project Wizard and Advanced Designer interfaces, see:

- [Building an Installer Using the Project Wizard](#)
- [Building an Installer Using the Advanced Designer](#)



Note • *InstallAnywhere opens displaying the first frame of the Project Wizard unless the default preference has been changed using the **Preferences** command in the **Edit** menu. To access an existing InstallAnywhere project, click **Open Existing Project**. Click the project name in the **Open Recent Project** list then click **Advanced Designer**.*

The general process for developing an InstallAnywhere project using either of these interfaces includes the following steps:

- Step 1: Create New Project
- Step 2: Set Project Information
- Step 3: Add Pre-Install Actions
- Step 4: Install Tasks
- Step 5: Add Post-install Actions
- Step 6: Set Installer UI Options
- Step 7: Configure the Project Uninstaller
- Step 8: Build
- Step 9: Test

Building an Installer Using the Project Wizard

The introductory tutorial builds an installer using the Project Wizard, which does not allow configuring Pre-Install or Post-Install Actions. It is possible to build a basic installer with the Project Wizard in six steps and about five minutes. The following tutorial teaches how to build an installer for a sample Java application, called “Office Suite for Java,” which is included in the InstallAnywhere folder.

- [Creating a New Project](#)
- [Defining the Installation Tasks](#)
- [Building the Installer](#)
- [Testing the Installer](#)

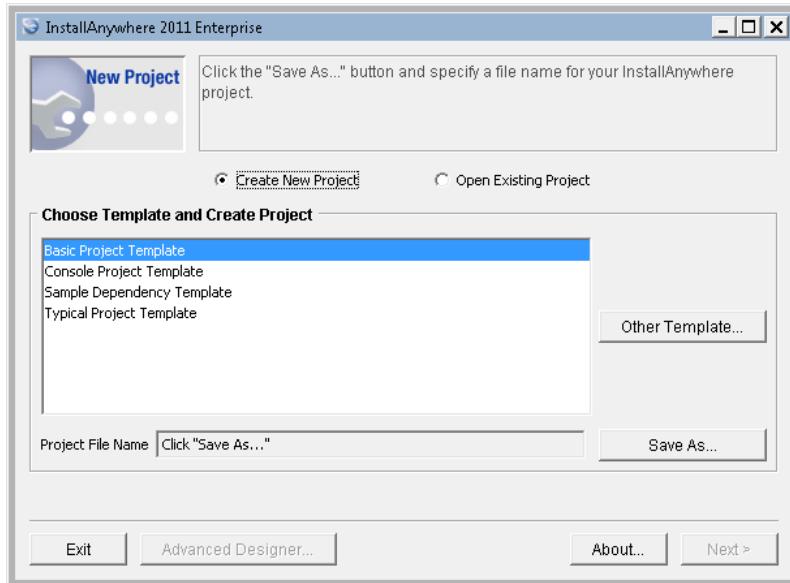
Creating a New Project

The first step in creating an installer using the Project Wizard consists of entering a name and specifying a location to save the new project.

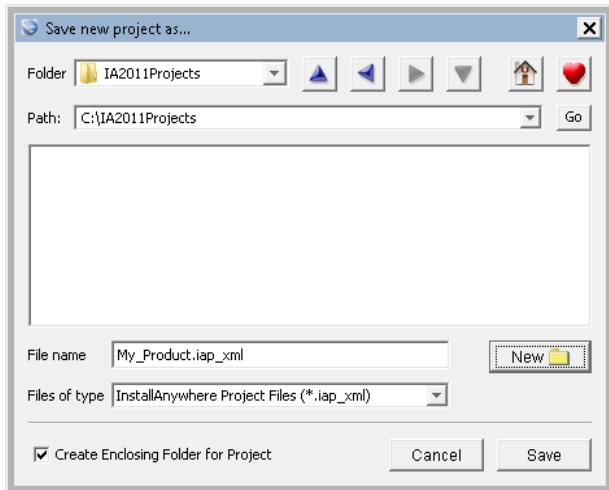


Task: *To create a new project*

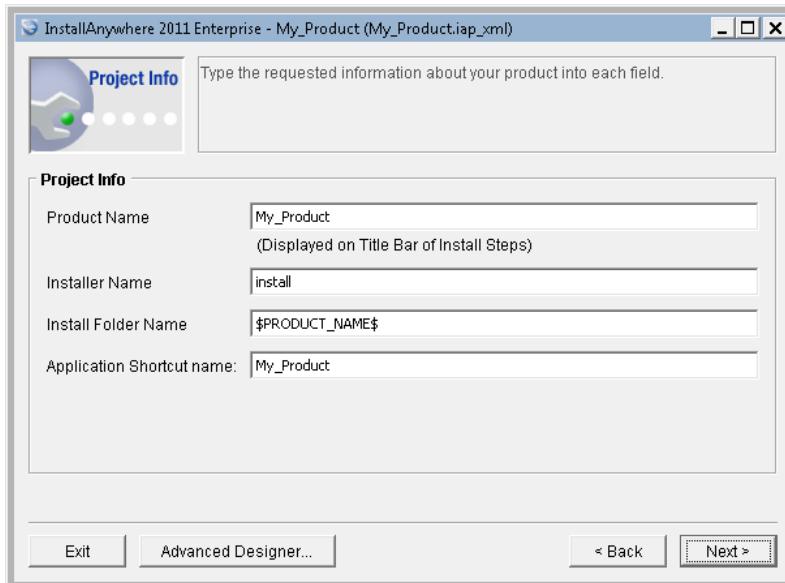
1. Launch InstallAnywhere. The **New Project** panel of the Project Wizard opens, with the **Create New Project** option selected by default.



2. Click **Save As**. The **Save New Project As** dialog box opens. By default, the project is named [My_Product.iap_xml](#).



3. Next to **Path**, click **Go** and select the location where you want to save this new project.
4. Click **Save** to confirm the name and close this dialog box and return to the Project Wizard.
5. Click **Next**. The **Project Info** panel of the New Project Wizard opens.



On the **Project Info** panel, you define basic information about the installer, such as the product name (as displayed on the installer), the name of the installer to be produced, the name of the destination folder, and the application name.

Chapter 4: InstallAnywhere Tutorials

Building an Installer Using the Project Wizard

6. For this tutorial, enter the following information:

Field	Enter
Installer Name	OfficeSuite
Install Folder Name	OS
Application Shortcut Name	OfficeSuite

7. Click **Next**. The **Add Files** panel opens. Proceed with the tasks in [Defining the Installation Tasks](#).

Defining the Installation Tasks

The next step in creating an installer using the Project Wizard consists of adding files, choosing the main classpath, and setting the classpath.

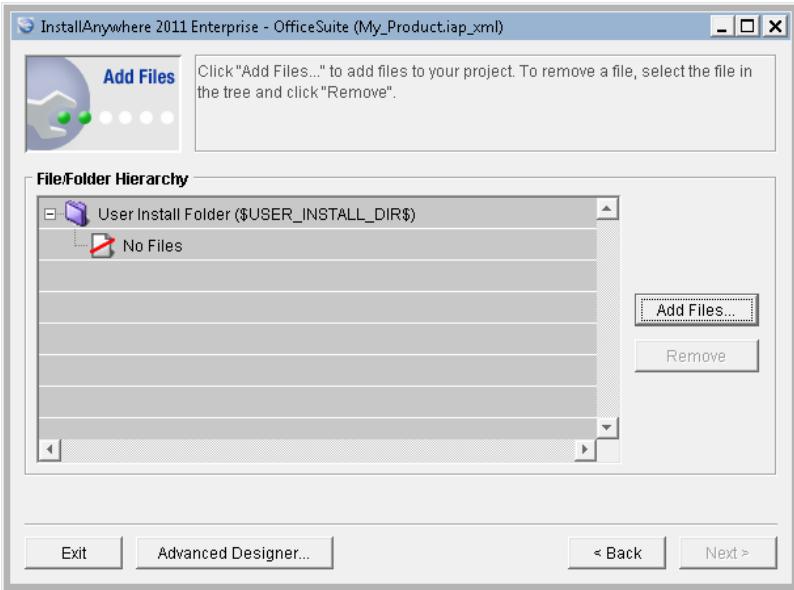
- [Adding Files](#)
- [Choosing the Main Class](#)
- [Setting the Classpath](#)

Adding Files

The next step is to click **Add Files** to add files and folders to your installer.

 **Task:** *To add files:*

1. Perform the steps in [Creating a New Project](#). The **Add Files** panel opens.

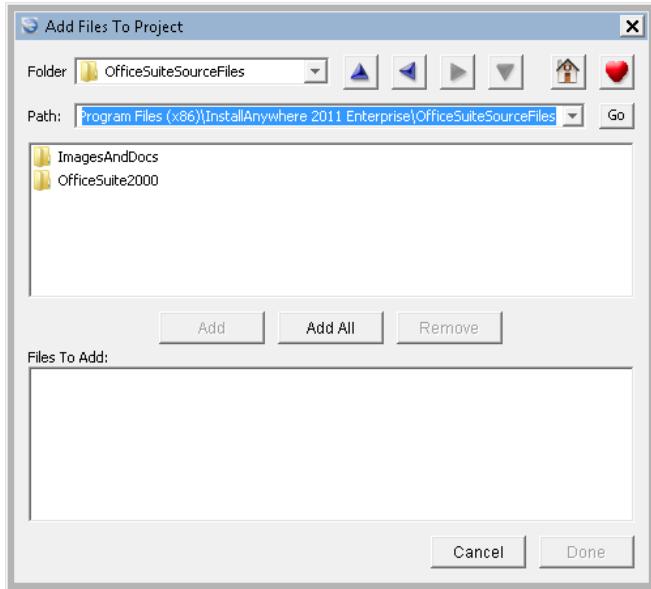


2. Click **Add Files**. The **Add Files to Project** dialog box opens.

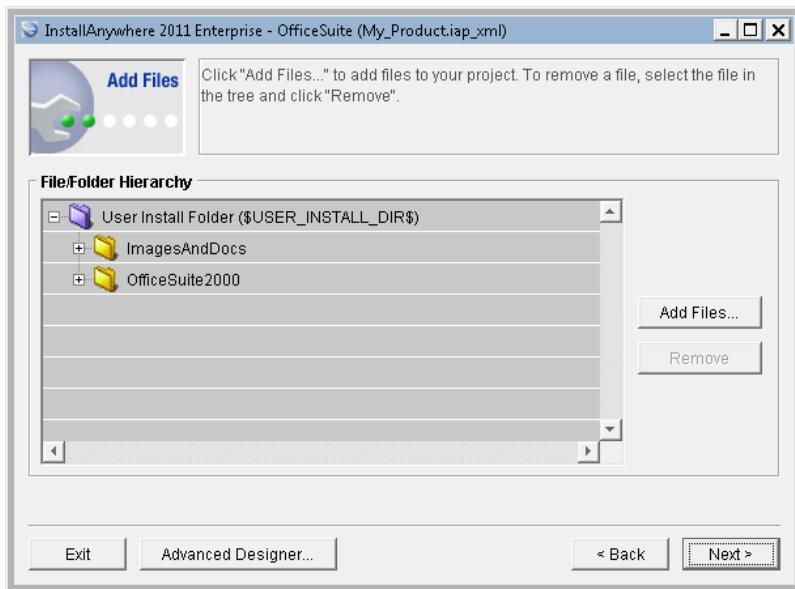
Chapter 4: InstallAnywhere Tutorials

Building an Installer Using the Project Wizard

3. Browse to the OfficeSuiteSourceFiles directory, which is located inside the InstallAnywhere installation directory. Two directories are listed: ImagesAndDocs and OfficeSuite2000.



4. Click **Add All** to add the ImagesAndDocs and OfficeSuite2000 subfolders of the OfficeSuiteSourceFiles folder. These folders are now listed in the **Files to Add** list.
5. Click **Done**. The selected folders are now listed in the **File/Folder Hierarchy** list.



6. Click **Next**. The **Choose Main** panel opens. Proceed with the tasks in [Choosing the Main Class](#).

Choosing the Main Class

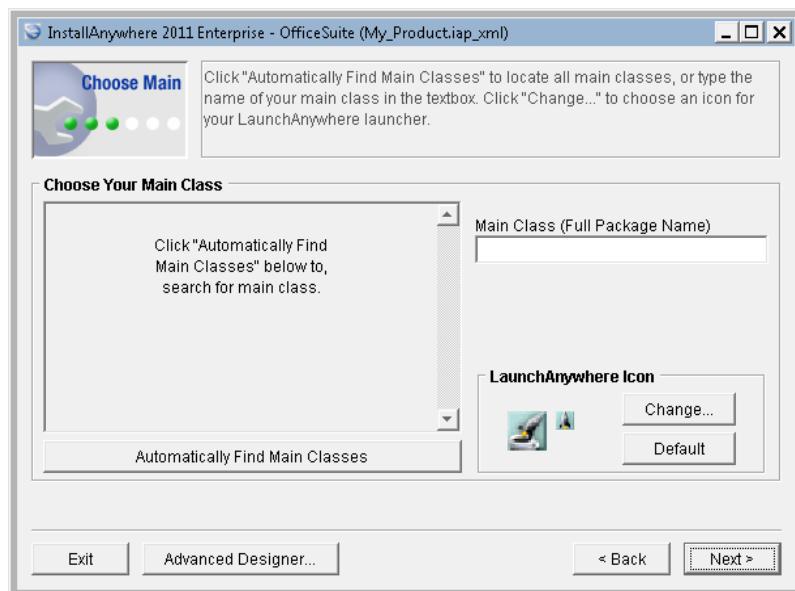
On the **Choose Main** panel, you select the starting class for the application. You may also specify custom icons (in GIF format) for your LaunchAnywhere executable.



Task:

To choose the main class:

1. Perform the steps in [Adding Files](#). The **Choose Main** panel opens.



2. Click **Automatically Find Main Classes**. InstallAnywhere automatically locates all main classes of the OfficeSuite application and adds them to the **Choose Your Main Class** list. For OfficeSuite there is only one main class (`com.acme.OfficeSuite`), and it is selected by default.



Note • If you were not installing a Java application, you could skip ahead by clicking **Next** without specifying a main class. Then, when the **Choose a Main Class** dialog box opens, you could click **No** to move to the next step.

3. Click **Next**. The **Classpath** panel opens. Proceed with the tasks in [Setting the Classpath](#).

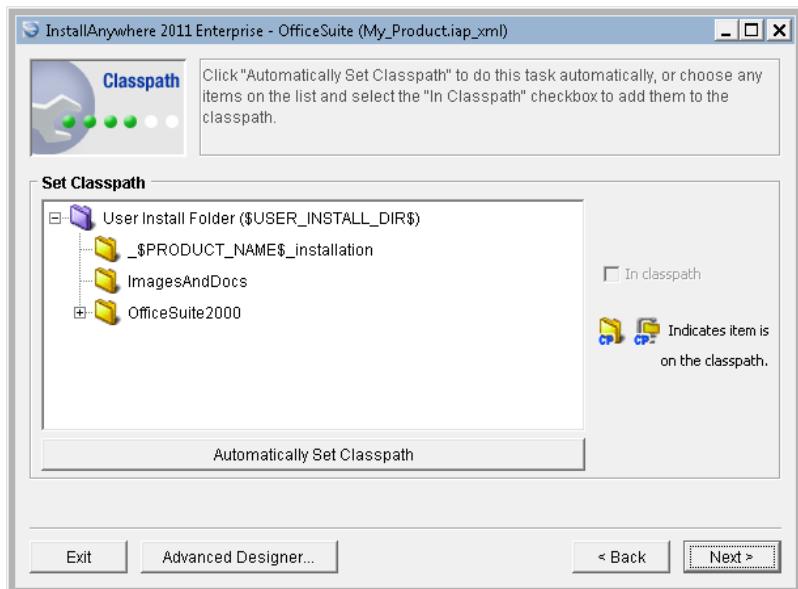
Setting the Classpath

On the **Classpath** panel, you configure a Java application classpath automatically. You could also choose any listed item and select the **In classpath** checkbox to add them to the classpath manually.

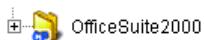


Task: *To set the classpath:*

1. Perform the steps in [Choosing the Main Class](#). The **Classpath** panel opens.



2. Click **Automatically Set Classpath**. InstallAnywhere calculates which files need to be added to the classpath. A small CP icon appears on the folders that contain those files. In this tutorial, a CP icon appears next to the **OfficeSuite2000** folder.



3. Click **Next**. The **Build Installer** panel opens. Proceed with the steps in [Building the Installer](#).

Building the Installer

On the **Build Installer** panel you select all of the target installation platforms that you want to include with your installer, and then you click **Build** to build the project.

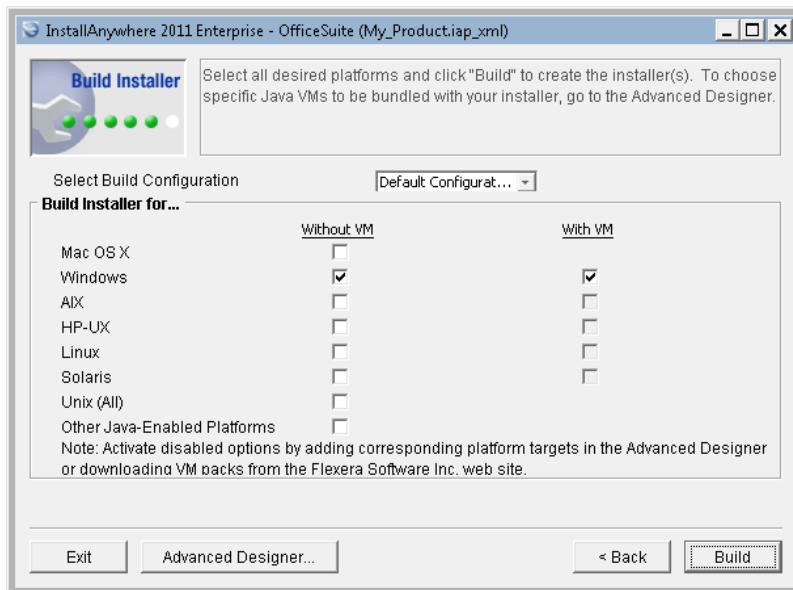
The first several items on the Build Installer screen, from **Mac OS X** through **Unix (All)**, represent installers that are double-clickable on their respective platforms. The final option, **Other Java-Enabled Platforms**, is a “pure” Java installer that can be invoked from the command line on any Java-enabled platform.

You may choose to build installers with an embedded Virtual Machine by clicking the **With VM** checkbox next to the target installation platform. Installers that are built without VMs are smaller, and download faster than installers bundled with one. The InstallAnywhere Web Install process will allow end users to choose the appropriate installer for their system.



Task: *To build the installer:*

1. Perform the steps in [Setting the Classpath](#). The **Build Installer** panel opens.



2. Select the appropriate target installation platforms, such as **Windows**.
3. Select the **With VM** checkbox next to the platform you selected.
4. Click **Build**. The installer will be built and the **Try Installer** panel will open. Proceed with the steps in [Testing the Installer](#).



Important • *The installer folder will be placed in the same location as the project file. This location cannot be changed.*

Testing the Installer

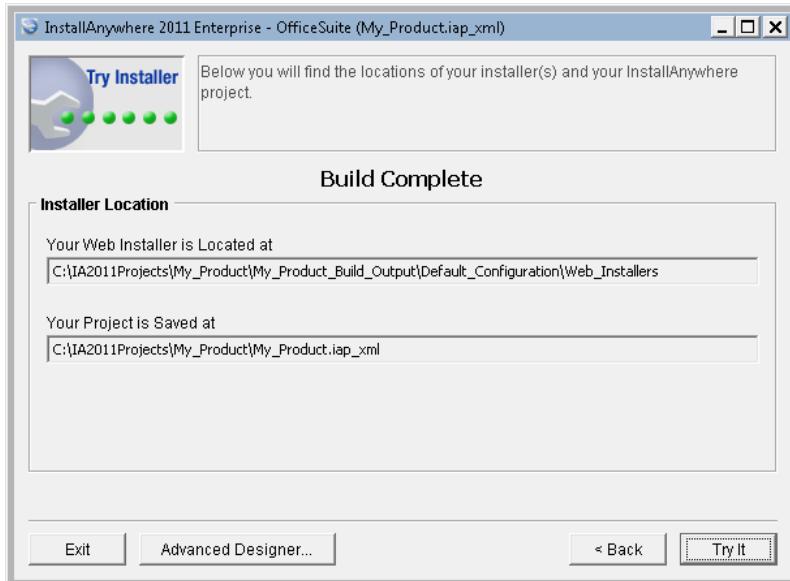
Now that an installer has been built, it can be tested.



Task:

To test an installer:

1. Perform the steps in [Building the Installer](#). The **Try Installer** panel opens.



2. Click **Try It**. The OfficeSuite installation opens.
3. Install OfficeSuite by following the prompts and accepting the defaults.
4. When the installation is complete, click **Exit** on the Project Wizard to close InstallAnywhere.
5. Launch the application by doing one of the following:
 - On Windows, go to the **OfficeSuite** program group and choose **OfficeSuite**.
 - On Unix, cd to the directory where the program was installed and type `./OfficeSuite`.
 - On Mac OS X, double-click the **OfficeSuite** icon.
6. After launching OfficeSuite, quit by selecting **Exit** from its **File** menu.



Note • It is possible to post the installer folder to a Web server and install the software onto another platform as well. Hold down the Control (Ctrl) key while the installer launches to see the debug output (Windows only).

Building Installers for Other Platforms

When building for a platform other than that on which the installer is being developed, transfer that installer, and run it manually. By default, installers will be located in the `Build_Output` directories found in the same folder as the `.iap_xml` project file. Within the `Build_Output` folder, will be `Web_Installers`, `CDROM_installers`, or both. From within each of these directories, choose the platform to test. For the CDROM installer, transfer the entire contents of the `CDROM_Installers` folder.

Building an Installer Using the Advanced Designer

The Advanced Designer offers a much wider range of configuration over InstallAnywhere's many options than the Project Wizard. You can use the Advanced Designer to:

- Assign files to different Install Sets.
- Install multiple LaunchAnywhere executables.
- Add custom panels to the installer.
- Customize the installer's look and feel.
- Set rules to selectively install files to specific platforms.
- Specify Build Configurations.

The InstallAnywhere Advanced Designer breaks down the installer creation process into discrete tasks. These tasks are listed on the left side of the screen.

Chapter 4: InstallAnywhere Tutorials

Building an Installer Using the Advanced Designer

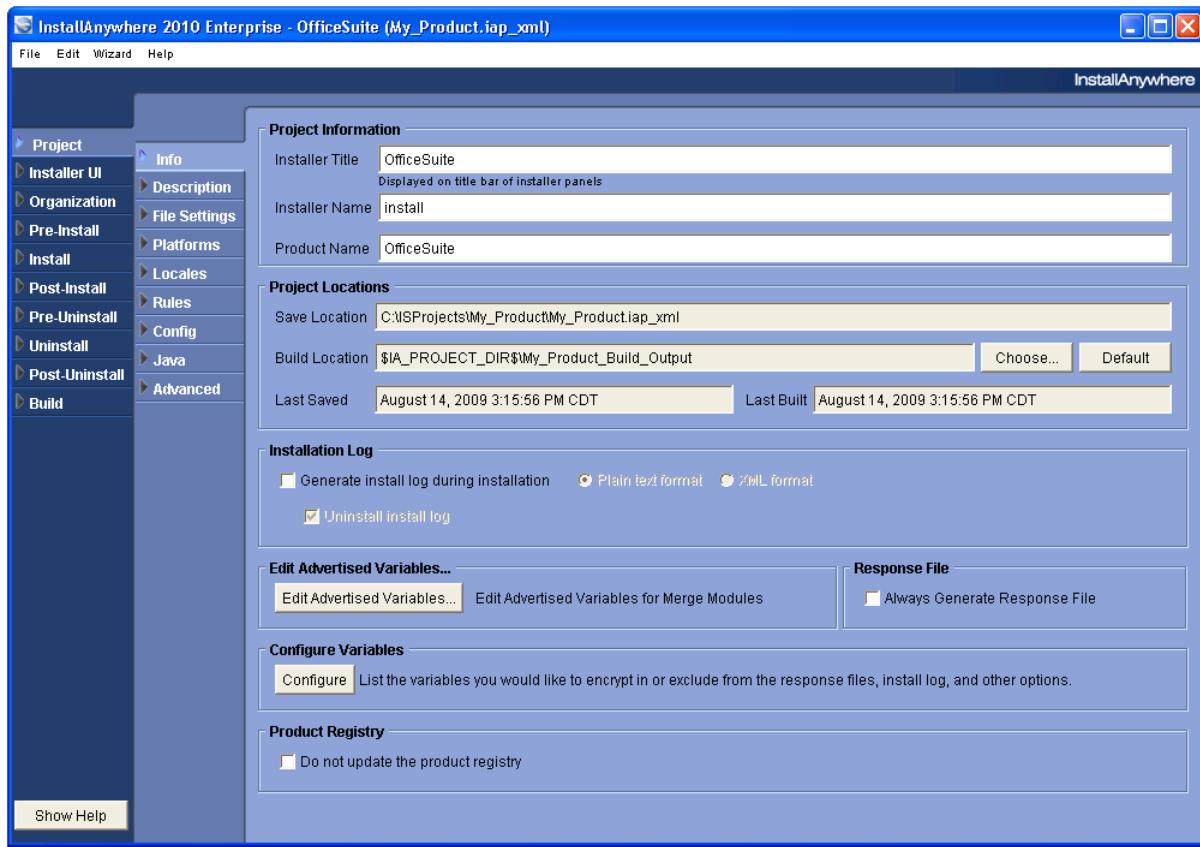


Figure 4-1: Tasks and Subtasks Displayed in the Advanced Designer

This tutorial will rebuild the previous Office Suite Installer using the Advanced Designer.

- [Creating a New Project](#)
- [Adding Pre-Install Actions](#)
- [Defining the Installation Tasks](#)
- [Adding a LaunchAnywhere Executable to the Install Task](#)
- [Adding Post-Install Actions](#)
- [Building the Installer](#)
- [Testing the Installer](#)



Note • You can enter the Advanced Designer mode from the Project Wizard by clicking the **Advanced Designer** button at any point of the installer creation process. Many developers use the Project Wizard to create a basic installer and then switch to the Advanced Designer in order to customize their installer with more advanced functionality.

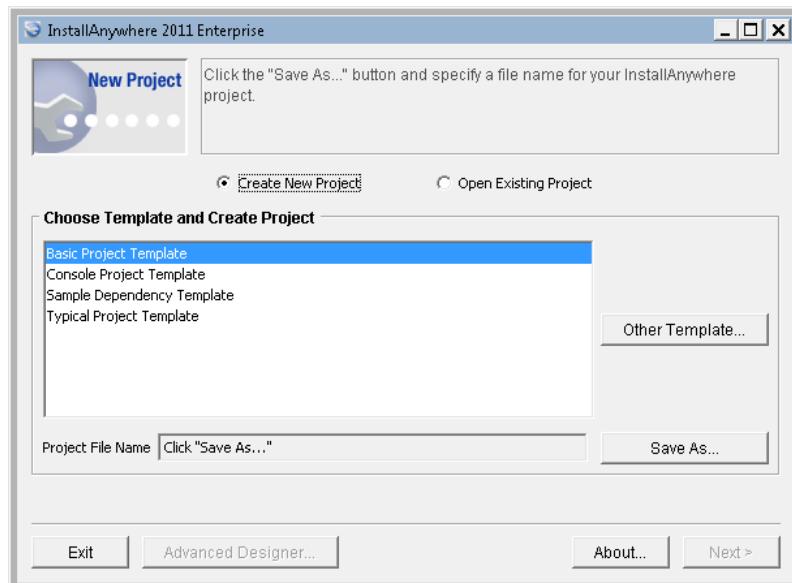
Creating a New Project

The first step in using the Advanced Designer is to create a new project and then open the Advanced Designer interface.



Task: *To create a new project*

1. Launch InstallAnywhere. The **New Project** panel opens, with the **Create New Project** option selected by default.

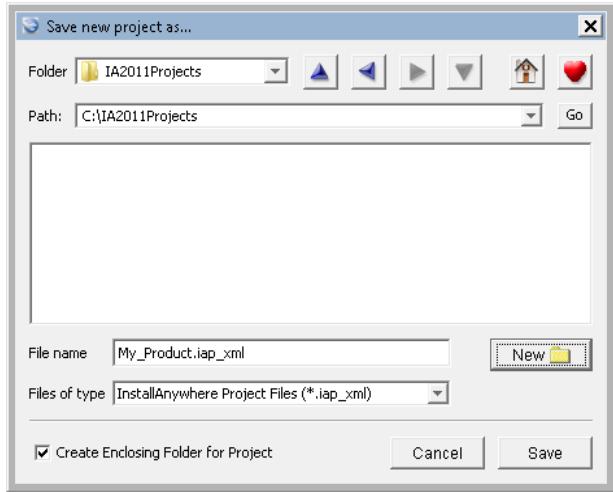


2. Select the **Basic Project Template**. This template should already be selected.

Chapter 4: InstallAnywhere Tutorials

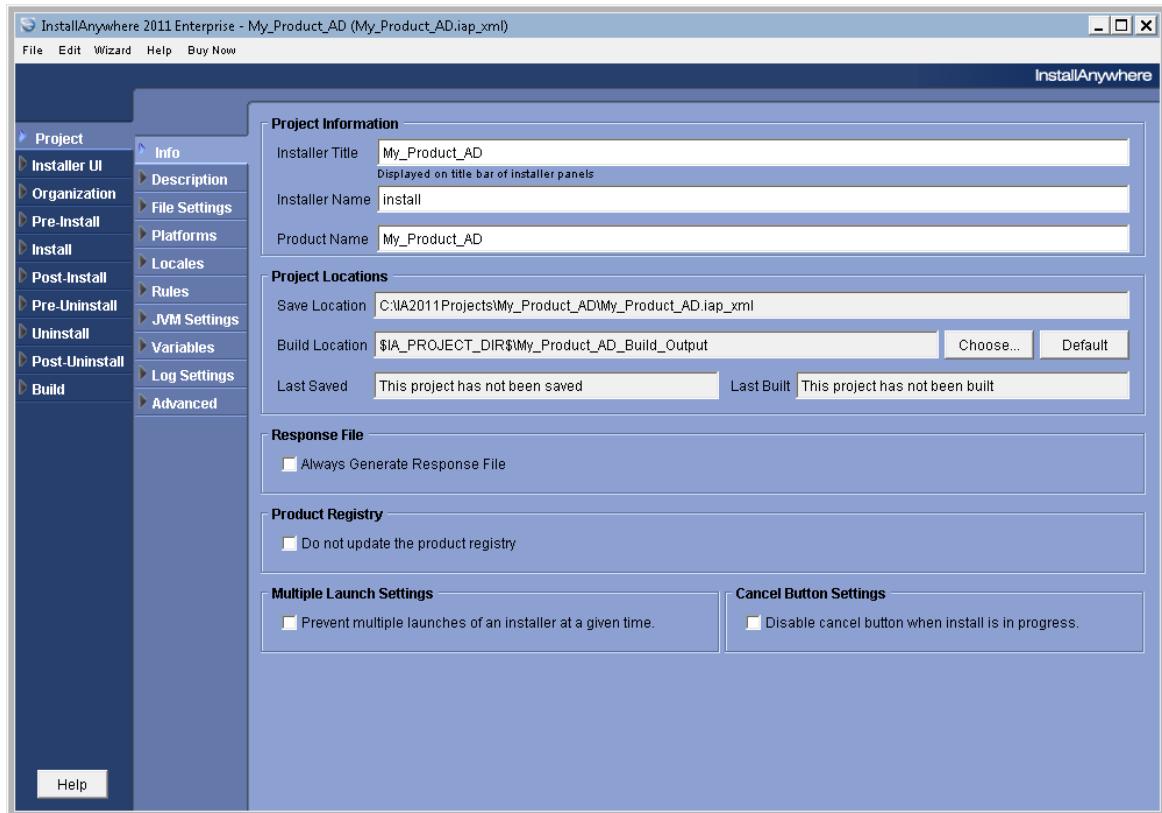
Building an Installer Using the Advanced Designer

3. Click **Save As**. The **Save New Project As** dialog box opens. By default, the project is named [My_Product.iap_xml](#).



4. In the File name box, edit the name of this project so that it is named [My_Product_AD.iap_xml](#).
5. Next to **Path**, click **Go** and select the location where you want to save this new project.
6. Click **Save** to confirm the name and close this dialog box and return to the **New Project** panel.

- Click the **Advanced Designer** button. This selection will open the newly created project file in the InstallAnywhere Advanced Designer. Advanced Designer will open to the **Project > Info** task.



On the **Project > Info** task, you set the basic installer options such as the name of the product, the installer title, and the installer name. The installer name will be the name of the executable file InstallAnywhere creates. This tab also sets the location to build the installer and the settings for the generation of installation logs.

- In the **Product Name** field, enter **OfficeSuitePro**.
- In the **Installer Title** field, enter **OSPro**.
- For now, skip the **Installer UI** and **Organization** tasks, and click on the **Pre-Install** task. The **Pre-Install** task opens. Proceed with the steps in [Adding Pre-Install Actions](#).

Adding Pre-Install Actions

The next step in creating an installer using the Advanced Designer is to review and edit the **Pre-Install** actions. In this tutorial, you will add a choice to download a Java Virtual Machine.

**Task:** *To add pre-install actions:*

1. Perform the steps in [Creating a New Project](#). The **Pre-Install** task opens.



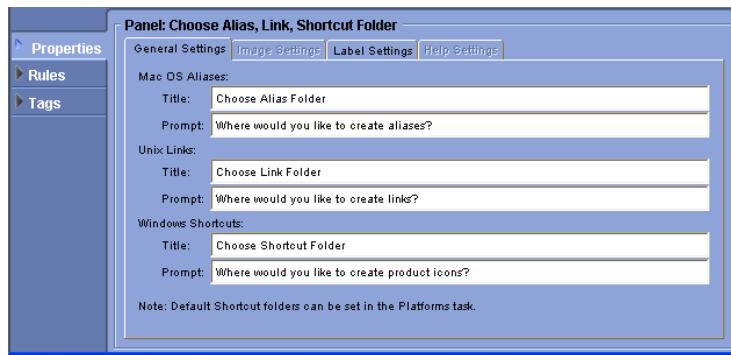
The **Pre-Install** task sets the panels and actions that occur prior to the installation of files. By default, a new InstallAnywhere project contains the following panels:

Panel	Description
Introduction	This panel allows developers to introduce the product or installation process.
Choose Install Folder	This panel allows end users to choose the installation location for the product.
Choose Alias, Link Shortcut Folder	This panel allows end users to specify the location for any Mac OS X Aliases, Windows Shortcuts, and Unix Symlinks (used as shortcuts) that will be installed.

Panel	Description
Pre-Install Summary	This panel provides end users with a summary of various installation settings prior to the installation of files.

Note the following regarding the **Pre-Install** task:

- **Order of actions**—Actions in the **Pre-Install** task will occur in the order set in the task list. The order of panels and actions can be manipulated using the Arrow buttons in the middle right of the Advanced Designer screen.
- **Customizers**—The behavior and content of panels can be modified by highlighting each panel. The dialog along the bottom half of the Advanced Designer will change to reflect the panel selected. In InstallAnywhere's vocabulary, this is known as a customizer, and is available for each action and panel in the installer. The following is an example of the customizer for the **Choose Alias, Link, Shortcut Folder** panel.

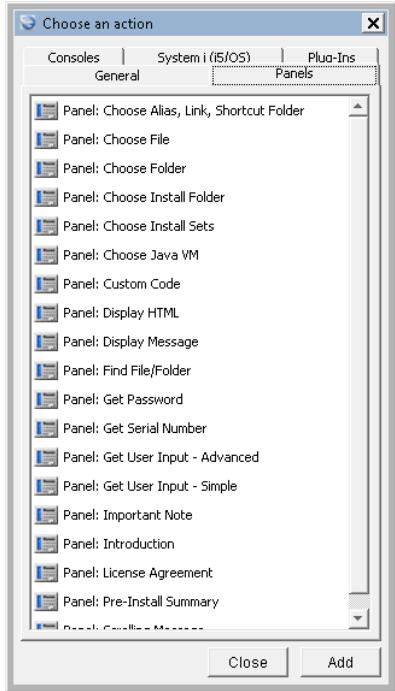


2. In the **Pre-Install Action List**, select **Panel: Choose Alias, Link, Shortcut Folder** and then click **Add Action**. The **Choose an Action** dialog box appears.

Chapter 4: InstallAnywhere Tutorials

Building an Installer Using the Advanced Designer

3. Select the **Panels** tab.



4. On the **Panels** tab, select **Panel: Choose Java VM** and click **Add**. The new action appears in the **Pre-Install Action** list and the **Choose Java VM** customizer opens.
5. Click **Close** to hide the **Choose an Action** dialog box.
6. Click on the **Install** task and proceed with the steps in [Defining the Installation Tasks](#).

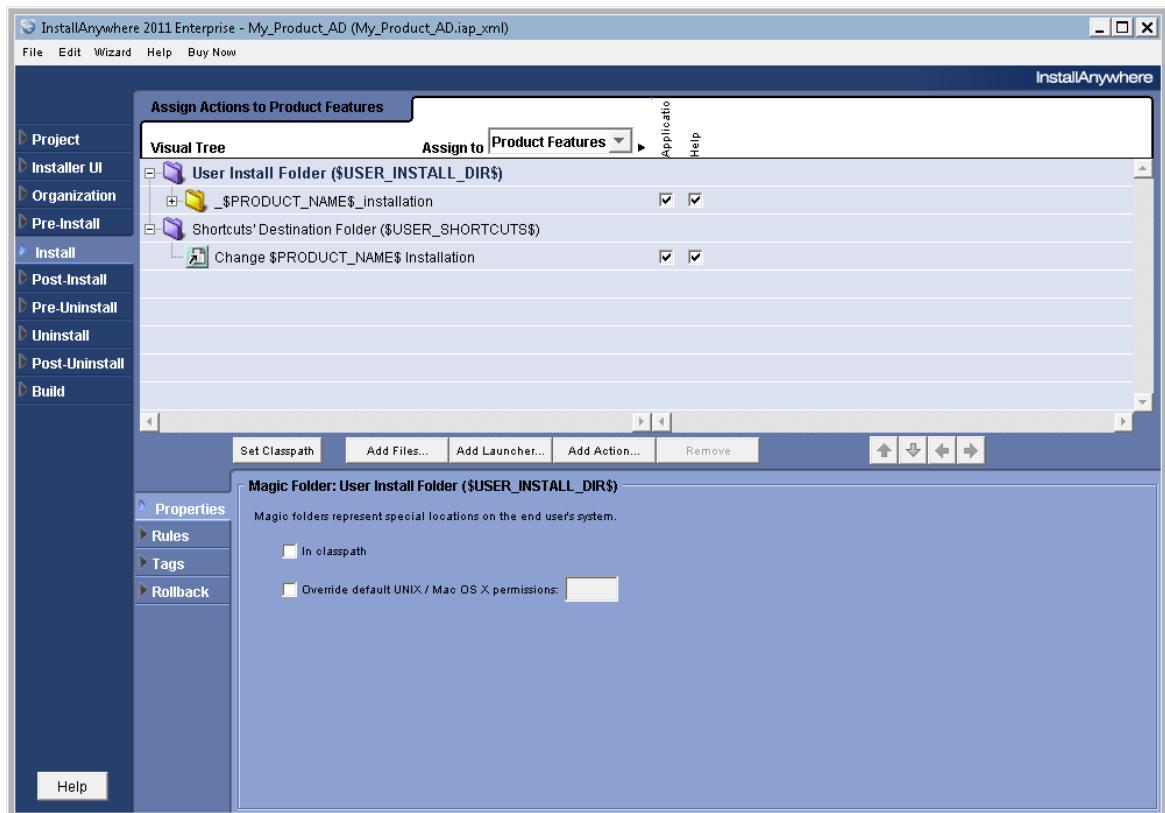
Defining the Installation Tasks

The **Install** task contains the install actions that take place during the step when InstallAnywhere deploys files to the target system. The **Install** task displays files and directories in the way the installer will deploy them on the target system. You can use the controls on the **Install** task to assign files, directories, and actions to either components or product features.



Task: *To define the installation tasks:*

1. Perform the steps in [Adding Pre-Install Actions](#). The **Install** task opens. The **Install** task defines the files to install, the folder location to install those files and the order of the tasks that need to happen as the files are being installed.



By default, the InstallAnywhere **Install** task has a folder named `$_PRODUCT_NAME$_installation`, which contains any InstallAnywhere uninstaller/Maintenance Mode actions, and a comment action with instructions pertaining to the uninstaller.

2. Actions (including, but not limited to, the installation of files) in the **Install** task list occur in order with actions at the top of the installation occurring first. Leave the Uninstaller creation in its default place in the installation (although the folder structure can be changed). For organizational purposes, it is generally best to have the uninstaller creation action first.



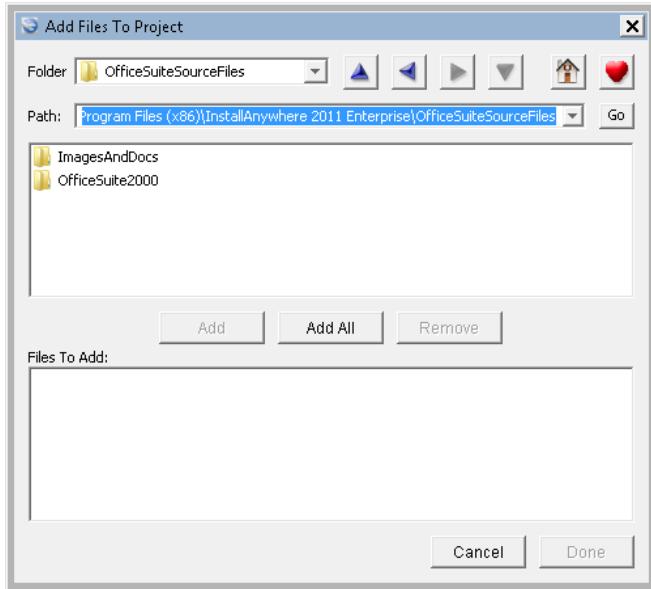
Tip • The Advanced Designer implements a Drag-and-Drop interface in many tabs and tasks. In the **Install** task, actions and files can be moved by selecting and dragging them. A dark underline appears in the location where the file or action will be placed.

3. Click **Add Files**. The **Add Files To Project** dialog box opens.

Chapter 4: InstallAnywhere Tutorials

Building an Installer Using the Advanced Designer

4. Browse to the OfficeSuiteSourceFiles directory, which is located inside the InstallAnywhere installation directory. Two directories are listed: ImagesAndDocs and OfficeSuite2000.



5. Click **Add All** to add the ImagesAndDocs and OfficeSuite2000 subfolders of the OfficeSuiteSourceFiles folder. These folders are now listed in the **Files to Add** list.
6. Click **Done**. The selected folders are now listed in the **Visual Tree** in the **Install** task.



File trees may be expanded or contracted within the **Install** task by clicking on the "+" or "-" boxes at the apex of the tree branches. Objects may be moved up and down or into and out of subfolders in the file tree by highlighting the object, and using the right, left, up, and down arrows found in the middle right of the **Install** task screen.

7. Proceed with the tasks in [Adding a LaunchAnywhere Executable to the Install Task](#).

Adding a LaunchAnywhere Executable to the Install Task

A LaunchAnywhere Executable (LAX) is a unique native executable, created by InstallAnywhere, that is used to launch a Java application. While the InstallAnywhere Wizard specifically asks to select a main class and automatically creates a single launcher, the Advanced Designer allows developers to add as many launchers as they would like. There are two ways to add a LaunchAnywhere Launcher to an InstallAnywhere Project file. The Create LaunchAnywhere for Java Application option may be selected from the Choose an Action dialog box or may be added by clicking the Add Launcher button on the middle control bar in the Advanced Designer.

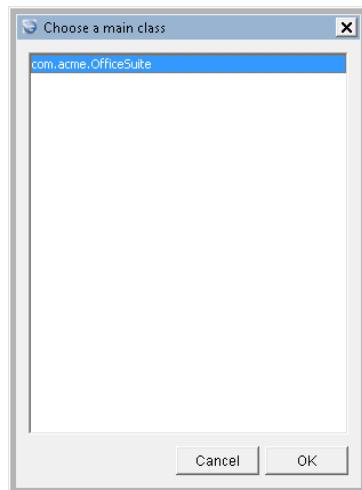


Task: *To add a LaunchAnywhere executable:*

1. Perform the steps in [Defining the Installation Tasks](#). The Install task is opened and the OfficeSuite2000 files and folders are listed in the **Visual Tree**.
2. Highlight the **User Install Folder** in the Visual Tree, and click the **Add Launcher** button. You are prompted to automatically find classes with main methods.



3. Click **OK**. The **Choose a main class** dialog box opens with a class listed.



Note • When adding a launcher, InstallAnywhere automatically inspects the added files (including introspecting into JAR and ZIP files) to find class files with Main Methods specified.

4. Choose the com.acme.OfficeSuite as main class for the application and click **OK**.

Chapter 4: InstallAnywhere Tutorials

Building an Installer Using the Advanced Designer

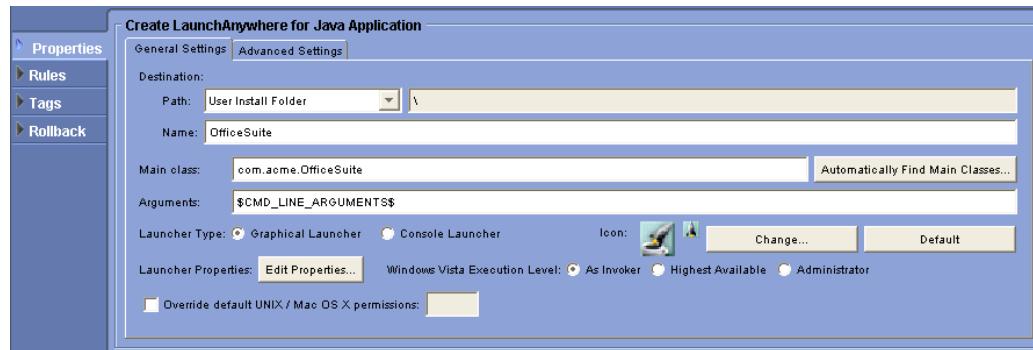


Note • Since **OfficeSuite** is a simple project, `com.acme.OfficeSuite` is the only class.

By clicking the **Add Launcher** button, you have not only added the launcher to the file structure (**OfficeSuite**), but also created a Shortcut, Link, or Alias action in the **Shortcuts' Destination Folder** Magic Folder. This location is variable and will be specified by the **Choose Alias, Link, and Shortcut** panel in the pre-install section.

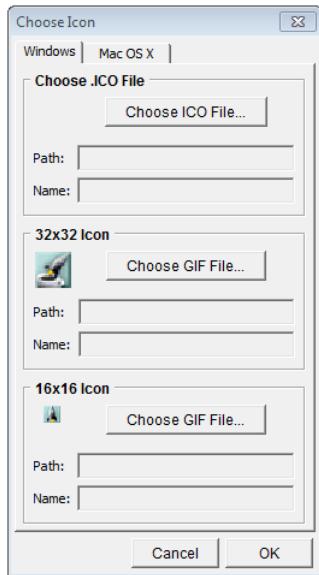


5. To customize the appearance the launcher will have as a shortcut, highlight the **OfficeSuite** launcher. The customizer along the lower portion of the Advanced Designer screen will change to reflect the options for the **Create LaunchAnywhere for Java Application** action.



In the lower right of the customizer (below the **Arguments** field) are a set of buttons that control the icon associated with the launcher. By default, a 32x32 pixel coffee cup icon and a 16x16 pixel rocket ship icon are chosen.

- Click **Change** to alter the icon. The **Choose Icon** dialog box opens.

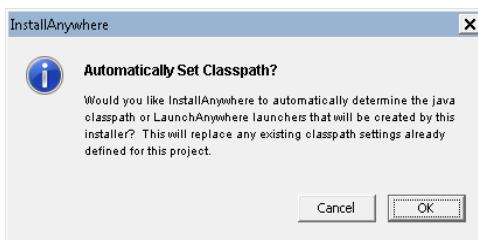


- On the **Windows** tab next to the **32x32 Icon**, click **Choose GIF File...**.
- Select `officeIcon.gif`, which is located in the `OfficeSuiteSourceFiles\Images` and `Docs` folder within the `InstallAnywhere` installation directory.



Note • Interlaced GIF files cannot be used with `InstallAnywhere`. The conversion process does not support these files and their use can result in blank icons. For Mac OS X, provide an ICNS file (created with `iconbuilder`—part of the Mac OS X Developer Tools). `InstallAnywhere` maintains a general classpath that is used to create launchers for the Java Application.

- For the **16x16 Icon**, select `officeIconSmall.gif` from the same directory.
- Click **OK** to close the **Choose Icon** dialog box. The new icons are now displayed in the **Create LaunchAnywhere for Java Application** customizer.
- Back on the **Install** task, click the **Set Classpath** button. You are prompted to specify whether you want to automatically set the classpath.

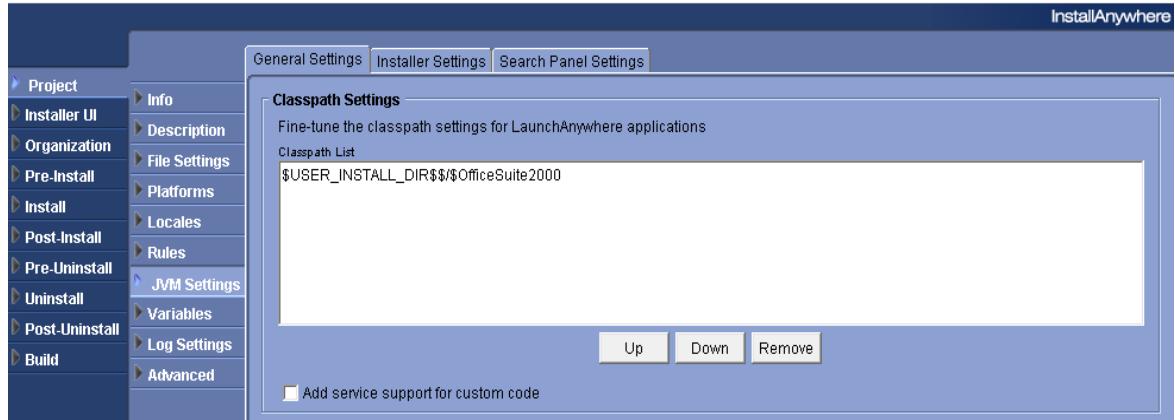


- Click **OK**. A blue CP icon appears on folders and archives that the process has added to the classpath.

Chapter 4: InstallAnywhere Tutorials

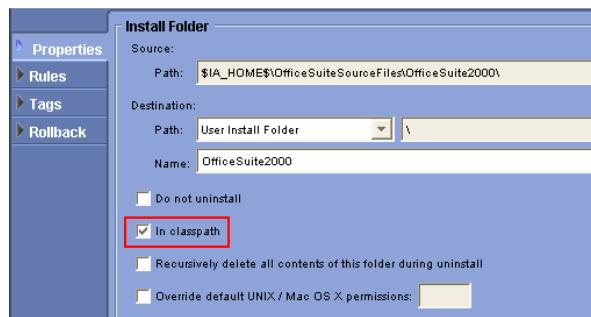
Building an Installer Using the Advanced Designer

13. To view the Classpath as determined by the Set Classpath action, open the **General Settings** tab of the **Project > JVM Settings** task.



Note the following regarding the **General Settings** tab of the **Project > JVM Settings** subtask:

- Since Office Suite is a simple product, only one main folder is listed: OfficeSuite2000 (which contains loose class files).
- If our example project contained JAR or ZIP files containing classes, they would also have been added to this list.
- If a file is added mistakenly to the **Classpath List**, it can be removed on this subtask or by selecting that file in the Visual Tree on the **Install** task and clearing the selection of the **In Classpath** check box in the customizer for that file.



14. Proceed with the steps in [Adding Post-Install Actions](#).

Adding Post-Install Actions

The **Post-Install** task list specifies actions and panels to occur after the installation of files. Like **Pre-Install**, the Post-Install step is ordered with the top actions occurring first. By default, InstallAnywhere has added two actions to the InstallAnywhere project: **Panel: Install Complete** and **Restart Windows**.



These actions are:

- **Panel: Install Complete**—This panel appears when the installation has completed successfully. This action is determined by the status of the \$INSTALL_SUCCESS\$ variable. This panel displays only if \$INSTALL_SUCCESS\$ contains no error conditions.
- **Restart Windows**—This action restarts a Windows system if the installer determines that it is necessary.

InstallAnywhere installations are controlled primarily by InstallAnywhere Rules. As an example of an InstallAnywhere Rule, highlight the **Restart Windows** action in the Office Suite Project. In the customizer in the lower portion of the screen, select the **Rules** tab. The **Rules** customizer will appear in the lower portion of the **Post-Install** task.

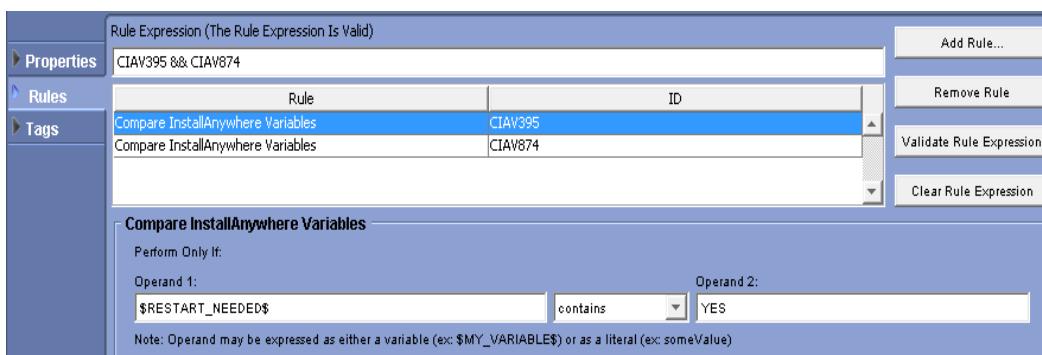


Figure 4-2: Restart Windows Action Rules Customizer

Chapter 4: InstallAnywhere Tutorials

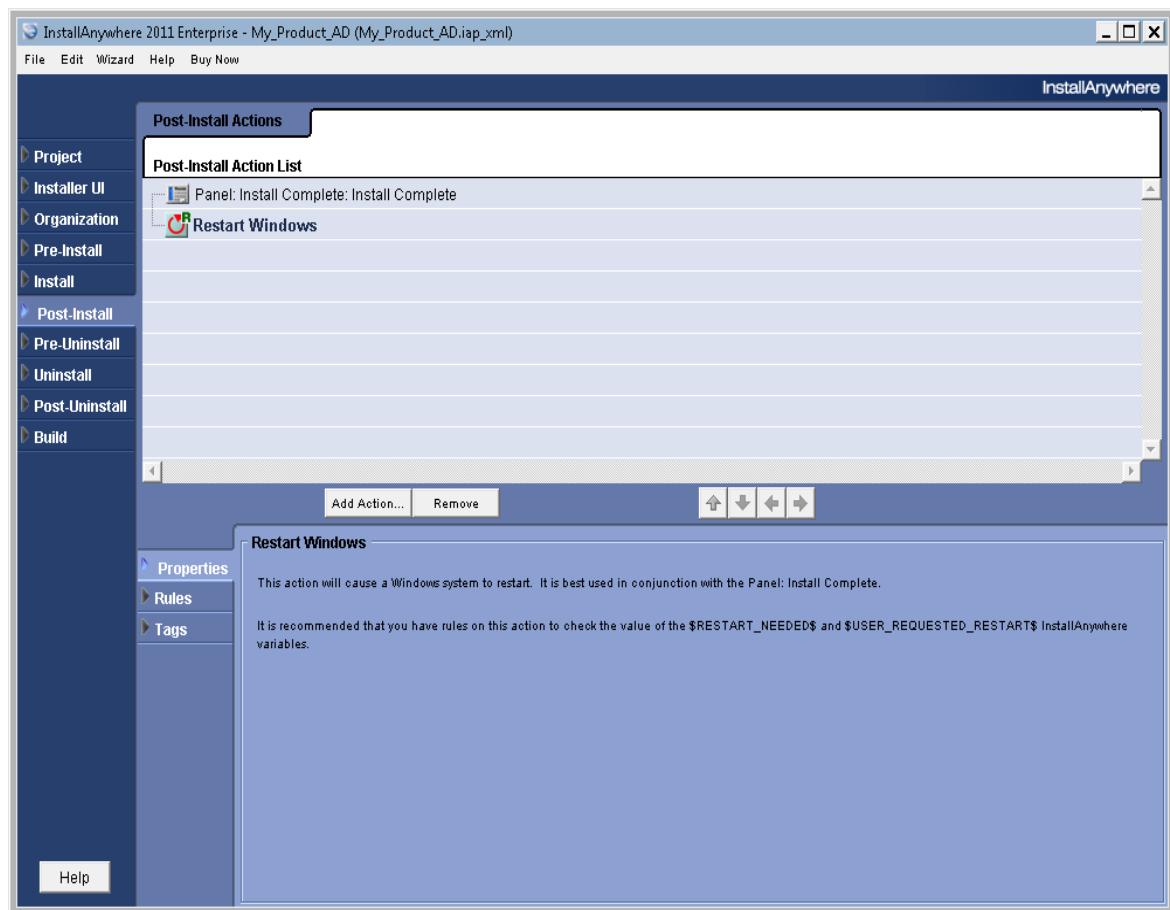
Building an Installer Using the Advanced Designer

The rules set on the **Restart Windows** action are rules set to compare InstallAnywhere variables. When a rule is added, the rule's unique ID (which is based on a combination of the abbreviation of the name of the rule and a numeric identifier) is displayed in the **Rule Expression** field. If you want to write complex rule expressions, you can edit the expression in the **Rule Expression** field to use multiple logical operators—such as AND (`&&`), OR (`||`), and NOT (`!`)—and precedence operators (parentheses) to express the relationship between two or more rules. By default, rules are joined by the AND operator. For more information, see [Building Complex Rule Expressions](#).

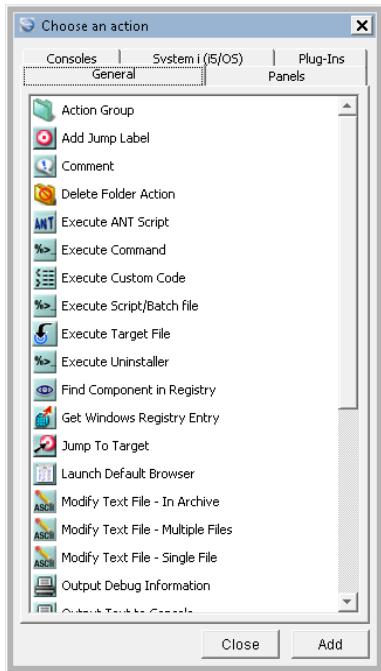


Task: **To add post-install actions:**

1. Perform the steps in [Adding a LaunchAnywhere Executable to the Install Task](#).
2. Open the **Post-Install** task.



3. Click the **Add Action** button to open the **Choose an Action** dialog box. The dialog box is divided by tabs that vary based on the task that is active at the time the **Add Action** button is clicked.



4. Select **Execute Target File** found under the **General** tab.

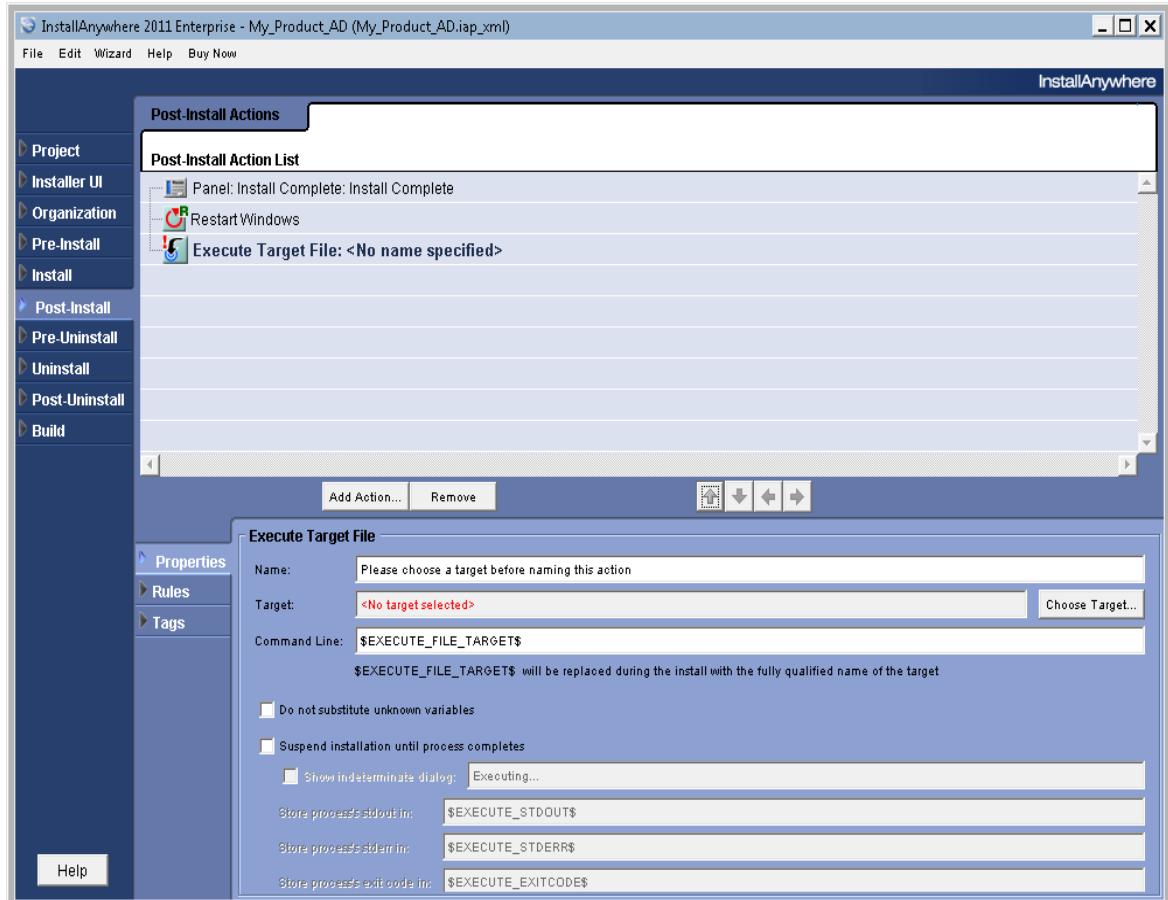


Note • The **Execute Target File** action is used to execute files that are included as part of the installation, and consequently it is available only in the **Install** and **Post-Install** tasks of the installation. The **Execute Target File** action is not available in the **Pre-Install** task because files cannot be executed that are not installed yet.

Chapter 4: InstallAnywhere Tutorials

Building an Installer Using the Advanced Designer

- Click **Add** or double click on the action to add it. The **Execute Target File** action is added to the **Post-Install Action List** and its customizer opens in the bottom of the screen.



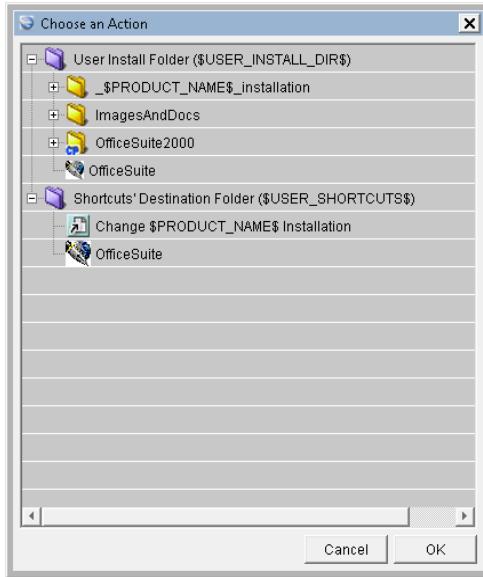
The **Choose an Action** dialog box will remain open so additional actions could be added.



Note • If the Execute action panel was added at a location other than the bottom of the **Post-Install** task, move it now. Either utilize the up and down arrows, or drag the action to the bottom of the task list.

- In the **Name** field of the **Properties** tab of the **Execute Target File** customizer, enter a name for the action. Naming the action will help identify the action when looking at the visual tree.

7. To select the target, click the **Choose Target** button next to the **Target** field. The **Choose an Action** dialog box opens and displays the file installation tree specified in the **Install** task.



8. File(s) can be executed in this stage. To execute the just installed Office Suite application, select the **OfficeSuite** icon under the **User Install Folder** (not the icon listed under the **Shortcuts Destination Folder**).



Note • You need to choose the actual *OfficeSuite launcher*, and not the shortcut, because shortcuts—especially on Windows and Mac OS systems—are pointers and are not inherently executable. *InstallAnywhere* will not execute a shortcut.



Important • By using the **Command Line** field, modifications can be made to the command line used to execute the file, such as adding a handler, or an argument to the execution. Do not remove or modify the **\$EXECUTE_FILE_TARGET\$** entry, as this represents the file to execute. To specify a handler, prepend an executable path, to specify an argument, append a file path. These paths must be absolute, however the paths can include *InstallAnywhere* variables.



Note • The user experience for the **Execute Target File** action can be tailored by using the **Suspend the installation until the process completes** option on the **Properties** tab of the **Execute Target File** customizer. This option is particularly useful in cases where a later step in the installation is dependant on the execution. You can also select a sub-option, **Show indeterminate dialog**, to specify an indeterminate progress bar with a message. This option can be used if the execution may take some time (for example, an execute that installs another product, or configures a database or other application).

9. Continue with the steps in [Building the Installer](#).

Chapter 4: InstallAnywhere Tutorials

Building an Installer Using the Advanced Designer

Building the Installer

The InstallAnywhere Build Task allows the options that will be used to build the installer(s) to be set. In this task platforms for the build can be set, configuration options for bundled virtual machines, and platform optimization and installer type.



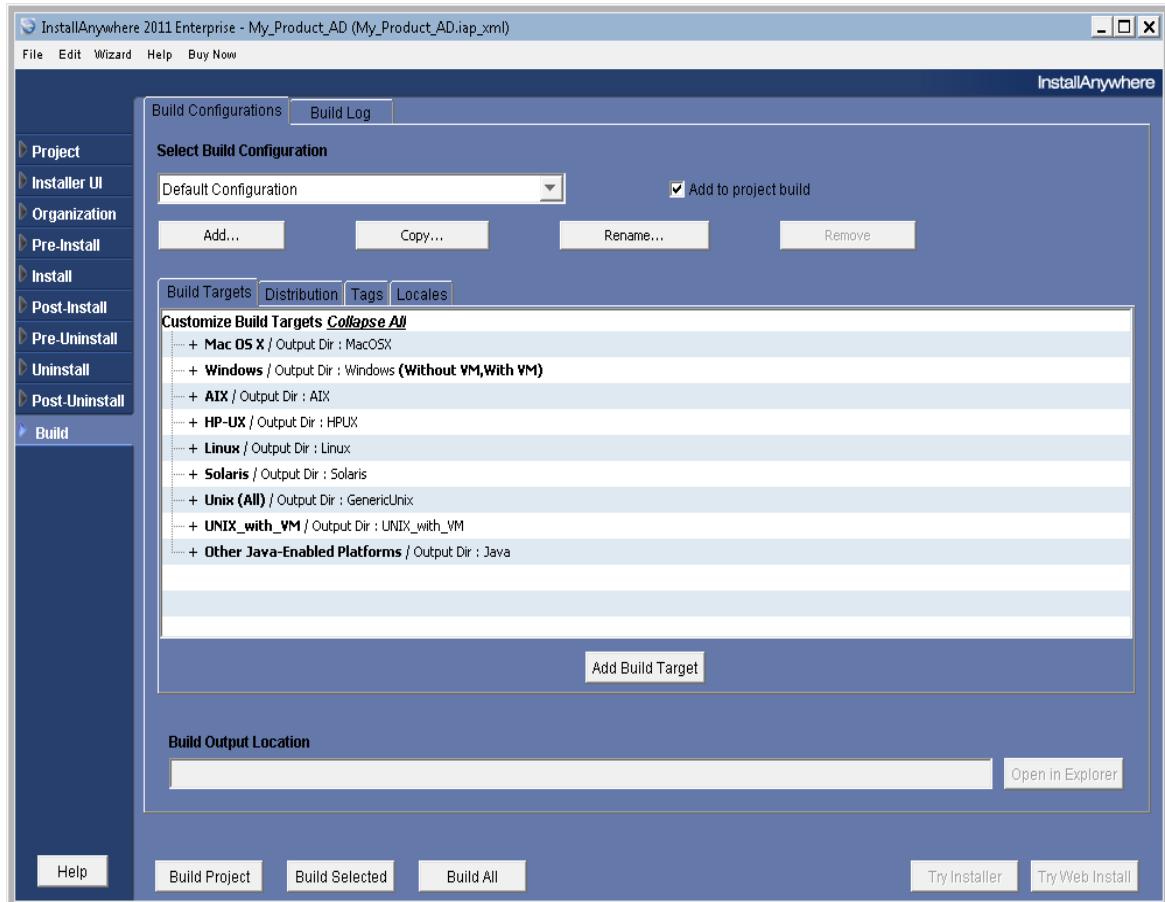
Tip • For early testing, build only for the development platform. Each additional platform adds to the time required to build, cycling through run-rebuild-run-rebuild stages. A faster build makes the development process easier.



Task:

To build the installer:

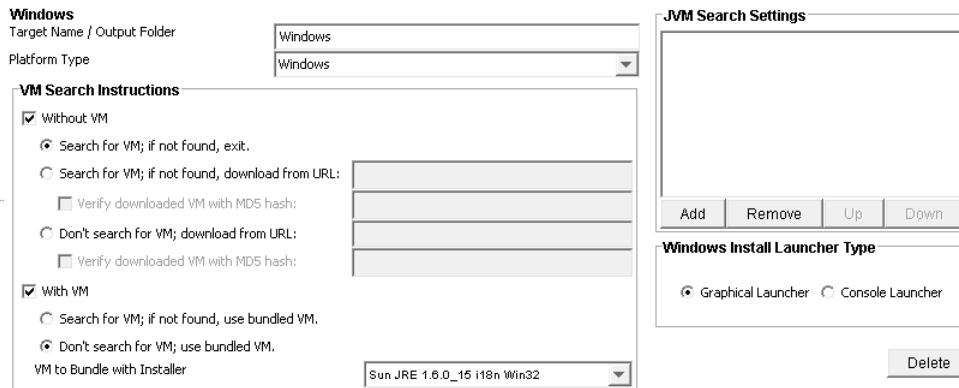
1. Perform the steps in [Adding Post-Install Actions](#).
2. Open the **Build** task of the Advanced Designer.



3. Set Select Build Configuration to Default Configuration.

Each InstallAnywhere project can have multiple Build Configurations, each representing how the installer will be built for particular set of platforms, files, build distributions, JVMs, and other settings. You can create and modify Build Configurations on the **Build Configurations** tab. For more information on Build Configurations, see [Creating and Editing Build Configurations](#).

4. On the Build Targets tab, click the plus sign next to Windows to display the Windows platform customizer.



5. Select the Without VM checkbox, and under it select Search for VM; if not found exit.

6. Select the With VM checkbox, and under it select Don't search for VM; use bundled VM. Because you selected **With VM**, InstallAnywhere will bundle the installer with a VM.

- **With VM** is only selectable for platforms which have a VM pack.
- VM packs should be placed in the [InstallAnywhere]/resource /installer_vms folder.
- InstallAnywhere should be restarted to refresh the available VM packs.

The **Build Configurations** tab of the **Build** task also includes three additional subtabs:

- **Distribution**—Use to set options for the type of installers to build, and the optimization options for each installer. If the installer did include platform specific files, these files would be optimized based on the application of the **Check Platform** rules. For more information, see [Distribution Subtab](#).
- **Tags**—You can use Tags to bundle different sets of actions, panels, features, and components into Build Configurations. After you define a set of Tags for a project, you then assign an appropriate Tag to all of those project elements that you want to include in some Build Configurations but exclude from others. Then, on the **Tags** subtab of the **Build Configurations** tab, you specify which Tags you want to include in the selected Build Configuration and which Tags you want to exclude from it. For more information, see [Tags Subtab](#).
- **Locales**—Use the **Locales** subtab to define the languages to include in each Build Configuration. For more information, see [Locales Subtab](#).

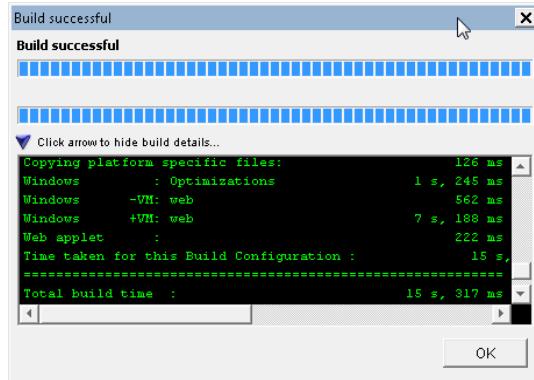


Note • The **Build Log** tab of the **Build** task displays the XML log of previous builds.

Chapter 4: InstallAnywhere Tutorials

Building an Installer Using the Advanced Designer

7. Click **Build Project** to build the OfficeSuite installer. The **Building** information dialog will appear.
8. Click the blue arrow on the lower left of that dialog to see the **Build Details**.



When the build is complete, **Build successful** will appear. For this tutorial, the build should take a minute or less.

9. Click **OK** to close the **Build** dialog box.
10. Proceed with the steps in [Testing the Installer](#)

Testing the Installer

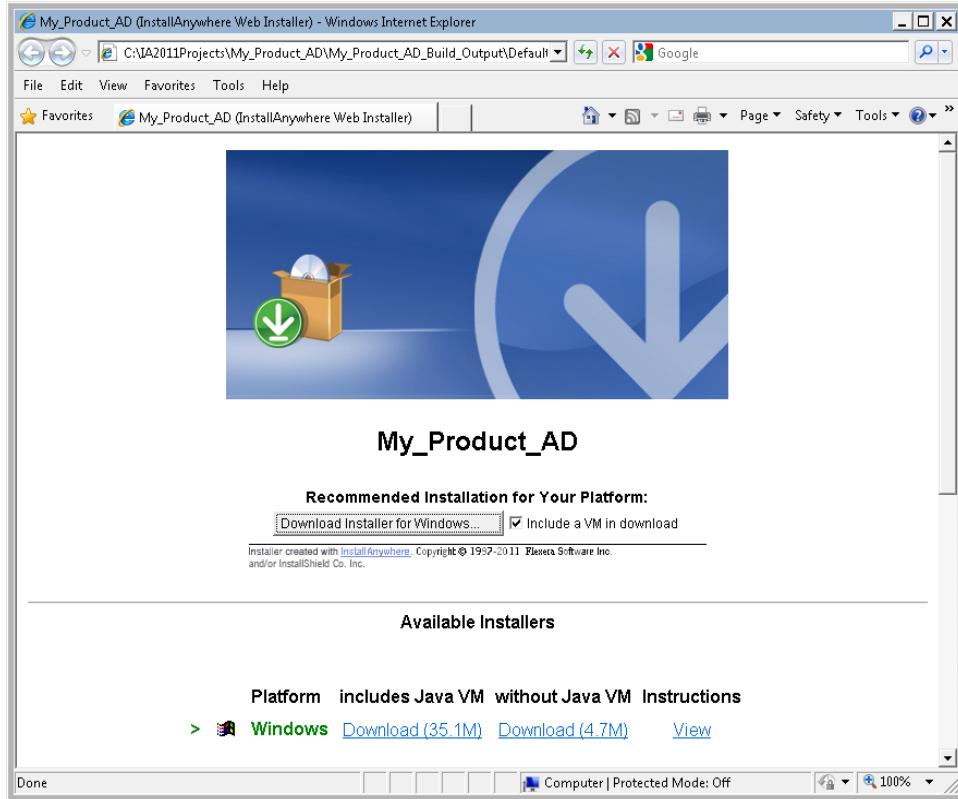
After the Build process is complete, try the installer by selecting either the **Try Web Install** or **Try Installer** button. In this case, use the **Try Web Install** button to launch our browser and the InstallAnywhere Web Install Page generated by the build process.



Task: *To test the installer:*

1. Perform the steps in [Building the Installer](#).
2. On the **Build** task, click **Try Web Install**. The Web Install Page will load, and should request a security access.

- Grant this access to allow the Web Install Applet to run the InstallAnywhere installer.



- Click the **Includes Java VM** download link below the image to download the installer. The download will begin.
- When the download is complete, run the installer, accepting the default values.
- On the **Install Complete** panel, click **Done**. The installer will close and OfficeSuite will be launched.



Note • You can also launch OfficeSuite on Windows by selecting it from the Windows Start menu.

Chapter 4: InstallAnywhere Tutorials

Building an Installer Using the Advanced Designer

Creating Basic Installers

This section provides procedures and suggestions about common situations when using InstallAnywhere to build a basic installer.

Table 5-1 • Creating Basic Installers Tasks

Section	Description
Creating a Basic Installation Project	Includes topics about some of the most common InstallAnywhere tasks: Naming a project, opening a project, and adding actions.
Organizing Features and Components	Describes various tasks and best practices related to using components and features in InstallAnywhere.
Defining Rules	Describes how to define rules for InstallAnywhere installers, specific actions, and groups of actions.
Implementing Maintenance Mode	Explains how to implement Maintenance Mode in an installer so that end users are able to add or remove features to previously installed products as well as repair broken installations.
Customizing the Uninstaller	Describes how to customize the Uninstaller by adding, removing or changing some of the uninstall actions.
Building Installers	Shows how to complete basic installer building tasks.
Working with VM Packs	Discusses how to download, install (for the InstallAnywhere IDE), and bundle VM packs.
Creating Launchers for Java Applications	Describes the process of creating a LaunchAnywhere launcher for any Java software your installer deploys.

Chapter 5: Creating Basic Installers

Creating a Basic Installation Project

Table 5-1 • Creating Basic Installers Tasks (cont.)

Section	Description
Improving Installation Performance	Identifies some optimization strategies for InstallAnywhere installers.
Launching Additional Installers	Explains how to launch another installer from your installer.
Installing Fonts	Shows how to use the \$FONTS\$ magic folder to install fonts on a target machine.
Team Development	Discusses tasks relevant to team development environments, especially source paths and source control management.
Referencing Developer Installation Manifests (DIMs)	Addresses tasks related to adding, customizing, and removing DIM references, as well as creating a shortcut to a file in a DIM reference.
Using FlexNet Connect	Describes how to obtain updates for InstallAnywhere.

Creating a Basic Installation Project

The following basic processes describe some of the most routine tasks for InstallAnywhere users.

Table 5-2 • Creating a Basic Installation Project Tasks

Topic	Description
Opening a Project	Describes the process of creating and naming a new project file.
Customizing the Pre-Install Panels and Actions	Explains the basic process of adding panels and actions in the Pre-Install task.
Defining the Install Task	Explains how to specify the install actions that take place during the step when InstallAnywhere deploys files to the target system.
Customizing the Post-Install Task	Explains how to specify actions and panels that will occur after the installation of files.



Note • See [Building an Installer Using the Project Wizard](#) to walk-through the process of creating your first project using the Project Wizard interface.

Opening a Project

You can either open an existing project or create a new project, using either the Project Wizard or Advanced Designer interface.

- [Creating a New Project](#)
- [Opening an Existing Project](#)

Creating a New Project

You can create a new project either in the Project Wizard or the Advanced Designer.



Task: *To name a project*

1. On the **New Project** or **Create New Project** dialog box, click **Save As**. The **Save New Project As** dialog box opens, which has options for naming a project and creating a new directory.
2. Enter a name for the new project in the **File Name** box.
3. In the **Path** field, specify the location where you want to save this new project.
4. Click **Save**.

Opening an Existing Project

You can open an existing project from either the Project Wizard or the Advanced Designer.

Opening an Existing Project from the Project Wizard

In the Project Wizard, the first dialog box allows you to either create a new project or open an existing project.



Task: *To open an existing project from the Project Wizard*

1. On the **New Project** dialog box, select the **Open Existing Project** option.
2. Select the project from the **Open Recent Project** list or click **Other Project** to select a different project.
3. Do one of the following:
 - To open the project in the Project Wizard, click **Next**.
 - To open the project in the Advanced Designer, click **Advanced Designer**.

Opening an Existing Project from the Advanced Designer

In the Advanced Designer, you can open an existing project using **Open** on the **File** menu.



Task: *To open an existing project from the Advanced Designer:*

1. On the **File** menu, click **Open**. The **Open Project File** dialog box opens.
2. Locate and select the project file you want to open.
3. Click **Open**.

Customizing the Pre-Install Panels and Actions

The **Pre-Install** task sets the panels and actions that occur prior to the installation of files. By default, a new InstallAnywhere project contains the following panels:

Table 5-3 • Pre-Install Task / Default Panels

Panel	Description
Introduction	This panel allows developers to introduce the product or installation process.
Choose Install Folder	This panel allows end users to choose the installation location for the product.
Choose Alias, Link Shortcut Folder	This panel allows end users to specify the location for any Mac OS X Aliases, Windows Shortcuts, and Unix Symlinks (used as shortcuts) that will be installed.
Pre-Install Summary	This panel provides end users with a summary of various installation settings prior to the installation of files.

To customize the Pre-Install task panels and actions, perform the following steps.



Task: *To customize the Pre-Install task panels and actions:*

1. Open a project in the Advanced Designer and open the **Pre-Install** task.
2. Click **Add Action**. The **Choose an Action** dialog box opens and displays five tabs: General, Panels, Console, System i (i5/OS), and Plug-Ins.

3. In the **Choose an Action** dialog box, click the action you want to add. The following types of actions are available on the **Pre-Install** task:

Action Type	Description
General Actions	Available in the Install task, as well as the pre- and post-install/uninstall tasks, these actions are typically transparent to the end user and require no end user interaction.
Panel Actions	Available in the Pre-Install, Post-Install, Pre-Uninstall, and Post-Uninstall tasks, these actions add panels to your installer for the purposes of communicating with the end user and acquiring end user input.
Console Actions	Similar to Panel actions, these actions communicate with end users and acquire end user input for installers that use a console interface.
System i (i5/OS) Actions	Consolidated in a single tab, these actions include general, install, panel, and console actions specifically for i5/OS installers.
Plug-In Actions	Developers can integrate properly packaged custom code into the design environment as plug-ins.

4. Click **Add**. InstallAnywhere adds the action to the Pre-Install Action List, but the **Choose an Action** dialog box remains open.
5. Use the arrow keys to move the action up or down in the Action List.
6. When you have completed adding actions, click **Close**. The actions you have added are now listed in the **Action** list.
7. Select each action and use its customizers to specify the action settings. See [Actions](#) for detailed information on each action's customizer settings.

Defining the Install Task

The **Install** task contains the install actions that take place during the step when InstallAnywhere deploys files to the target system. On the **Install** task, you define the files to install, the folder location to install those files, the order of the tasks that need to happen as the files are being installed, and the install actions that take place when InstallAnywhere deploys files to the target system.

The **Install** task displays files and directories in the way the installer will deploy them on the target system. You can use the controls on the **Install** task to assign files, directories, and actions to either components or product features.

Information about defining the **Install** task is presented in the following sections:

- [Starting to Define the Install Task](#)
- [Adding Files to a Project](#)

Chapter 5: Creating Basic Installers

Creating a Basic Installation Project

- Adding Install Actions
- Adding LaunchAnywhere Executables to the Install Task
- Recreating the InstallAnywhere Uninstaller and Associated Components

Starting to Define the Install Task

To get started defining the **Install** task, perform the following steps:



Task: To get started defining the **Install** task:

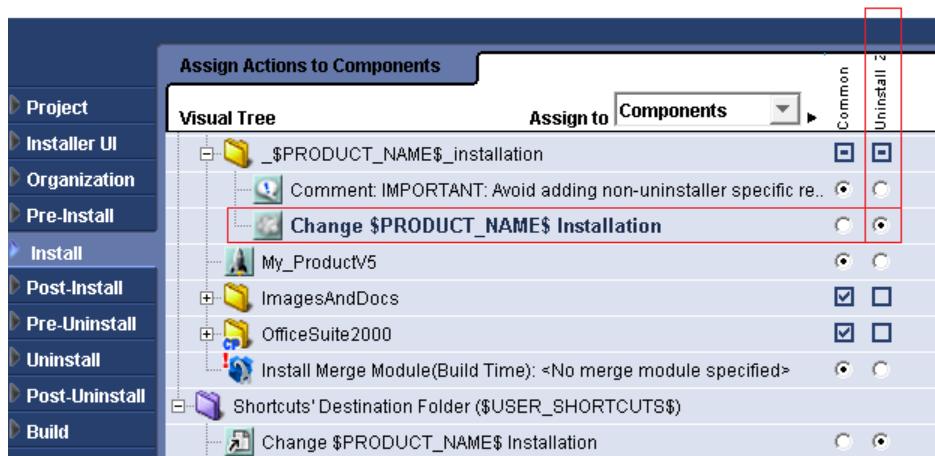
1. Open a project in the Advanced Designer.
2. Open the **Install** task. By default, the following items are listed in the Visual Tree:



- **\$PRODUCT_NAME\$_installation**—By default, this folder contains any InstallAnywhere uninstaller/Maintenance Mode actions, and a comment action with instructions pertaining to the uninstaller. For organizational purposes, it is generally best to have the Uninstaller creation action first. Leave the Uninstaller creation in its default place in the installation (although the folder structure can be changed).
- **Shortcuts' Destination Folder (\$USER_SHORTCUTS\$)**—This magic folder is the directory specified by the end-user as the Alias, Link, Shortcut folder location. The value of this location can be changed by the end-user if the **Choose Alias, Link, Shortcut Folder** panel action is turned on in the **Pre-Install** task of the installer.



Important • By default, the Uninstall Launcher, **Change \$PRODUCT_NAME\$ Installation**, is created in the **\$PRODUCT_NAME\$_installation** folder of the **Install** phase. Make sure that the Uninstall Launcher is associated with the InstallAnywhere **Uninstall Component** (as shown below). If the Uninstall Launcher is not associated with this component (which is automatically created and is listed on the **Organization > Components** subtask), the project build will always fail and the user will be prompted to **Associate the Uninstall Launcher to the Uninstaller Component**.



3. Add files to the project, as described in [Adding Files to a Project](#).
4. Add actions to the **Install** task, as described in [Adding Install Actions](#).



Tip • The Advanced Designer implements a drag-and-drop interface in many tabs and tasks. In the **Install** task, actions and files can be moved by selecting and dragging them. A dark underline appears in the location where the file or action will be placed.

5. Organize features and components, as described in [Organizing Features and Components](#).
6. Save the project.

Adding Files to a Project

To add files and directories to a project, perform the following steps.



Task:

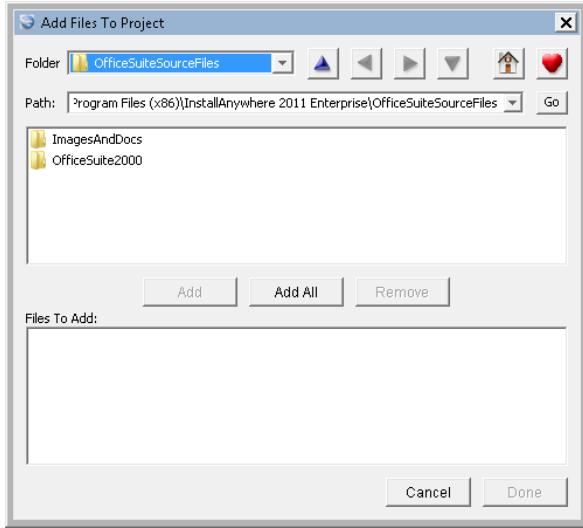
To add files to a project:

1. Open a project in the Advanced Designer interface.
2. Open the **Install** task.

Chapter 5: Creating Basic Installers

Creating a Basic Installation Project

3. Click **Add Files**. The **Add Files to Project** dialog box opens.



4. Browse to directory containing the files you want to add.
5. Use the **Add** button (to add one file) or the **Add All** button (to add all files and folders in the selected directory) to add the desired files and folders to the project. These files and folders are now listed in the **Files to Add** list.
6. Click **Done**. The selected files and folders are now listed in the **Visual Tree**.



Note • Objects may be moved up and down or into and out of subfolders in the file tree by highlighting the object, and using the right, left, up, and down arrows.

7. Select each new item in the Visual Tree and set options on its customizer:
 - **Properties tab**—See the [Create Folder](#) and [Install File](#) actions.
 - **Rules tab**—See [Assigning a Rule to an Action](#).
 - **Tags tab**—See [Using Tags to Customize Build Configurations](#).
 - **Rollback tab**—See [Setting Installation Rollback Options](#).

Adding Install Actions

Just like the **Pre-Install/Uninstall** and **Post-Install/Uninstall** tasks, you may add actions directly into the file and folder hierarchy that is shown in the **Install** task. Actions are task sensitive, so the actions available from the **Install** task are different than those available from the other tasks. The actions available in the **Install** task create folders and aliases, copy or delete files and folders, expand folders, or add functionality—like creating LaunchAnywhere launchers or adding platform specific actions.

To add Install actions to a project, perform the following steps.



Task: *To add Install actions:*

1. Open a project in the Advanced Designer and open the **Install** task.
2. Click **Add Action**. The **Choose an Action** dialog box opens and displays four tabs: Install, General, System i (i5/OS), and Plug-Ins.
3. In the **Choose an Action** dialog box, select the action you want to add. The following types of actions are available on the **Install** task:

Action Type	Description
Install Actions	Available only in the Install task, Install actions deploy the installer payload to the target machine.
General Actions	These actions are typically transparent to the end user and require no end user interaction.
System i (i5/OS) Actions	Consolidated in a single tab, these actions include general, install, panel, and console actions specifically for i5/OS installers.
Plug-In Actions	Developers can integrate properly packaged custom code into the design environment as plug-ins.

4. Click **Add**. InstallAnywhere adds the action to the Visual Tree, but the **Choose an Action** dialog box remains open.
5. When you have completed adding actions, click **Close** to close the **Choose an Action** dialog box.
6. Use the arrow keys to move the action up or down in the Visual Tree.
7. Select each action and use its customizers to specify the action settings. See [Actions](#) for detailed information on each action's customizer settings.

Adding LaunchAnywhere Executables to the Install Task

A LaunchAnywhere Executable (LAX) is a unique native executable, created by InstallAnywhere, that is used to launch a Java application. While the InstallAnywhere Wizard specifically asks to select a main class and automatically creates a single launcher, the Advanced Designer allows developers to add as many launchers as they would like. There are two ways to add a LaunchAnywhere Launcher to an InstallAnywhere project file from the **Install** task:

- Click **Add Action** and select **Create LaunchAnywhere for Java Application** from the **Choose an Action** dialog box
- Click the **Add Launcher** button on the middle control bar in the Advanced Designer.

Chapter 5: Creating Basic Installers

Creating a Basic Installation Project

To add a LaunchAnywhere executable to a project, perform the following steps.



Task: **To add a LaunchAnywhere executable:**

1. Open a project in the Advanced Designer and open the **Install** task.
2. Highlight the **User Install Folder** in the Visual Tree, and click the **Add Launcher** button. You are prompted to automatically find classes with main methods.
3. Click **OK**. The Choose a main class dialog box opens with a class listed.



Note • When adding a launcher, InstallAnywhere automatically inspects the added files (including introspecting into JAR and ZIP files) to find class files with Main Methods specified.

4. Choose one of the listed items as the main class for the application and click **OK**.

By clicking the **Add Launcher** button, you have not only added the launcher to the file structure, but also created a Shortcut, Link, or Alias action in the **Shortcuts' Destination Folder** Magic Folder. This location is variable and will be specified by the **Choose Alias, Link, and Shortcut** panel in the pre-install section.

5. To customize the appearance the launcher will have as a shortcut, highlight the launcher. The customizer along the lower portion of the Advanced Designer screen will change to reflect the options for that **Create LaunchAnywhere for Java Application** action.

In the lower right of the customizer (below the **Arguments** field) are a set of buttons that control the icon associated with the launcher. By default, a 32x32 pixel coffee cup icon and a 16x16 pixel rocket ship icon are chosen.

6. If you want to change the icon used for the launcher, click **Change**. The **Choose Icon** dialog box opens.
7. Specify a 32x32 and a 16x16 icon to use for the launcher.
8. Click **OK** to close the **Choose Icon** dialog box. The new icons are now displayed in the **Create LaunchAnywhere for Java Application** customizer.
9. Back on the **Install** task, click the **Set Classpath** button. You are prompted to specify whether you want to automatically set the classpath.
10. Click **OK**. A blue CP icon appears on folders and archives that the process has added to the classpath.



Note • To view the Classpath as determined by the Set Classpath action, open the **Project** task and then open the **General Settings** tab of the **JVM Settings** subtask.

Recreating the InstallAnywhere Uninstaller and Associated Components

When a new InstallAnywhere project is created, an **InstallAnywhere Uninstall Component** is automatically created and is displayed on the **Organization > Components** task.

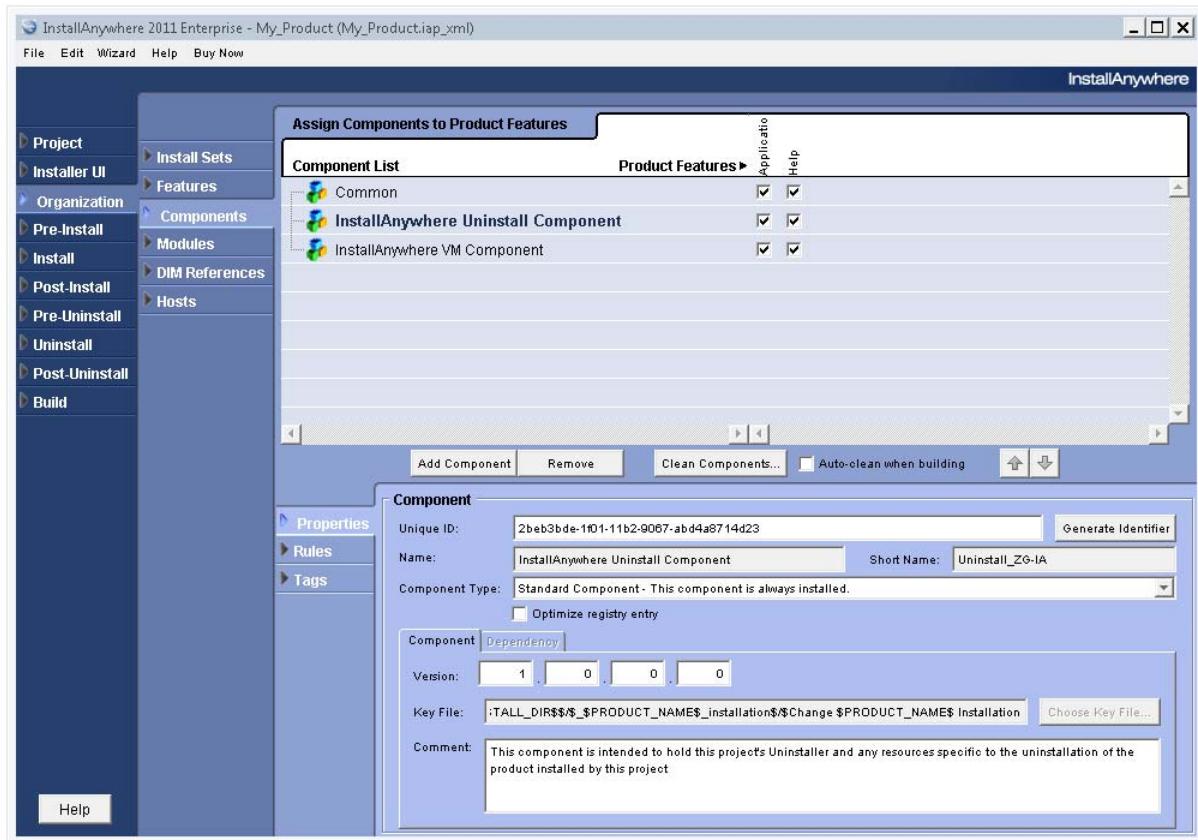


Figure 5-1: InstallAnywhere Uninstall Component in Organization > Components Task

By default, the component's name is **InstallAnywhere Uninstall Component** and its short name is **Uninstall_ZG-IA**, both of which are not editable on this task.

If you accidentally delete the InstallAnywhere Uninstall Component and want to recreate it, perform the following steps:



Task:

To recreate the *InstallAnywhere Uninstall Component*:

1. In the **Install** phase, add a **Create Uninstaller** action to the project, if one is not already present.
2. On the **Organization > Components** task, click **Add Component**. A new, untitled component is added to the **Components List**.
3. In the **Component** customizer, enter **InstallAnywhere Uninstall Component** in the **Name** field.

The name **Install** is automatically entered in the **Short Name** field, which becomes read only.

Chapter 5: Creating Basic Installers

Creating a Basic Installation Project

4. Open the **Install** phase and select **Components** from the **Assign to** list.
5. Select the **Change \$PRODUCT_NAME\$ Installation** item in the **Visual Tree** and select the **Install** option in the **Components** list.
6. Return to the **Organization > Components** task and select the **InstallAnywhere Uninstall Component** in the **Component List**.
7. In the **Component** customizer, click **Choose Key File** and select the location of the uninstaller: `$_PRODUCT_NAME$_installation`.
8. Save and close the project and exit InstallAnywhere.
9. Open this InstallAnywhere project in a text editor and locate the `object` named `com.zerog.ia.installer.InstallComponent` for which the `componentName` property is set to **InstallAnywhere Uninstall Component**.
10. Set the `shortName` property of this component from `Install` to `Uninstall_ZG-IA`.
11. Save the project.
12. Open the project in the InstallAnywhere Advanced Designer.
13. Open the **Organization > Components** task. You will see that the **Short Name** field for the **InstallAnywhere Uninstall Component** is now set to `Uninstall_ZG-IA`.

Customizing the Post-Install Task

The **Post-Install** task list specifies actions and panels to occur after the installation of files. Like **Pre-Install**, the Post-Install step is ordered with the top actions occurring first. By default, InstallAnywhere adds two actions to the InstallAnywhere project: **Panel: Install Complete** and **Restart Windows**.



These actions are:

- **Panel: Install Complete**—This panel appears when the installation has completed successfully. This action is determined by the status of the `$INSTALL_SUCCESS$` variable. This panel displays only if `$INSTALL_SUCCESS$` contains no error conditions.
- **Restart Windows**—This action restarts a Windows system if the installer determines that it is necessary.

InstallAnywhere installations are controlled primarily by InstallAnywhere Rules. As an example of an InstallAnywhere Rule, highlight the **Restart Windows** action in the Office Suite Project. In the customizer in the lower portion of the screen, select the **Rules** tab. The **Rules** customizer will appear in the lower portion of the **Post-Install** task.

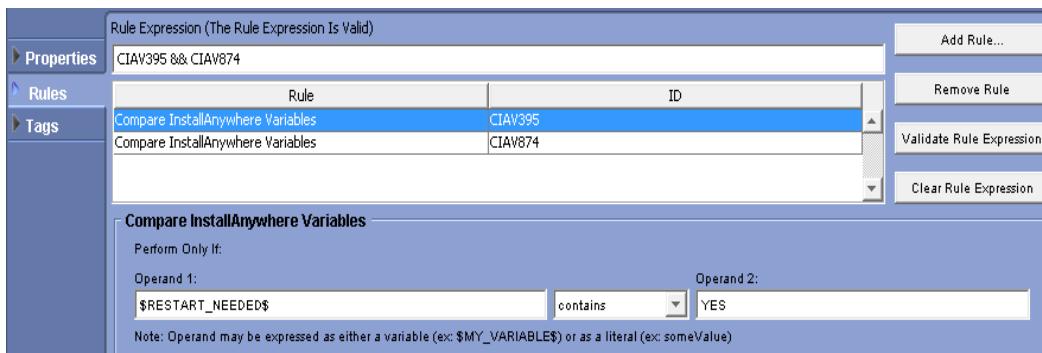


Figure 5-2: Restart Windows Action Rules Customizer

The rules set on the Restart Windows action are simple rules set to compare InstallAnywhere variables. InstallAnywhere Rules are Boolean and allow the file, panel, or action to be installed, displayed, or run only if the rule resolves to True.



Task:

To add post-install actions:

1. Open a project in the Advanced Designer and open the **Post-Install** task.
2. In the **Choose an Action** dialog box, click the action you want to add. The following types of actions are available:

Action Type	Description
General Actions	Available in the Install task, as well as the pre- and post-install/uninstall tasks, these actions are typically transparent to the end user and require no end user interaction.
Panel Actions	Available in the Pre-Install, Post-Install, Pre-Uninstall, and Post-Uninstall tasks, these actions add panels to your installer for the purposes of communicating with the end user and acquiring end user input.
Console Actions	Similar to Panel actions, these actions communicate with end users and acquire end user input for installers that use a console interface.
System i (i5/OS) Actions	Consolidated in a single tab, these actions include general, install, panel, and console actions specifically for i5/OS installers.
Plug-In Actions	Developers can integrate properly packaged custom code into the design environment as plug-ins.

3. Click **Add**. InstallAnywhere adds the action to the **Post-Install Action List**, but the **Choose an Action** dialog box remains open.
4. When you have completed adding actions, click **Close**. The actions you have added are now listed in the **Post-Install Action List**.
5. Use the arrow keys to move the actions up or down in the Action List.
6. Select each action and use its customizers to specify the action settings. See [Actions](#) for detailed information on each action's customizer settings.

Organizing Features and Components

Components are the lowest level of organization in an installer. Each product must have at least one component but most installers will by default contain at least two components, as the uninstaller is considered a component of its own.

InstallAnywhere's component architecture is designed to allow developers to plan for future releases, suite installers, and other uses of their software elements in their deployment plan. From your perspective as the installation developer, components are the building blocks of applications or features. End users never see or interact with components.

InstallAnywhere automatically creates components as you add files to your project and assign them to Features. This approach, while working well for most projects, does not give you the most flexibility. To realize the ultimate benefits of componentized software, you should manually manage the creation of components.

Table 5-4 • Organizing Features and Components Tasks

Topic	Description
Best Practices for Components	Includes some pointers about working with components.
Adding Components	Describes the process of adding components on the Organization > Components subtask.
Assigning Files to Components	Explains how to assign files to components on the Install task.
Removing Empty Components	Explains how to remove components on the Organization > Components subtask.
Integrating Components Already Installed on Target System	Discusses the integration of existing components with the Find Component in Registry action.
Adding Features	Describes the process of adding features on the Organization > Features subtask.
Assigning Components to Features	Explains how to assign components to features.

Best Practices for Components

When using components, first determine and organize which components to add. When doing this, keep the following in mind:

- **Make unique components for files that will need to be updated separately.** For example, a “Help” feature may have both a User Guide and Javadocs. However, the User Guide may be updated more frequently than the Javadocs. Make the two items separate components so a unique “User Guide” component may be added which can be versioned and updated individually.
- **Components should make logical sense.** When building a Suite Installer, keep in mind the pieces of applications that are shared between different products. When componentizing a product for versioning purposes, designate the version of the component in the **Organization > Component > Properties** task when the component is added.

Adding Components

You add components to a project in the **Organization > Components** task. In the **Components** task, you add and remove components, set component properties, and associate components with features.

You can use shared components and component dependencies to create installers that leverage external components, either developed in-house or by a third party, as part of your software distribution strategy. These components can be included as part of a suite installer, be defined as prerequisite dependencies for your package, and can even enable multiple applications to share common components across a system.



Task:

To add a component to an installer:

1. Open a project in the Advanced Designer and open the **Organization > Components** subtask.
2. Click **Add Component**. An untitled component is added to the **Component List** and that component's customizer is opened.
3. On the **Properties** tab of the **Component** customizer, specify the following controls:

Control	Description
Unique ID	<p>When you add a new component, an alphanumeric string is automatically entered to uniquely identify this component. Click Generate Identifier to generate a new ID.</p> <p>The Unique ID value is a UUID that must be different from the UUID of any other component, except for a different version of the same component.</p>  <p>Note • The Find Component in Registry action enables you to detect a component based on its UUID and version.</p>
Name	Enter a name to identify this Component in the InstallAnywhere user interface.

Control	Description
Short Name	Enter a short reference name, which will be used to uniquely identify the Component in the Registry.
Component Type	<p>Select one of the following options to identify this component's type:</p> <ul style="list-style-type: none"> • Standard Component—This component will always be installed, regardless of previous installations or dependencies. The uninstaller for the product with which a standard component is installed will remove the standard components. • Shared Component—This component will be installed if it has not already been installed on the system. A shared component can be made available to other applications or installations. At uninstall, a shared component will be removed only if no other installed application references it. The uninstaller for the last product referencing the shared component will uninstall it. • Dependency—If a component is specified as a Dependency, instead of installing this component, the installer will require that this component has already been installed on the system. Dependencies are not elements of your install per se, but rather are prerequisite requirements that the installer will enforce.
Optimize registry entry	<p>Use this option to instruct InstallAnywhere how to update the global registry when installing a new version of a previously installed component.</p> <ul style="list-style-type: none"> • Selected—Only the latest version of the component is stored in the global registry. For example, if you install version 1.0 of Component A and then install version 2.0 of Component A, the global registry would only contain an entry for version 2.0 of Component A, the latest version. • Not selected—Both versions of the component are stored in the global registry. For example, if you install version 1.0 of Component A and then install version 2.0 of Component A, the global registry would contain two entries for Component A: one for version 1.0 and one for version 2.0.

4. On the **Component** subtab of the **Properties** tab, specify the following controls:

Control	Description
Version	Enter a 4-digit version.
Key File	<p>A key file is a single file that identifies the component. A key file should always be included in a component.</p> <p>You can enter the name and location of the key file or click Choose Key File to select one of the project files to be the key file.</p>  <p>Note • A component's key file must be present in all subsequent versions of the component. The key file is used to define the component's location when the Find Component in Registry action is used.</p>
Comment	Enter a comment to identify the purpose of this component.

5. On the **Dependency** subtab of the **Properties** tab, specify the following controls:

Control	Description
Matches Key File Location	Enter the location of the key file of the dependent component. A key file is a single file that identifies the component.
Version at least Version at most	Optionally, specify a version number range to make the dependency search more specific.
Matched Key File Location	This variable will contain where the dependency is installed.
Dependency State Variable	<p>You can use this variable to see if the search was successful.</p> <ul style="list-style-type: none"> • If the dependency check passes, the Dependency State Variable will be an empty string. • If the dependency check fails, the Dependency State Variable will contain the Dependency Failed Message.
Dependency Failed Message	If the dependency check fails, the Dependency State Variable will contain the message entered in this field.



Note • For more information about dependencies check out the Sample Dependencies Template that comes with InstallAnywhere. It is sample project that helps illustrate how dependencies work, and what variables are set.



Note • You can control when the installer searches for dependencies explicitly by using the **Evaluate Dependencies** action. See [Evaluate Dependencies](#).

6. On the **Rules** tab of the **Component** customizer, you can associate rules with the component. The rules for a component are evaluated before the component is installed. For more information, see [Assigning a Rule to an Action](#).
7. On the **Tags** tab of the **Component** customizer, you can associate **Build Configuration** Tags with the component. For more information, see [Assigning Tags to Project Elements](#).

Assigning Files to Components

To assign files to a component, use the **Assign to > Components** option in the **Install** task.



Task: *To assign a file to a component:*

1. Open a project in the Advanced Designer and open the **Install** task.
2. Select a file in the Visual Tree.
3. From the **Assign to** list above the Visual Tree, select **Components**. All of the project components are listed.
4. Select the component that you want to assign the selected file to.
5. Save the project.

Removing Empty Components

Sometimes you may have components that are no longer needed or do not have any files assigned to them. You can choose to automatically delete all empty components.



Task: *To automatically delete all empty components:*

1. Open a project in the Advanced Designer.
2. Open the **Organization > Components** subtask.
3. Click **Clean Components**. You will be prompted to confirm that you want to delete all components that have no actions assigned to them.
4. Click **OK**. The empty components are deleted.

Integrating Components Already Installed on Target System

If the installer needs a component that should already be installed on the target system, use the **Find Component in Registry** (referring to the InstallAnywhere registry and not the Windows registry) action to locate that component. This action searches for the component by using the UUID, the unique identifier specified for the component. The component's location can then be used as an installation location by the installer.

You can then request that only the highest version be found, that the installer compares versions, or search for the key file. The action sorts the count of components found, the versions found, and the locations found in variables you specify. The following figure shows the customizer for the **Find Component in Registry** action.

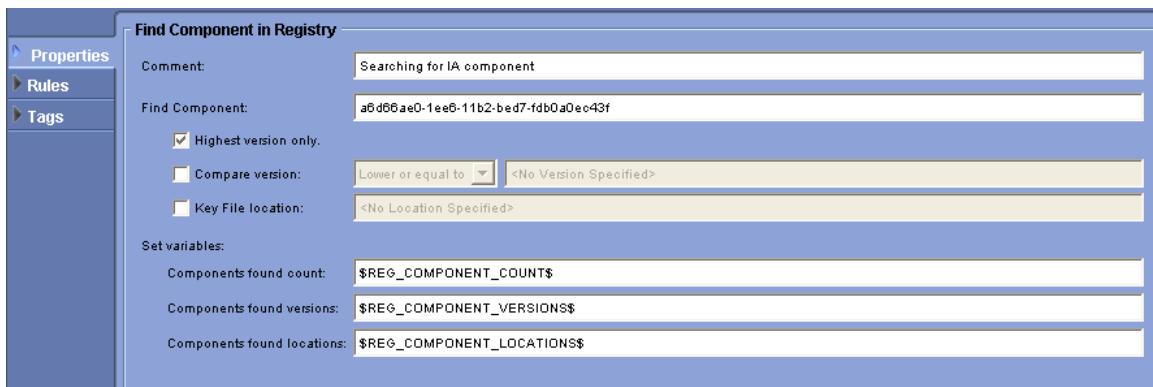


Figure 5-3: Find Component in Registry Action Customizer

After the action runs, you can use the resulting variables in subsequent actions and rules. For example, to display a panel only if a component is not found, you can add a **Display Message** panel with a rule that uses the value of \$REG_COMPONENT_COUNT\$.

Adding Features

Features are managed in the **Organization > Features** task. You can add and remove Features, as well as assign Features to Install Sets.



Task: *To add a feature to a project:*

1. Open a project in the Advanced Designer and open the **Organization > Features** subtask.
2. Click **Add Product Feature**. An untitled Feature is added to the **Feature Tree** and that feature's customizer is opened.
3. In the **Product Feature** customizer, specify the following controls:

Control	Description
Name	Enter a name to identify this Product Feature in the InstallAnywhere user interface.
Short Name	Enter a short reference name, which will be used to uniquely identify the Feature in the Registry.
Description	Enter a description of this Product Feature to identify its purpose.

4. In the **Install Sets** column, select the Install Set or Sets that you want the new Product Feature to belong to.
5. Save the project.

Assigning Files to Features

To assign a file to a feature, use the **Assign to > Product Features** option in the **Install** task.



Task: *To assign a file to a component:*

1. Open a project in the Advanced Designer and open the **Install** task.
2. Select a file in the Visual Tree.
3. From the **Assign to** list above the Visual Tree, select **Product Features**. All of the project features are listed.
4. Select the feature or features that you want to assign the selected file to. A file can belong to more than one feature.
5. Save the project.

Assigning Components to Features

You can manually assign Components to Features on the **Organization > Components** task, or you can assign Files to Features directly (see [Assigning Files to Features](#)), which creates the Feature to Component assignments automatically.



Tip • If you assign a file to a Feature that does not already have a Component, a Component will be created automatically for you.



Note • When manually managing Components, avoid assigning files directly to Features, as this will create and change Component assignments. Instead, make sure you assign files to Components yourself.



Task: [**To assign a component to a feature:**](#)

1. Open a project in the Advanced Designer.
2. Open the **Organization > Components** subtask. All project **Components** and **Product Features** are listed.
3. In the **Component List**, select a Component.
4. In the **Product Features** column, select the Product Feature or Features that you want the selected Component to belong to.
5. Save the project.

Defining Rules

Rules are conditions a setup author places on a project's installers, specific files or actions, or groups of actions. When you assign a rule to an action, for example, InstallAnywhere installers run that action only when the rules assigned to it return a true result.

Table 5-5 • Defining Rules Tasks

Topic	Description
Assigning a Rule to the Installer	Explains how to make the execution of the whole installer subject to one or more rules.
Assigning a Rule to an Action	Describes how to conditionalize the execution of any InstallAnywhere action.
Assigning a Rule to a Group of Actions	Shows how to use an Action Group to apply rules to a set of actions.
Building Complex Rule Expressions	Explains how to connect rules with logical ANDs and ORs to express more complex conditions.
Customizing Built-in Rules	Describes how to customize some of InstallAnywhere's built-in rules.
Rules Are Evaluated Multiple Times	Explains how during installation, rules are evaluated multiple times and at points in the sequence that are much earlier than where the rule is actually located. This may be important to consider because a rule could be referring to an InstallAnywhere variable that has not yet been set at the point when the rule is evaluated.

Assigning a Rule to the Installer

When you assign a rule to an installer, you are setting conditions for the entire installer. If the rules you assign do not evaluate to true, your installer does not run.



Task: To assign a rule to the installer

1. Navigate to the **Project > Rules** task. The rules you define here apply to all installer you build from this project.
2. Click **Add Rule**. InstallAnywhere opens the **Choose a Rule** dialog box.
3. Choose the rule you want to add for the installer and click **Add**. InstallAnywhere adds the rule to the Rules list and shows the corresponding rule customizer.
4. Customize the rule according to your install requirements.



Note • For descriptions of InstallAnywhere's built-in rules, see [Rules Reference](#).

Assigning a Rule to an Action

When you assign a rule to an action, the action only executes when that rule returns a true result.



Task: To assign a rule to an action

1. In the Advanced Designer, choose the action to which you want to assign a rule.
2. In the rule's customizer, click **Rules** to open the customizer's **Rules** tab.
3. On the **Rules** tab, click **Add Rule**. InstallAnywhere opens the **Choose a Rule** dialog box.
4. Choose the rule you want to add for the installer and click **Add**. InstallAnywhere adds the rule to the Rules list and shows the corresponding rule customizer.
5. Customize the rule according to your install requirements. See [Rules Reference](#) for detailed information.



Note • This procedure applies to all actions, files, and folders that appear in any of the Pre-Install, Post-Install, Install, Pre-Uninstall, and Post-Uninstall tasks.

Assigning a Rule to a Group of Actions



Edition • The Action Group action is available only in InstallAnywhere Enterprise edition.

If you want to apply one or more rules to two or more actions, first create an **Action Group** as the parent of those actions and then apply the rules to the Action Group.



Task: To assign a rule to a group of actions

1. In a task with multiple existing actions, click **Add Action**.
2. In the **Choose an Action** dialog box, click **General**.
3. In the **General** tab, click **Action Group** and then click **Add**.
4. Use the arrow buttons to move the actions you want to apply rules to under the Action Group.



Tip • When you nest actions within an Action Group, any rules you assign to the Action Group must evaluate to true for these actions to run.

5. On the Action Group's customizer, click **Rules**.
6. On the **Rules** tab, click **Add Rule**. InstallAnywhere opens the Choose a Rule dialog box.
7. Choose the rule you want to add for the installer and click **Add**. InstallAnywhere adds the rule to the Rules list and shows the corresponding rule customizer.
8. Customize the rule according to your install requirements.

Building Complex Rule Expressions

InstallAnywhere allows you to define an unlimited number of rules and assign them to installers, files, panels, consoles, actions, and action groups. The location in the Advanced Designer where you define rules varies depending upon the item you are assigning it to:

- **Installers**—To evaluate a rule before any installation takes place, assign it on the **Project > Rules** subtask.
- **Files, Panels, Consoles, Actions, Action Groups**—To evaluate a rule on a file, panel, console, action, or action group, assign it on the **Rules** tab of the item customizer in the **Pre-Install**, **Install**, **Post-Install**, **Pre-Uninstall**, or **Post-Uninstall** task.

When you click **Add Rule** to add a rule to an item, the rule is listed, and the rule's unique ID (which is based on a combination of the abbreviation of the name of the rule and a numeric identifier) is displayed in the **Rule Expression** field.

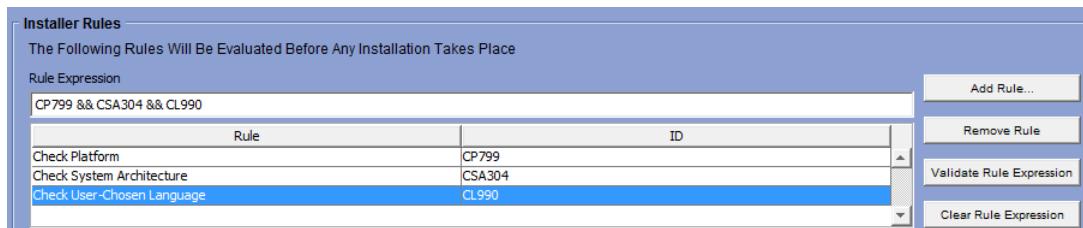


Figure 5-4: Rules Expression Containing Multiple Rules



Important • When a rule is added, InstallAnywhere automatically generates a unique ID for the rule. However, you can edit these rule IDs both in the **Rule Expression** field and in the **ID** column of the Rules table. If you edit these IDs, note the following:

- Make sure that the Rule ID listed in the **Rule Expression** field matches the Rule ID listed in the **ID** column of the Rules table.
- Make sure that two different rules do not use the same Rule ID

If you want to write complex rule expressions, you can edit the expression in the **Rule Expression** field to use multiple logical operators (such as AND, OR, and NOT) and precedence operators (parentheses) to express the relationship between two or more rules. By default, rules are joined by the AND operator.

The following is an example of a complex rule expression:

Rule1 AND Rule2 OR (Rule3 AND NOT Rule4)

Chapter 5: Creating Basic Installers

Defining Rules

For the logical operators, use the following symbols:

Table 5-6 • Logical Operators

Symbol	Operator	Purpose
&&	and	Implies that both rules must evaluate true for the installer to continue. If any rule fails to pass, the installer stops and issues the failure message.
	or	Implies that at least one of the rules must evaluate to true. If none of the rules pass, the installer stops and issues the failure message.
!	not	Implies that at least one of the rules must evaluate to false. If all of the rules pass, the installer stops and issues the failure message.

Therefore, the sample complex rule expression shown above would be written as:

Rule1 && Rule2 || (Rule3 && ! Rule4)

In the **Rules Expression** field, rules are represented by unique ID numbers, so this sample complex rule would look like this:

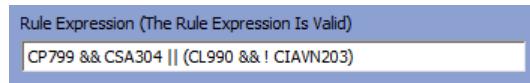


Figure 5-5: Complex Rule Expression



Tip • It is possible to create one rule set for an Action Group while setting secondary rules on the individual actions within the Action Group. This approach can yield robust, nuanced conditions in the installer. See [Assigning a Rule to a Group of Actions](#).



Note • The scope of the Rule Expression being formed is limited to the Files, Panels, Consoles, Actions, and Action Groups to which the rules have been added.

Validation of Complex Rule Expressions

You can choose to validate complex rule expressions both at design time and at build time:

- Validation at Design Time
- Validation at Build Time

Validation at Design Time

To validate a rule expression when you are defining it using the Advanced Designer, click the **Validate Rule Expression** button:

- **Valid**—If the expression is valid, [The Rule Expression Is Valid](#) is displayed.
- **Invalid**—If the expression is invalid, [Invalid Expression](#) is displayed in red.

Click the **Clear Rule Expression** button to clear all content in the **Rule Expression** field.

Validation at Build Time

Rule validation is also performed when a project is built. If the validation of a rule fails, the build will still be successful. However, a warning message will be displayed on the **Building...** dialog box (or the console) that points out the phase where the validation failed. If the validation of a rule fails, it will not be evaluated at runtime.

Customizing Built-in Rules

This section includes topics that describe how to configure settings for some of InstallAnywhere's built-in rules.

Table 5-7 • Customizing InstallAnywhere's Built-In Rules

Topic	Description
Customizing a Check File/Folder Attributes Rule	Describes how to define the file and folder attributes for the Check File/Folder Attributes rule.
Customizing a Check If File/Folder Exists Rule	Describes how to set up a Check If File/Folder Exists rule. (This rule is only available in the Install task.)
Customizing a Check Platform Rule	Describes how to specify platform conditions for the Check Platform rule.
Customizing a Check Running Mode Rule	Describes how to use Check Running Mode rules to enable you to customize the events that occur when an end user launches Maintenance Mode.
Customizing Evaluate Custom Rule Rules	Explains how to use an Evaluate Custom Rule rule to tailor custom rules built using the specifications outlined in the InstallAnywhere API to fit the needs of the installation.
Customizing a Compare InstallAnywhere Variable Numerically Rule	Explains how to use the Compare InstallAnywhere Variable Numerically rule to compare two InstallAnywhere variables numerically or to compare an InstallAnywhere variable against a specific value.

Customizing a Check File/Folder Attributes Rule

The Check File/Folder Attributes rule examines key attributes of a given file or folder. With this rule, you can

- Specify a file or folder to check
- Determine if that file or folder exists
- Determine whether the given path points to a file or a folder
- Determine if that file or folder is readable, writable, or both readable and writable
- Determine if that file or folder is idle or in-use

You can choose to test one or more of these attributes when you configure the Check File/Folder Attributes rule, but be aware that all selected attributes must evaluate to true for the rule to produce a true result.

**Task:****To customize a Check File/Folder Attributes rule:**

1. In the File/Folder Path text box, enter the location of the file or folder you want to evaluate.



Note • The File/Folder Path can include Magic Folders and other InstallAnywhere variables.

2. Set rule criteria.

Example: Verifying a File Exists Before Running the Installer

For many silent installs, the installer requires an installer properties file to provide settings that govern the installation. In this example, we'll use a **Check File/Folder Attributes** rule to prevent the installer from running unless the response file (installer.properties) exists in the same directory as the installer.

**Task:****To verify a file exists before running the installer**

1. Navigate to **Project > Rules** task.
2. Click **Add Rule**.
3. In the **Choose a Rule** dialog box, click **Check File/Folder Attributes** and then click **Add**.
4. On the **Check File/Folder Attributes** customizer, type the **File/Folder Path** value, `$INSTALLER_LAUNCH_DIR$/installer.properties`.
5. Retain the default rule criteria. (Perform only if the file/folder **already exists** and Perform only if the file/folder is a **file**.)



Note • For file- and folder-related install actions in the **Install** task, InstallAnywhere includes a separate **Check if File/Folder Exists** rule. This rule is not available in **Project > Rules**, but can be added via the customizer's **Rules** tab in the **Install** task.

Customizing a Check If File/Folder Exists Rule



Note • This rule is only available for file- and folder-related actions in the **Install** task.

By default, the **Check If File/Folder Exists** rule allows the specified file or folder to be installed only if that file does not exist on the target system.



Task: To customize a Check If File/Folder Exists rule

In the rule customizer, choose a setting to either permit or prevent overwriting an existing file or folder. Choose from **It does not already exist on the user's system** or **It already exists on the user's system**.

Customizing a Check Platform Rule

By default, the **Check Platform** rule initially includes all the built-in platforms in the **Do Not Perform On** list. Hence, without customization, this rule prevents the component to which it is assigned from running. You must move platforms into the **Perform On** list to define a set of platforms on which the installer, action, file, or folder can run.



Task: To customize the Check Platform rule:

1. Open the customizer of a **Check Platform** rule.
2. In the **Do Not Perform On** list, select the platforms on which you want your action or installer to run.
3. Click the right arrow (->). This moves the selected platforms to the **Perform On** list.
4. If you want to write a complex rule expression, you can edit the expression in the **Rule Expression** field to use multiple logical operators (such as AND, OR, and NOT) and precedence operators (parentheses) to express the relationship between two or more rules. For instructions, see [Building Complex Rule Expressions](#).



Note • If the platform on which your installer or action must execute is not one of the built-in platforms, you can create a custom platform expression. See [Example: Custom Platform Rule](#).

Example: Windows-Only Rule

Some actions may only make sense for Windows systems. Follow the steps below to create a “Windows-only” rule.



Task: To create a Windows-only rule

1. In the **Check Platform** customizer, click **Windows (All)**.
2. Click the right arrow (->).



Note • The Windows (All) rule returns true when the target machine's Java VM reports any "Windows" value from a System.getProperty (os.name) method call.

Example: Custom Platform Rule

If your installer includes an action that must run on Windows 98 systems but not on any other system, you can create a custom platform expression. The steps in this example below use custom expressions to create such a rule.

**Task:** *To create a custom platform rule*

1. In the **Check Platform** customizer, click **Windows (All)**.
2. Click **Remove Platform**. (No Windows systems can produce a true result unless we remove **Windows (All)** from the **Do Not Perform On** list.)
3. Click **More Platforms**. This opens the Add Platform dialog box.
4. In the Add Platform dialog box, type **Windows 98** in the text box.
5. Click **OK**. **Windows 98** appears in the **Perform On** list.

Customizing a Check Running Mode Rule

As described in [About Maintenance Mode Action Groups](#), when you enable Maintenance Mode, **Check Running Mode** rules are automatically added to the Action Groups that are created in the **Pre-Install** and **Pre-Uninstall** tasks. However, you can also manually apply a Check Running Mode rule to any Action Group, Action, or Panel in your installation project to specify whether or not that element should be executed when Maintenance Mode is run. This enables you to customize the events that occur when an end user launches Maintenance Mode.



Important • When a Check Running Mode rule is assigned to an Action Group, it is applied to all panels and actions in that Action Group.

**Task:** *To add a Check Running Mode rule to an Action Group, action, or panel:*

1. In the Advanced Designer, select the Action Group, Action, or Panel that you would like to add a **Check Running Mode** rule to.
2. Select the **Rules** tab.
3. Click **Add Rule**. The **Choose a Rule** dialog box opens.
4. Select **Check Running Mode** and click **Add**. The Check Running Mode rule is now listed in the Rules List.
5. Click **Close** to close the **Choose a Rule** dialog box.

6. In the Check Running Mode customizer, select one of the following options to associate the selected element with one of the Maintenance Mode options:

Task	Available Check Running Mode Options
Pre-Install	<p>In the Pre-Install task, the following Check Running Mode options are available:</p> <ul style="list-style-type: none"> • Installation • Add Features • Repair Installation  <p>Note • If a panel in the Pre-Install phase does not have a Check Running Mode rule assigned either to itself or any of its parents, then this panel will be run during Install, Maintenance Mode/Add Features, and Maintenance Mode/Repair Installation.</p>
Install	<p>In the Install task, the following Check Running Mode options are available:</p> <ul style="list-style-type: none"> • Installation • Add Features • Repair Installation
Post-Install	<p>In the Post-Install task, the following Check Running Mode options are available:</p> <ul style="list-style-type: none"> • Installation • Add Features • Repair Installation
Pre-Uninstall	<p>In the Pre-Uninstall task, the following Check Running Mode options are available:</p> <ul style="list-style-type: none"> • Uninstallation • Remove Features  <p>Note • If a panel in the Pre-Uninstall phase does not have a Check Running Mode rule assigned either to itself or any of its parents, then this panel will be run during Uninstall and Maintenance Mode/Remove Features.</p>
Uninstall	<p>In the Uninstall task, the following Check Running Mode options are available:</p> <ul style="list-style-type: none"> • Uninstallation • Remove Features
Post-Uninstall	<p>In the Post-Uninstall task, the following Check Running Mode options are available:</p> <ul style="list-style-type: none"> • Uninstallation • Remove Features

For example, when an end user chooses to run Maintenance Mode to add features, only those actions and panels with a Check Running Mode rule set to **Add Features** will be executed.



Important • You should always add a Check Running Mode rule to the Uninstaller action for your product, and you should set that Check Running Mode rule to **Installation** so that it executes only once.

Customizing Evaluate Custom Rule Rules

Custom rules built using the specifications outlined in the InstallAnywhere API can be tailored to fit the needs of the installation.

The process of creating a custom rule is similar to creating a custom action. To create a rule, you create a custom class that extends `com.zerog.ia.api.pub.CustomCodeRule`, and this class must implement an `evaluateRule` method that returns true if the rule succeeds and false if the rule fails.

For example, the implementation of a rule that always succeeds would appear similar to the following:

```
import com.zerog.ia.api.pub.*;  
  
public class AlwaysSucceedsRule extends CustomCodeRule  
{  
    public boolean evaluateRule()  
    {  
        return true; // always succeed  
    }  
}
```

You compile the rule class the same way you compile other custom code (by including `IAClasses.zip` in the compiler classpath), and package the `.class` file in a `.jar` file or `.zip` file as before.



Task: *To customize an Evaluate Custom Rule rule:*

1. Compile the class containing the custom code and package it in a `.jar` or `.zip` file.
2. In the Advanced Designer, select the Action Group, Action, or Panel that you would like to add an **Evaluate Custom Rule** rule to.
3. Select the **Rules** tab.
4. Click **Add Rule**. The **Choose a Rule** dialog box opens.
5. Select **Evaluate Custom Rule** and click **Add**. The Evaluate Custom Rule rule is now listed in the Rules List.
6. Click **Close** to close the **Choose a Rule** dialog box.
7. In the **Evaluate Custom Rule** customizer, click the **Choose JAR or ZIP** button next to the **Path** field and browse to the `.jar` or `.zip` file containing the custom class.
8. In the **Class** field, enter the fully qualified custom rule class name (such as `com.acme.MyCustomCodeRule`) of the Java class that implements the rule.



Note • This class must extend `com.zerog.ia.api.pub.CustomCodeRule` and must implement a public boolean `evaluateRule()` method.

9. Optionally, click the **Configure Dependencies** button to open the **Custom Rules Dependencies** dialog box, where you can select a .jar or .zip file which contains classes referenced by your custom rule so that those classes are included in the archive and are available to the rule at runtime.

At run time, if the rule succeeds, the action associated with it will be installed or performed, and if the rule fails the action will be skipped.



Note • For examples of Evaluate Custom Code rules, see [Evaluate Custom Rule Examples](#).

Customizing a Compare InstallAnywhere Variable Numerically Rule

You can use the **Compare InstallAnywhere Variables Numerically** rule to compare two InstallAnywhere variables numerically or to compare an InstallAnywhere variable against a specific value.

When this rule is applied, it converts the content of the variables into numeric values and then compares them. Integer, floating point, long and double operations are supported in this rule. However if the value represented by the variable is not parseable as a numeric value, then the rule returns false.



Task:

To add a Compare InstallAnywhere Variable Numerically rule to an Action Group, action, or panel:

1. In the Advanced Designer, select the Action Group, Action, or Panel that you would like to add a **Compare InstallAnywhere Variable Numerically** rule to.
2. Select the **Rules** tab.
3. Click **Add Rule**. The **Choose a Rule** dialog box opens.
4. Select **Compare InstallAnywhere Variable Numerically** and click **Add**. The new rule is now listed in the Rules List.
5. Click **Close** to close the **Choose a Rule** dialog box.
6. In the **Operand 1** and **Operand 2** text boxes on the **Compare InstallAnywhere Variable Numerically** customizer, enter the two InstallAnywhere variables that you want to compare, or enter an InstallAnywhere variable as one operand and enter a specific value as the second operand.



Note • For the operands, you can enter either a variable or (such as `$MY_VARIABLE$`) or a literal, specific value (such as `4502`).

7. From the list, select one of the following operators:

- **greater than or equal to (\geq)**
- **equal to ($=$)**
- **less than ($<$)**
- **less than or equal to (\leq)**
- **greater than ($>$)**
- **greater than or equal to (\geq)**

Rules Are Evaluated Multiple Times

When an installer is run, rules get evaluated multiple times and at points in the sequence that are much earlier than where the rule is actually located. This may be important to consider because a rule could be referring to an InstallAnywhere variable that has not yet been set at the point when the rule is evaluated, which may cause the rule to fail—especially in custom rules. You may want to evaluate this to ensure that the rule's parameters (InstallAnywhere variables) are set correctly.

The number of times and when each rule will be evaluated depends upon where the rule is placed in the InstallAnywhere project. For example, if you have a rule on a component, whenever that component is referenced, the rule is evaluated. Even if a file (not a component) is being installed, the rules of all of that file's parents, including the component, is evaluated.

Suppose you have the following hierarchical structure of items:

- **Item A** [with **Rule RA**]
 - **Item B** [with **Rule RB**]
 - **Item C** [with **Rule RC**]

When this installer is run, the following would occur:

- **Rule RC** would be evaluated once, just when **Item C** is installed.
- **Rule RB** would be evaluated twice: once when **Item B** is installed, and again when **Item C** is installed.
- **Rule RA** would be evaluated three times: once when **Item A** is installed, then when **Item B** is installed, and a third time when **Item C** is installed.



Note • This example assumes that Items A, B, and C do not have any other elements assigned to them.

Regarding when and how many times a rule is evaluated, also note the following:

- **Panels**—In case of panels, the rule is evaluated once during the preflight of all panels at install start, and during the uninstall start. The rules of the panel are also evaluated each time the panel is about to be visited.
- **OR based conditions**—The rules on the installer are visited once during install start. However, if you have any OR-based conditions, then the rules are evaluated for every panel and every action.



Tip • If you are unsure as to when an **InstallAnywhere** variable that you are using is populated, you may want to consider using a **Compare InstallAnywhere Variable** rule along with your custom rule.

Implementing Maintenance Mode



Edition • The Maintenance Mode option is available in **InstallAnywhere Enterprise Edition**.

You can choose to implement Maintenance Mode in an installer so that end users are able to add or remove features to previously installed products as well as repair broken installations. The end user will be able to launch Maintenance Mode by:

- Executing the **Change Installation** launcher.
- Selecting **Add or Remove Programs** from the Windows Control Panel (Microsoft Windows OS only).

When an end user launches Maintenance Mode, they will be prompted to select from the listed options, which may include **Add Features**, **Remove Features**, **Repair Product**, and **Uninstall Product**.

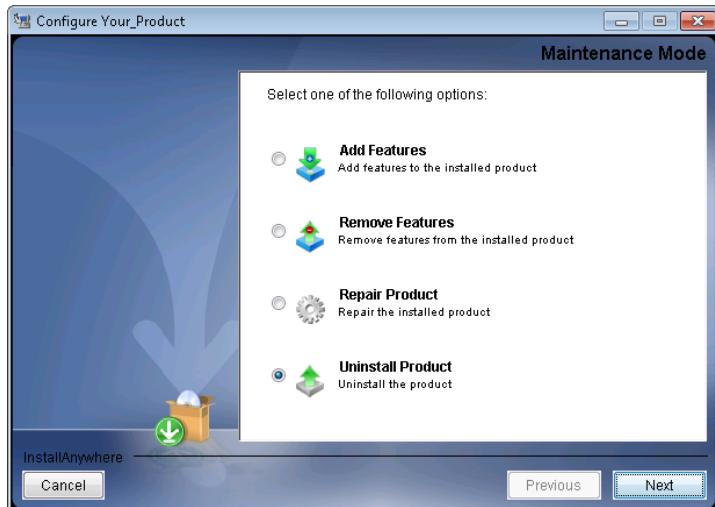


Figure 5-6: Maintenance Mode Panel



Note • The options listed on this panel depend upon the options you select in the **Maintenance Mode** area of the **Project > Advanced** subtask. For more information, see [Enabling and Configuring Maintenance Mode](#).

Chapter 5: Creating Basic Installers

Implementing Maintenance Mode

Information about implementing Maintenance Mode is presented in the following sections:

Table 5-8 • Implementing Maintenance Mode

Section	Description
Enabling and Configuring Maintenance Mode	Explains how to enable Maintenance Mode support in an InstallAnywhere project, specify the options to include, and configure how Maintenance Mode will be implemented.
Maintenance Mode End User Experience	Provides examples of the different end user experiences for the various Maintenance Mode options.
About the Uninstaller / Maintenance Mode Launcher	Explains the methods the end user can use to launch Maintenance Mode and the Uninstaller.

Enabling and Configuring Maintenance Mode

To include Maintenance Mode support in your installation project, you need to first enable the Maintenance Mode options you want to include, and then customize those options for the product being installed.

Table 5-9 • Steps to Enable Maintenance Mode

Step	Description
Enabling Maintenance Mode	Explains how to enable Maintenance Mode support in a project and the purpose of each of the Maintenance Mode options.
About Maintenance Mode Action Groups	Describes the panels that are automatically added to your project when you select Maintenance Mode options, and explains how to customize those panels.
Customizing Maintenance Mode Text and Images	Explains how to customize the text and images that are displayed on the Maintenance Mode panel of the Change Installation Launcher.
About Check Running Mode Rules	Explains the purpose of the Check Running Mode rules that are automatically added to your installation project when Maintenance Mode is enabled, and how to use them to customize the events that occur when an end user launches Maintenance Mode.
Specifying Instance Management Options	Explains how to configure an installation project to specify whether multiple instances of a product can be installed on the same machine.

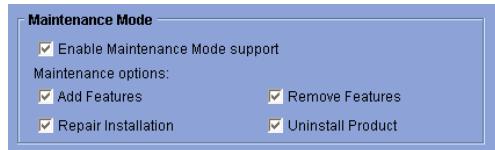
Enabling Maintenance Mode

You can enable Maintenance Mode support and identify an installer's supported Maintenance Mode options on the **Project > Advanced** subtask of the Advanced Designer.



Task: *To enable Maintenance Mode support and identify supported options:*

1. Open an installation project in the Advanced Designer.
2. On the **Project** task, click **Advanced**. The **Project > Advanced** subtask opens.
3. Under **Maintenance Mode**, select **Enable Maintenance Mode support**. The Maintenance Mode options are then enabled.



Important • After selecting **Enable Maintenance Mode support**, you are required to select at least one of the four **Maintenance options**. Failure to do so will result in build failure.



Note • For all new or migrated projects, the **Enable Maintenance Mode support** option will be disabled by default.

4. Select one (or more) of the following options:

Option	Description
Add Features	<p>Select this option to enable the end user to use Maintenance Mode to install previously uninstalled product features. An end user can use Add Features to:</p> <ul style="list-style-type: none"> • Install product features that they did not choose to install during the initial installation. • Re-install product features that had originally been installed but were removed using the Remove Features option of Maintenance Mode. <p>When the end user launches Maintenance Mode and chooses Add Features, the installer will advance to the Pre-Install phase of the installation and then proceed as a regular installation.</p>  <p>Note • The set of actions executed when the end user selects Add Features do not include the whole set of actions in Pre-Install, but a smaller subset for which the Check Running Mode Rule is set to Pre-Installation. See About Check Running Mode Rules.</p>  <p>Note • For more information, see Adding a Feature User Experience.</p>
Remove Features	<p>Select this option to enable the end user to use Maintenance Mode to remove previously installed product features.</p>  <p>Tip • The InstallAnywhere Uninstaller also provides you with the ability to selectively remove individual features.</p> <p>When the end user launches Maintenance Mode and chooses Remove Features, the installer will advance to the Pre-Uninstall phase of the uninstallation and then proceed with the uninstallation of the selected features.</p>  <p>Note • The set of actions executed when the end user selects Remove Features do not include the whole set of actions in Pre-Uninstall, but a smaller subset for which the Check Running Mode Rule is set to Remove Features. See About Check Running Mode Rules.</p>  <p>Note • For more information, see Removing a Feature User Experience.</p>

Option	Description
Repair Installation	<p>Select this option to enable the end user to use Maintenance Mode to repair a previously installed product.</p> <p>When the end user launches Maintenance Mode and chooses Repair Product, all of those actions in the Pre-Install phase with a Check Running Mode Rule set to Repair Installation will be executed (not just those actions that exist in the Repair Installation Action Group). See About Check Running Mode Rules.</p>  <p>Note • For more information, see Repairing a Product User Experience.</p>
Uninstall Product	<p>Select this option to enable the end user to use Maintenance Mode to uninstall a previously installed product.</p>  <p>Tip • The InstallAnywhere Uninstaller also provides you with the ability to uninstall a product.</p> <p>When the end user launches Maintenance Mode and chooses Uninstall Product, the installer will advance to the Pre-Uninstall phase of the uninstallation and proceed with the uninstallation of the product.</p>  <p>Note • For more information, see Uninstalling a Product User Experience.</p>



Tip • You can assign a **Check Running Mode** rule to associate elements of your installation project with the Add/Remove/Repair/Uninstall Maintenance Mode options. For example, you may want to add a Check Running Mode rule to an action that should not be executed during the Repair phase. For more information, see [About Check Running Mode Rules](#).

5. Save the InstallAnywhere project.

About Maintenance Mode Action Groups

When you enable Maintenance Mode support, the Advanced Designer adds new Action Groups to the default **Pre-Install** and **Pre-Uninstall** tasks. Depending on the Maintenance Mode options you selected (**Add Features**, **Remove Features**, **Repair Installation**, **Uninstall Product**), separate Action Groups will be created to run when Maintenance Mode is launched.

- [Pre-Install Task](#)
- [Pre-Uninstall Task](#)



Important • This section details the Action Groups and panels that are automatically added to your project when you enable Maintenance Mode support. However, you may want to customize these panels.

Pre-Install Task

When Maintenance Mode is enabled, the following Action Groups are created in the **Pre-Install** task:

- Pre-Installation Action Group
- Add Features Action Group
- Repair Installation Action Group

Pre-Installation Action Group

When you create a new project, and Maintenance Mode is not selected, the **Pre-Install** Action List consists of several panels:



Note • In the case of InstallAnywhere projects created using previous versions which are migrated to InstallAnywhere 2011, all panels and actions present in the Pre-Install phase would, by default, be grouped under this new Pre-Installation Action Group. The Check Running Mode Rule would be set to **Pre-Installation** to enable the existing panels to run as-is.

As soon as you enable Maintenance Mode, those panels are grouped into a new Action Group named **Pre-Installation**, and a Check Running Mode rule with a value of **Pre-Installation** is assigned to that group.



Figure 5-7: Pre-Installation Action Group on the Pre-Install Action List

These default Action Groups are provided as a convenience; you are not required to group actions into Action Groups in order to assign a Check Running Mode rule to an action or panel. You are permitted to assign Check Running Mode Rules directly to individual actions or panels. You can choose to keep these default Action Groups or delete them, but remember to add the appropriate Check Running Mode Rule to those actions and panels that you want to be executed during Maintenance Mode.



Note • Because a Check Running Mode rule with the value of **Pre-Installation** is assigned to the Pre-Installation Action Group, these panels will only be executed during installation, not during the Add Features or Repair Installation phases of Maintenance Mode. For more information, see [About Check Running Mode Rules](#).

Add Features Action Group

If you select the **Add Features** Maintenance Mode option, the **Add Features** Action Group is added to the **Pre-Install** Action List, and a Check Running Mode rule with a value of **Add Features** is assigned to that group.



Figure 5-8: Add Features Action Group on the Pre-Install Action List

By default, the **Add Features** Action Group consists of an **Introduction** and a **Choose Install Sets** panel.



Note • Because a Check Running Mode rule with the value of **Add Features** is assigned to the Add Features Action Group, these panels will only be executed during the Add Features phase of Maintenance Mode. For more information, see [About Check Running Mode Rules](#).

Repair Installation Action Group

If you select the **Repair Installation** Maintenance Mode option, the **Repair Installation** Action Group is added to the **Pre-Install** Action List, and a Check Running Mode rule with a value of **Repair Installation** is assigned to that group.



Figure 5-9: Repair Installation Action Group on the Pre-Install Action List

By default, the **Repair Installation** Action Group consists of an **Introduction** panel.

By default, when an end user chooses the Repair Installation option, the product would be reinstalled. If, at the same time, you would like to perform some other kind of repair operations, such as running a script that repairs a database, you may want to add an additional action to the **Repair Installation** Action Group.



Note • Because a Check Running Mode rule with the value of **Repair Installation** is assigned to the Repair Installation Action Group, these panels will only be executed during the Repair Installation phase of Maintenance Mode. For more information, see [About Check Running Mode Rules](#).

Pre-Uninstall Task

When Maintenance Mode is enabled, the following Action Groups are created in the **Pre-Uninstall** task:

- Pre-Uninstallation Action Group
- Remove Features Action Group

Pre-Uninstallation Action Group

When you create a new project, and Maintenance Mode is not selected, the **Pre-Uninstall** task consists of several panels:

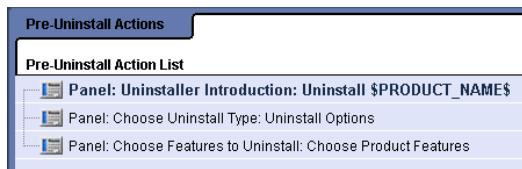


Figure 5-10: Default Panels on the Pre-Uninstall Action List

As soon as you enable Maintenance Mode, those panels are grouped into a new Action Group named **Pre-Uninstallation**, and a Check Running Mode rule with a value of **Pre-Uninstallation** is assigned to that group.

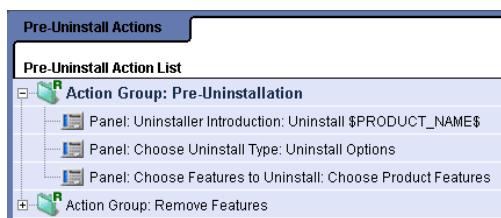


Figure 5-11: Pre-Uninstallation Action Group on the Pre-Uninstall Action List



Note • Because a Check Running Mode rule with the value of **Pre-Uninstallation** is assigned to the Pre-Uninstallation Action Group, these panels will only be executed during uninstallation, not during the **Remove Features** phase of Maintenance Mode. For more information, see [About Check Running Mode Rules](#).

Remove Features Action Group

If you select the **Remove Features** Maintenance Mode option, the **Remove Features** Action Group is added to the **Pre-Uninstall Action List**, and a Check Running Mode rule with a value of **Remove Features** is assigned to that group.



Figure 5-12: Remove Features Action Group on Pre-Uninstall Action List

By default, the **Remove Features** Action Group consists of an **Introduction** and a **Choose Features to Uninstall** panel.



Note • Because a Check Running Mode rule with the value of **Remove Features** is assigned to the Remove Features Action Group, these panels will only be executed during the **Remove Features** phase of Maintenance Mode. For more information, see [About Check Running Mode Rules](#).

Customizing Maintenance Mode Text and Images

You can customize the text and images that are displayed on the **Maintenance Mode** panel of the Change Installation Launcher.

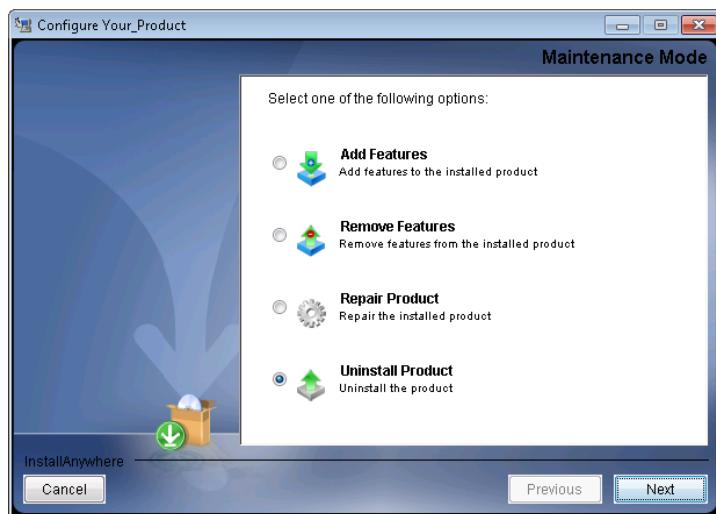


Figure 5-13: Maintenance Mode Panel of the Change Installation Launcher

Chapter 5: Creating Basic Installers

Implementing Maintenance Mode

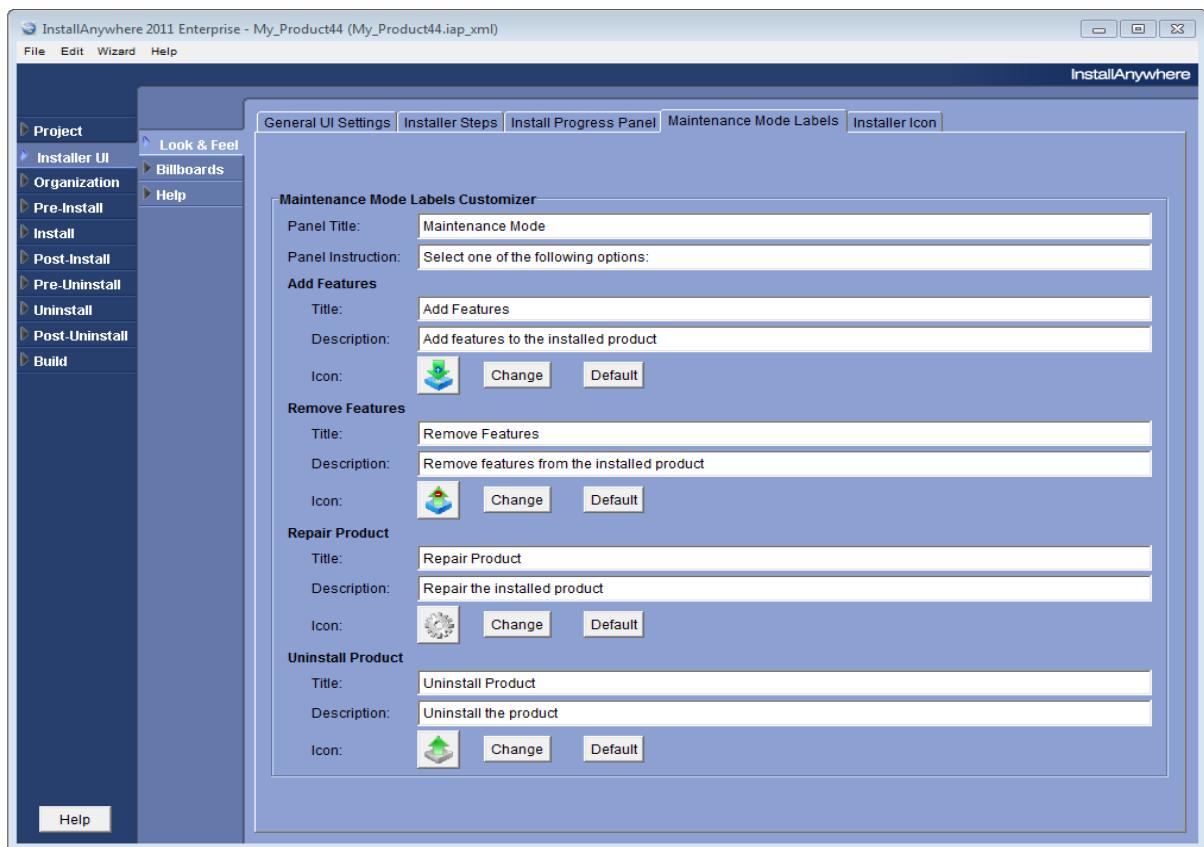
You can make these customizations on the **Maintenance Mode Labels Customizer** which is displayed when you select the **Maintenance Model Labels** tab of the **Installer UI > Look & Feel** subtask in the Advanced Designer.

To customize Maintenance Mode text and images, perform the following steps:



Task: To customize Maintenance Mode text and images:

1. Open an installation project in the Advanced Designer.
2. On the **Installer UI** task, click **Look & Feel**.
3. Open the **Maintenance Model Labels** tab. The **Maintenance Mode Labels Customizer** opens.



4. Edit the following properties:

Property	Description
Panel Title	Enter the overall title of the Maintenance Mode panel. The default value is Maintenance Mode .
Panel Instruction	Enter the instructional sentence that appears directly under the Panel Title . The default value is Select one of the following options :

Property	Description
Add Features	Set the following properties to define the selection options on the Maintenance Mode panel:
Remove Features	
Repair Product	
Uninstall Product	<ul style="list-style-type: none"> • Title—Label which identifies the selection. This text appears in bold. • Description—Text which describes the selection. This text appears in plain text directly under the Title. • Icon—The default icon that represents the selection. To select a different icon, click Change and select an image (.gif, .jpg, .jpeg, or .png). To revert back to the default image, click Default.

5. Save the project.

About Check Running Mode Rules

As described in [About Maintenance Mode Action Groups](#), when you enable Maintenance Mode, **Check Running Mode** rules are automatically added to the Action Groups that are created in the **Pre-Install** and **Pre-Uninstall** tasks. However, you can also manually apply a Check Running Mode rule to any Action Group, Action, or Panel in your installation project to specify whether or not that element should be executed when Maintenance Mode is run. This enables you to customize the events that occur when an end user launches Maintenance Mode.

For instructions on adding or customizing a Check Running Mode rule, see [Customizing a Check Running Mode Rule](#).



Important • When a Check Running Mode rule is assigned to an Action Group, it is applied to all panels and actions in that Action Group.

Specifying Instance Management Options



Edition • This feature is included with [InstallAnywhere Enterprise Edition](#).

InstallAnywhere's Maintenance Mode support includes an **Instance Management** feature that enables you to specify whether multiple instances of a product can be installed on the same machine.

If an installation includes both Maintenance Mode support and support for multiple instances of the product on the same machine, when an end user launches Maintenance Mode, they are able to specify which instance of the product they want to perform maintenance on.



Note • The instance count is only as secure as the security of the [InstallAnywhere product registry file](#). For more information, see [Product Registry](#).

Chapter 5: Creating Basic Installers

Implementing Maintenance Mode

The **Instance Management** options are on the **Project > Advanced** subtask of the Advanced Designer.

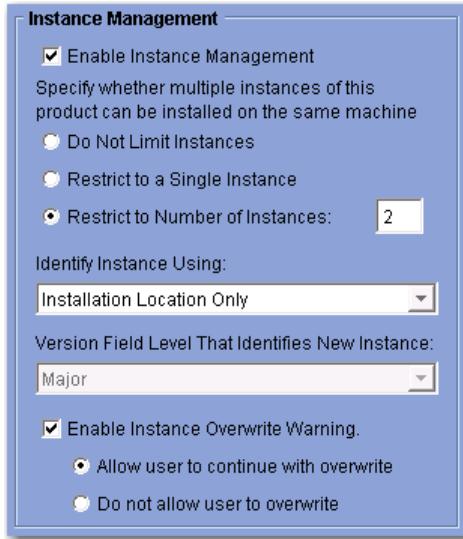


Figure 5-14: Instance Management Options on Project > Advanced Subtask



Important • The **Instance Management** options are only enabled if the **Enable Maintenance Mode support** option is selected. However, you can provide Maintenance Mode support without enabling Instance Management support.

To specify Instance Management options, perform the following steps:



Task:

Specifying Instance Management options:

1. Open an installation project in the Advanced Designer.
2. On the **Project** task, click **Advanced**. The **Project > Advanced** subtask opens.
3. Under **Maintenance Mode**, select **Enable Maintenance Mode Support**. The Maintenance Mode and **Enable Instance Management** options are then enabled.



Note • For all new or migrated projects, the **Enable Maintenance Mode Support** option will be disabled by default.

4. Set your desired Maintenance Mode options, as described in [Enabling Maintenance Mode](#).
5. Under **Instance Management**, select **Enable Instance Management**. The Instance Management options are enabled.

6. Select one of the following options:

Option	Description
Do Not Limit Instances	<p>Select this option to enable end users to install an unlimited number of instances of this product on a machine.</p> <p>End users may want to install multiple instances on the same machine if they want to configure each instance differently (each with different features, configuration settings, environments) and want to launch each instance independently (such as launching multiple instances on WebSphere/Tomcat using different JVM versions).</p> <p>If an installer was created with the Do not limit instances option selected, the following occurs:</p> <ul style="list-style-type: none"> When an end user attempts to install a second instance of that product, the Manage Instances dialog box opens, prompting the user to specify that they want to Install a New Instance (rather than Modify an Existing Instance). When an end user has installed multiple instances of a product on a machine and then launches Maintenance Mode, the Manage Instances dialog box opens, prompting the user to select the instance that they want to modify.  <p>Note • For more information, see Maintenance Mode User Experience on a Machine With Multiple Installed Instances.</p>
Restrict to a Single Instance	<p>Select this option to restrict end users to be able to install only one instance of the product on a machine.</p> <p>By selecting the Restrict to a Single Instance option, you are preventing the end user from being able to do either of the following:</p> <ul style="list-style-type: none"> Installing additional instances of the product on the same machine. Overwriting an existing instance of the product. <p>If an installer was created with the Restrict to a Single Instance option selected, when an end user attempts to install a second instance of that product, the Manage Instances dialog box opens, prompting the user to modify the existing instance, rather than to install a new instance.</p>
Restrict to Number of Instances	<p>Select this option and enter a number in the text box to restrict end users to be able to install only a specified number of instances of the product on a machine.</p>

Option	Description
Identify Instance Using	<p>Select one of the following options to specify how to identify an instance of an application:</p> <ul style="list-style-type: none"> • Installation Location Only—Identify an instance by examining the installation location. No filters are applied to the detected instances. • Installation Location and Version—Identify an instance by examining the installation location and the application version of the installed application. When this option is selected, the Version Field Level That Identifies New Instance option is enabled, which enables you to identify what level of version change constitutes a new version.
Version Field Level That Identifies New Instance	<p>When you select Installation Location and Version from the Identify Instance Using option, this option is enabled. You are prompted to specify what level of version change constitutes a new version by selecting one of the following options:</p> <ul style="list-style-type: none"> • Major • Minor • Revision • Subrevision <p></p> <p>Note • Versions are conventionally represented in the following format: [Major].[Minor].[Revision].[Subrevision], such as: 1.0.2.1047.</p> <p>For example, if you select the version level of Minor to identify a new version, versions 1.5.4 and 1.5.6 would be considered to be the same version because both the Major and Minor version are identical (even though the Revision version is different), while versions 1.5.4 and 1.6.2 would be considered to be two different versions because the Minor version does not match.</p> <p></p> <p>Note • Reinstalling an application to the same location as an existing application would result in only one instance, irrespective of the version.</p> <p></p> <p>Note • If you choose the Installation Location and Version option from the Identify Instance Using list, the installation does not recognize those instances which are from a more recent version, because an installer of a previous version might not be able to successfully manage an instance of the installer of a more recent version. For example, a Version 2.0.0.0 installer will not consider Version 3.x instances, but will consider all Version 1.x instances and Version 2.0.0.0 instances.</p>

Option	Description
Enable Instance Overwrite Warning	<p>Select this option to warn the end user when they are about to overwrite an existing instance of an application and specify whether they will be permitted to overwrite the instance. After you select this option, select one of the following options:</p> <ul style="list-style-type: none"> • Allow user to continue with overwrite—Permit end user to overwrite an existing instance of an application. • Do not allow user to overwrite—Do not permit end user to overwrite an existing instance of an application. Instead, prompt end user to change the installation location in order to continue with the installation.  <p>Note • If this option is selected, you are required to include a Panel: Choose Install Folder action in the Pre-Install task.</p>

7. Save your installation project.



Important • When an installer that has the *Instance Management* feature enabled is launched, it needs to determine how many instances of the application have already been installed. To do this, the installer queries the product registry to locate the uninstaller executable for that application. By default, the uninstaller executable is named *Change ProductName Installation.exe* (in *InstallAnywhere* 2010 or later) or *Uninstall ProductName.exe* (in versions prior to *InstallAnywhere* 2010). However, if you have chosen to rename the executable file of the uninstaller, that file will not be located and the *Instance Management* feature will not work properly.



Note • When using the *Instance Management* feature, note the following:

- *Instance Management* uses the *global/local zerog* registries. If these directories are altered, then the *Instance Management* feature will not work properly.
- The **InstallAnywhere Uninstall Component**, which is a standard component in an *InstallAnywhere* project, should have the **Key File** set to the *Uninstaller* executable file, which is the default setting. However, if the **Key File** is modified, then the *Instance Management* feature may not be in a position to manage instances because it would be unable to find the *Uninstaller*.

The **Key File** for the **InstallAnywhere Uninstall Component** is set on the **Properties > Component** subtab of the **Organization > Components** subtask.

About the Uninstaller / Maintenance Mode Launcher

By default, an InstallAnywhere project is configured to create one Uninstaller / Maintenance Mode executable named Change *Product_Name* Installation.exe.



Note • You can modify the name of this executable by selecting the **Uninstall** launcher on the **Install** task and editing the **Name** field on the **Properties** tab of the **Create Uninstaller** customizer.



Note • If you are migrating a project from an earlier version of InstallAnywhere, the launcher will retain the name that it had in that earlier version. In previous releases, the default name for this launcher was Uninstall *ProductName*.exe. You might want to consider changing the name of the uninstall launcher to match the default name assigned to projects created in InstallAnywhere 2010 or later: Change *Product_Name* Installation.exe.

The behavior of this launcher varies depending upon whether or not you have implemented Maintenance Mode:

- **Maintenance Mode not enabled**—If you have not enabled Maintenance Mode, opening this launcher launches the standard Uninstall runtime.

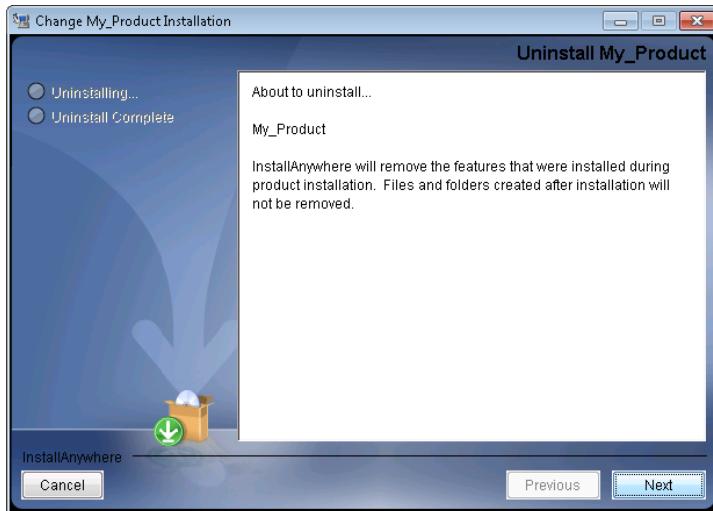


Figure 5-15: Uninstall Panel of the Change Installation Launcher

- **Maintenance Mode enabled**—If you have enabled Maintenance Mode, running this launcher opens the **Configure Maintenance Mode** panel, which prompts the end user to select from the listed options (which may include **Add Features**, **Remove Features**, **Repair Product**, and **Uninstall Product**).

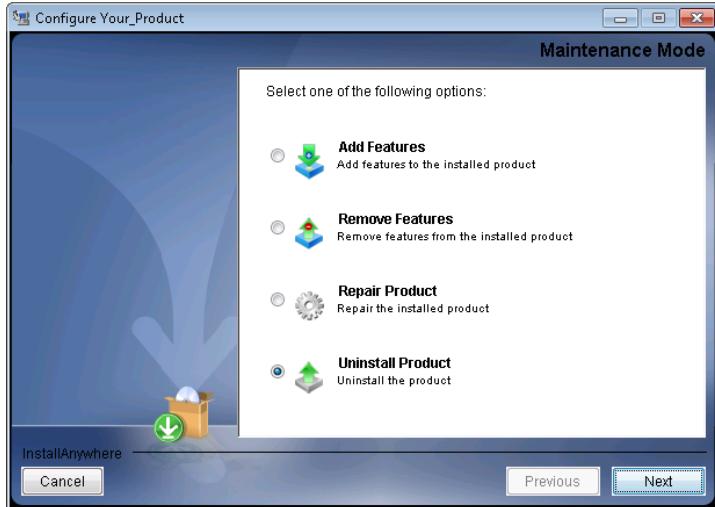


Figure 5-16: Maintenance Mode Panel of the Change Installation Launcher

Maintenance Mode End User Experience

When an end user launches a Change Installation launcher created using the Maintenance Mode option, the user experience varies depending upon the Maintenance Mode options you specified on the **Project > Advanced** subtask. This section describes those user experiences.

- Initial Experience
- Repairing a Product User Experience
- Adding a Feature User Experience
- Removing a Feature User Experience
- Uninstalling a Product User Experience
- Maintenance Mode User Experience on a Machine With Multiple Installed Instances

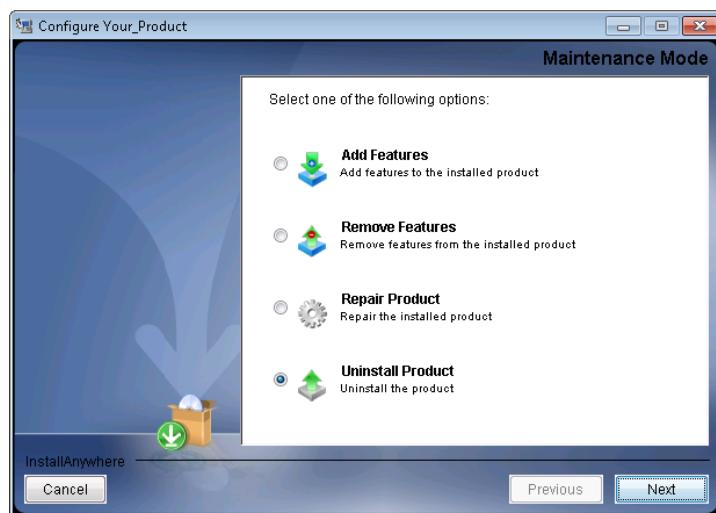
Initial Experience

When an end user launches a Change Installation launcher created using the Maintenance Mode option, their user experience begins as follows:

**Task:** **To launch the “Change Installation” Maintenance Mode launcher:**

1. Launch the Maintenance Mode by doing one of the following:
 - Opening Change *Product_Name* Installation.exe.
 - Selecting **Add or Remove Programs** from the Windows Control Panel (Microsoft Windows OS only).

The **Maintenance Mode** panel opens, and the options that were selected on the **Product > Advanced** subtask of the Advanced Designer are displayed:



Note • The Maintenance Mode panel appears only once. After the user selects an option and clicks next, they will be unable to return to this panel by clicking the **Previous** button.

2. Select one of the supported options. The Maintenance Mode runtime executes the selected option, as described in the following topics:
 - [Adding a Feature User Experience](#)
 - [Removing a Feature User Experience](#)
 - [Repairing a Product User Experience](#)
 - [Uninstalling a Product User Experience](#)
 - [Maintenance Mode User Experience on a Machine With Multiple Installed Instances](#)

Adding a Feature User Experience

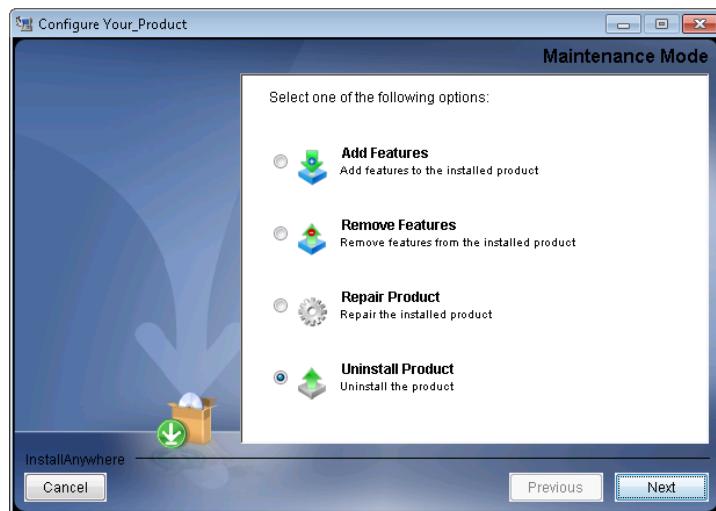
This section explains what an end user will see when they launch the Change Installation launcher and choose the **Add Feature** option.



Task:

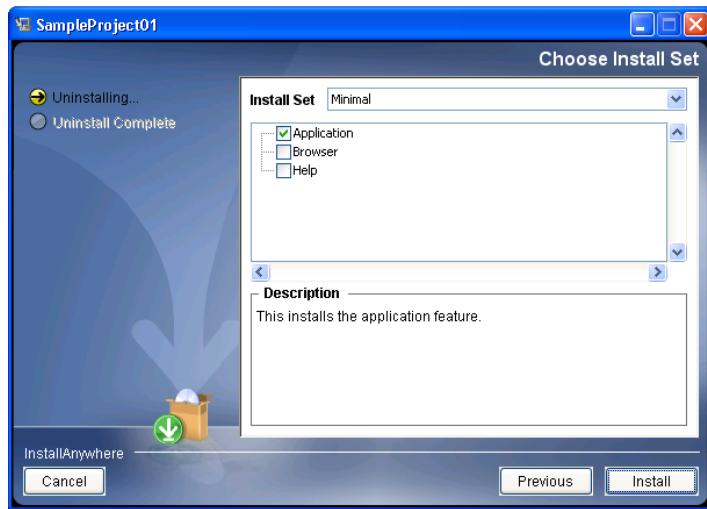
To add features to an installed product:

1. Open the Change Product Installation launcher, as described in [Initial Experience](#). The **Maintenance Mode** panel opens.



2. Select **Add Features** and click **Next**. The Maintenance Mode runtime displays the panels in the **Add Features** Action Group on the **Pre-Install** Action List and proceeds as a regular installation.

The **Choose Install Set** panel displays the features that are already installed and only allows the selection of the ones that are not yet installed.





Note • Only those Action Groups, Actions, and Panels that have been assigned a **Check Running Mode** rule with a value of **Add Features** will be executed.

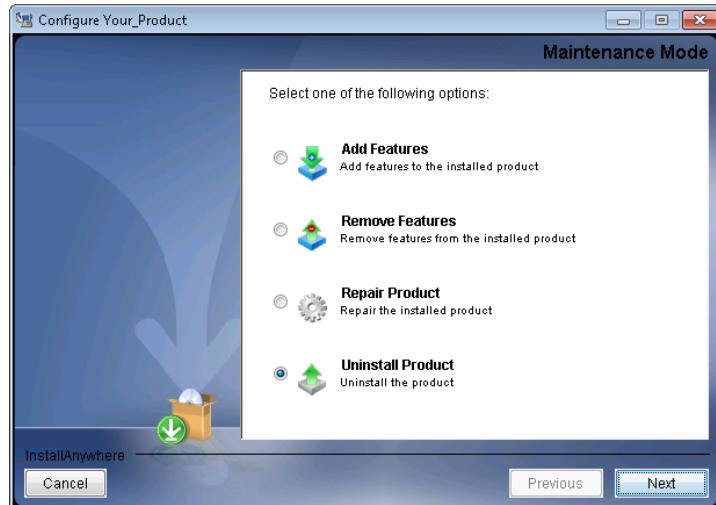
Removing a Feature User Experience

This section explains what an end user will see when they launch the Change Installation launcher and choose the **Remove Features** option



Task: *To remove features to an installed product:*

1. Open the Change Installation launcher, as described in [Initial Experience](#). The **Maintenance Mode** panel opens.



2. Select **Remove Features** and click **Next**. The Maintenance Mode runtime displays the panels in the **Remove Features** Action Group on the **Pre-Uninstall** Action List and proceeds as a regular installation.



Note • Only those Action Groups, Actions, and Panels that have been assigned a **Check Running Mode** rule with a value of **Remove Features** will be executed.

3. The **Choose Product Features** panel opens, and you are prompted to uncheck the features that you want to uninstall.



4. Select the features to uninstall and click **Uninstall** to proceed.

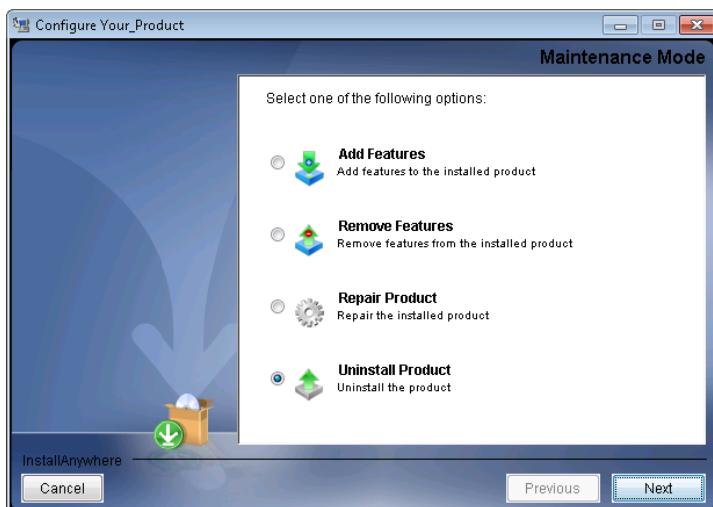
Repairing a Product User Experience

This section explains what an end user will see when they launch the Change Installation launcher and choose the **Repair Product** option



Task: *To repair an installed product:*

1. Open the Change Installation launcher, as described in [Initial Experience](#). The **Maintenance Mode** panel opens.



Chapter 5: Creating Basic Installers

Implementing Maintenance Mode

2. Select **Repair Product** and click **Next**. The Maintenance Mode runtime displays the panels in the **Repair Installation** Action Group on the **Pre-Install** Action List and proceeds as a regular installation.



Note • Only those Action Groups, Actions, and Panels that have been assigned a **Check Running Mode** rule with a value of **Repair Installation** will be executed.

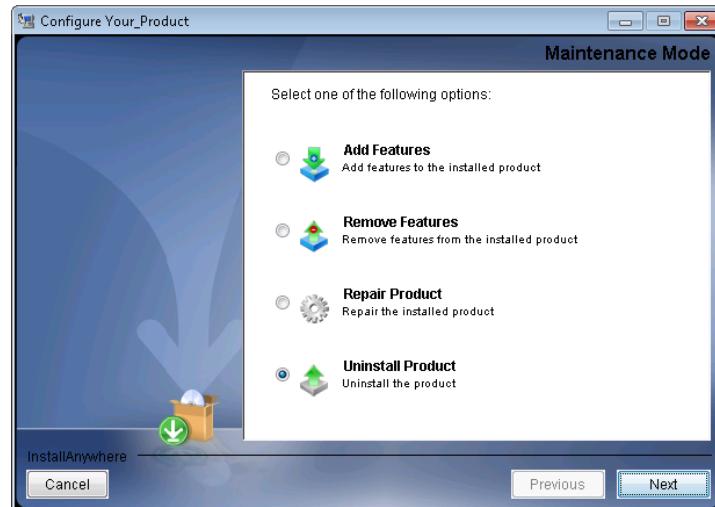
Uninstalling a Product User Experience

This section explains what an end user will see when they launch the Change Installation launcher and choose the **Uninstall Feature** option



Task: *To uninstall a product:*

1. Open the Change Installation launcher, as described in [Initial Experience](#). The **Maintenance Mode** panel opens.



2. Select **Uninstall Product** and click **Next**. The Maintenance Mode runtime displays the panels in the **Pre-Uninstallation** Action Group on the **Pre-Uninstall** Action List and proceeds as a regular uninstallation.



Note • Only those Action Groups, Actions, and Panels that have been assigned a **Check Running Mode** rule with a value of **Pre-Uninstallation** will be executed.

Maintenance Mode User Experience on a Machine With Multiple Installed Instances

As described in [Specifying Instance Management Options](#), you can create an installer that permits an end user to install multiple instances of a product on the same machine.

If multiple instances of a product are installed on the same machine, when the end user launches Maintenance Mode, the **Manage Instances** dialog box opens, where they are prompted to either choose to install a new instance or select the instance that they want to modify.

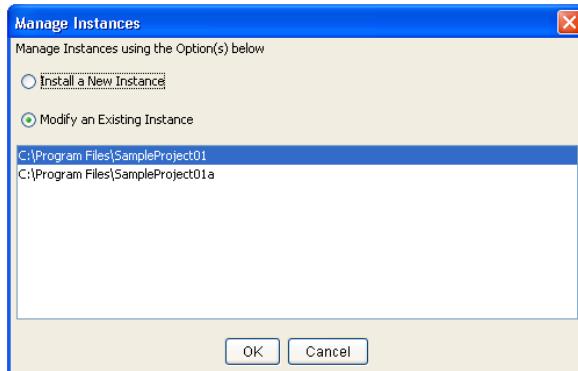


Figure 5-17: Manage Instances Dialog Box

Customizing the Uninstaller



Edition • This feature is included with InstallAnywhere Enterprise Edition.

InstallAnywhere automatically creates an uninstaller for the project. The Uninstaller, much like the Installer, is a collection of panels, consoles, and actions. The standard Uninstaller uninstalls the application by executing each action's uninstallation procedure.

However, in some situations, you may want additional flexibility and have more control over how the uninstallation is performed. Therefore, you may want to use the **Uninstall** task in the Advanced Designer to customize the Uninstaller by adding, removing or changing some of the uninstall actions. For example, you may want to disable the uninstallation of an entire set of resources, rename files, copy and move files, display additional dialog messages, or execute some custom code at uninstall time.

Information about Uninstaller customization is presented in the following topics:

- [About Uninstaller Customization](#)
- [Adding Uninstall Categories and Actions to the Uninstall Task](#)
- [Preventing an Uninstall Category From Being Uninstalled](#)
- [Reordering Uninstall Categories in the Uninstall Task](#)

About Uninstaller Customization



Edition • This feature is included with *InstallAnywhere Enterprise Edition*.

This section lists some benefits of customizing the Uninstaller and provides an overview of the **Uninstall** task on the Advanced Designer.

- [Benefits of Customizing the Uninstaller](#)
- [Overview of the Uninstall Task](#)

Benefits of Customizing the Uninstaller

By customizing the **Uninstall** phase, you are able to configure how uninstallation will occur by adding, changing or even removing steps of it. That level of flexibility enables you to address scenarios such as the following:

- **Prevent uninstallation of some resources**—You can choose to prevent the uninstallation of specific resources. For example, you could choose to keep some registry entries unchanged without uninstalling them for future reference.
- **Customize the uninstall to provide some actions/custom code before a category is uninstalled**—For example, before having the Uninstaller undeploy a WAR file from an application server, you can choose to run a script to stop the application server.
- **Reorder uninstallation steps**—You can choose to reorder the uninstallation steps. This enables you to run scripts before any resource has been uninstalled. For example, you could choose to run scripts to clean up external resources that were installed in a location other than the installation directory—such as resources that were generated during installation—prior to beginning the main uninstallation steps.
- **Uninstall Merge Modules**—You can choose to uninstall Merge Modules from the base installation during the **Uninstall** phase, rather than adding an **Execute Uninstaller** action to end of the **Pre-Uninstall** phase to perform this task.

Overview of the Uninstall Task

You can customize the uninstaller by using the **Uninstall** task in the Advanced Designer. On the **Uninstall** task, actions are grouped into **Uninstall Categories**.



Figure 5-18: Uninstall Categories and Actions on the Uninstall Task

By default, the following Uninstall Categories are created:

- Files
- LaunchAnywhere
- Shortcuts/Links/Aliases
- Registry Entries
- Native Packages
- Folders
- Others Category

By default, each of these Uninstall Categories contains one Uninstall action that would run the uninstallation of the category. You can reorder items in this list, add additional [General Actions](#), remove items from the list, and re-add deleted items. To add custom uninstallation behavior, you can add additional custom Uninstall Categories.

Adding Uninstall Categories and Actions to the Uninstall Task



Edition • This feature is included with InstallAnywhere Enterprise Edition.

To customize the **Uninstall** phase, you can add **Uninstall**, **General** and **Plug-Ins** actions and group those actions into Uninstall Categories.

- [About Adding Uninstall Categories and Actions](#)
- [Adding Categories and Actions](#)
- [Available Uninstall Actions](#)

About Adding Uninstall Categories and Actions

Regarding adding Uninstall Categories and actions, note the following:

- **Only Uninstall, General, and Plug-Ins actions can be added**—Only **Uninstall Actions**, **General Actions**, and **Plug-In Actions** can be added to Uninstall Categories because they are not bound to Magic Folders and they do not change the product registry. To show messages to the user during a console installation, one Console action—**Show Message Console ‘Dialog’**—is also available.
- **Each Uninstall Category and Uninstall action can be added only once**—Each of the Uninstall Categories and Uninstall actions can be added to the **Uninstall** Visual Tree only once. If they are added more than one time, the second instance is ignored.
- **Rules can be assigned**—You can assign rules to Uninstall Categories and actions in those categories to prevent them from being executed/uninstalled.
- **Uninstall Categories and Uninstall actions cannot be assigned to Components or Features**—While you are permitted to assign **General**, **Plug-Ins**, and **Console** actions in the **Uninstall** task to Components or Features, you are not permitted to assign **Uninstall Categories** or **Uninstall** actions to Components or Features. This is because **Uninstall Categories** and **Execute Uninstall Actions** are special actions which are not Feature/Component-specific. These are shared across all Components and Features.

Adding Categories and Actions

To add an Uninstall Category or action to the Uninstall task, perform the following steps.



Task:

To add an Uninstall Category or action to the Uninstall task:

1. Open the **Uninstall** task in the Advanced Designer. The Visual Tree of the Uninstall task lists Uninstall Categories and actions previously added.



Note • By default, all of the Uninstall actions are listed in the Visual Tree of the **Uninstall** task. If you delete one of these Uninstall actions and want to re-add it later, you can select it from the **Uninstall** tab of the **Choose an Action** dialog box.

2. To add an Uninstall Category, perform the following steps:
 - a. Click **Add Action**. The **Choose an Action** dialog box opens, listing four tabs: **Uninstall**, **General**, **Plug-Ins**, and **Console**.
 - b. On the **Uninstall** tab, select **Uninstall Category** and click **Add**. The new Uninstall Category is added to the list.
 - c. In the **Properties** tab of the **Uninstall Category** customizer, enter a name for the new category in the **Category Name** text box.
 - d. With the new Uninstall Category selected, using the arrow keys to move the category up or down in the Visual Tree.
3. To add an action, perform the following steps:
 - a. Select the Uninstall Category that you want to add an action to.
 - b. Click **Add Action**. The **Choose an Action** dialog box opens.
 - c. Select an action on the **Uninstall**, **General**, **Plug-Ins**, or **Console** tab and click **Add**. The new action is added to the Visual Tree.



Note • Each of the **Uninstall** actions can be added to the Uninstall Visual Tree only once. If they are added more than one time, the second instance is ignored.

- d. If you added a **General**, **Plug-Ins**, or **Console** action, you can assign it to a Product Feature or Component by selecting the appropriate check boxes. See [Assigning Actions to Product Features and Components](#).
4. Save and build the project.

Available Uninstall Actions

The following Uninstall actions are available in the **Uninstall** task.

Table 5-10 • Uninstall Actions

Action	Description
Uninstall Category	Groups sets of uninstall actions.
Uninstall AIX Entries	Uninstalls all entries made in the SWVPD registry of the AIX machine.
Uninstall DB Scripts	Runs the Uninstall scripts mentioned in the Run SQL action against the databases.
Uninstall Files	Uninstalls all of the files installed or created by the installer during installation.
Uninstall Folders	Uninstalls all of the folders installed or created by the installer during installation.
Uninstall JEE Archives	Undeploys WAR/EAR files deployed during installation.
Uninstall LaunchAnywheres	Uninstalls all of the LaunchAnywheres installed or created by the installer during installation.
Uninstall RAIR Entries	Uninstalls the RAIR component entries in i5/OS machines.
Uninstall Registry Entries	Uninstalls all of the registry entries installed or created by the installer during installation.
Uninstall RPM Packages	Uninstalls all RPM entries made by the installer.
Uninstall Shortcuts/Links/Aliases	Uninstalls all of the shortcuts, links, and aliases installed or created by the installer during installation.

Assigning Actions to Product Features and Components

To assign actions on the **Uninstall** task to product features or components, use the **Assign to** option.



Note • While you are permitted to assign **General**, **Plug-Ins**, and **Console** actions in the **Uninstall** task to Components or Features, you are not permitted to assign **Uninstall Categories** or **Uninstall** actions to Components or Features. This is because **Uninstall Categories** and **Execute Uninstall Actions** are special actions which are not Feature/Component-specific. These are shared across all Components and Features.



Task: *To assign an action to a product feature or component:*

1. Open a project in the Advanced Designer and open the **Uninstall** task.
2. Select an action in the Visual Tree.
3. From the **Assign to** list above the Visual Tree, select **Product Features** or **Components**. All of the project features or components are listed.
4. Select the product features or components that you want to assign the selected action to.
5. Save the project.

Preventing an Uninstall Category From Being Uninstalled



Edition • This feature is included with InstallAnywhere Enterprise Edition.

When an end user launches the Uninstaller, it displays a list of the Uninstall Categories and starts uninstalling all of them one by one. If you want to prevent an entire Uninstall Category of actions from being uninstalled during uninstallation, you can remove an action from that Uninstall Category so that that category is skipped during uninstallation, or you can assign a rule so that the Uninstall Category is not executed.

Removing the Uninstall Action from an Uninstall Category

To remove an Uninstall Action from an Uninstall Category, to prevent all of the actions in that category from being uninstalled during uninstallation, perform the following steps:



Task: *To remove an Uninstall Action from an Uninstall Category:*

1. Open the **Uninstall** task in the Advanced Designer. The Visual Tree of the **Uninstall** task lists Uninstall Categories and the **Uninstall** and **General** actions previously added.
2. Locate and select the **Uninstall** action in the Uninstall Category that you want to prevent from being uninstalled and click **Remove**. The **Uninstall** action is no longer listed in the Visual Tree.
3. Save and build the project.

Assigning a Rule

To assign a rule to an Uninstall Category, to prevent all of the actions in that category from being uninstalled during uninstallation, perform the following steps:



Task: *To assign a Rule to an Uninstall Category:*

1. Open the **Uninstall** task in the Advanced Designer. The Visual Tree of the **Uninstall** task lists Uninstall Categories and the **Uninstall** and **General** actions previously added.
2. Locate and select the Uninstall Category that you want to prevent from being uninstalled.
3. Open the **Rules** tab and click **Add Rule**. The **Choose a rule** dialog box opens.
4. Select a rule and click **Add**. The rule is now listed in the **Rules List** and its customizer is now displayed.
5. Specify the rule conditions that must be met in order for this Uninstall Category to be executed.
6. Save the project.

Reordering Uninstall Categories in the Uninstall Task



Edition • This feature is included with InstallAnywhere Enterprise Edition.

You can choose to reorder the Uninstall Categories in the Uninstall task Visual Tree. This will change the order in which these items are uninstalled.



Task: *To reorder Uninstall Categories in the Uninstall task:*

1. Open the **Uninstall** task in the Advanced Designer. The Visual Tree of the **Uninstall** task lists Uninstall Categories and the **Uninstall** and **General** actions previously added.
2. Select one of the Uninstall Categories.
3. Move it up or down in the list by using the arrow buttons.
4. Save and build the project.

Building Installers

The following topics discuss the steps necessary to complete various installer-building procedures.

Table 5-11 • Building Installers Procedures

Topic	Description
Creating and Editing Build Configurations	Explains how to use the Build Configurations tab of the Build task to create multiple Build Configurations, each representing how the installer will be built for particular set of locales, platforms, files, build distributions, JVMs, and other settings.

Table 5-11 • Building Installers Procedures

Topic	Description
Defining Build Targets	Explains how to use the Build Targets subtab of the Build Configurations tab of the Build task to create build targets.
Setting Build Distribution Options	Explains how to use the Distribution subtab of the Build Configurations tab of the Build task to create Web installers, CD-ROM/DVD installers, and Merge Modules
Building Installers Using Build Configurations	Explains how to generate installers by building a project's specified Build Configurations.
Testing Installers	Explains how to test a project's built installers.



Note • For information about building installers with the Project Wizard interface, see [Build Installer](#) and [Building an Installer Using the Project Wizard](#).

Creating and Editing Build Configurations

Each InstallAnywhere project can have multiple Build Configurations, each representing how the installer will be built for particular set of platforms, files, build distributions, JVMs, locales, and other settings. You can create and modify Build Configurations on the **Build Configurations** tab of the Advanced Designer's **Build** task. On this tab, you can add, rename, copy, or delete a Build Configuration.

This section explains how to perform the following tasks:

- [About Build Configurations](#)
- [Creating a New Build Configuration](#)
- [Creating Migrated Build Configurations](#)
- [Renaming a Build Configuration](#)
- [Copying a Build Configuration](#)
- [Removing a Build Configuration](#)
- [Adding a Build Configuration to the Project Build](#)
- [Using Tags to Customize Build Configurations](#)

About Build Configurations



Edition • The Build Configurations feature is available in InstallAnywhere Enterprise Edition.

In some situations, you may need to generate multiple installers for the same application, each with a slightly different configuration. For example:

- You may want to provide different locale support for different platforms.
- You may want to create two different editions of an application—with both editions containing most of the same content but diverging in a few select files.

One way you could do this would be to create multiple InstallAnywhere projects, each with different settings. However, a much more efficient way of accomplishing this is to use the InstallAnywhere Build Configuration feature.

Benefits of Using Build Configurations

You can create multiple Build Configurations in the same InstallAnywhere project, each with different settings. Each Build Configuration can define how an installer will be built for particular set of platforms, build distributions, JVMs, locales, and other settings. You can store as many Build Configurations with a project as necessary.

When building your InstallAnywhere project, you can choose to build a selected Build Configuration, all Build Configurations that exist in the project, or just a specified set of Build Configurations.

For each Build Configuration that you build, a separate installer, each in its own directory, is generated.

Default Build Configurations for New and Migrated Projects

A new InstallAnywhere project will automatically have a Build Configuration named [Default Configuration](#).

If you migrate an InstallAnywhere project created using a previous version or edition of InstallAnywhere where the Build Configuration features was unavailable, the migrated project will automatically have a Build Configuration named [Migrated Configuration](#), which will have the same build settings that were found in the original project.



Note • If you migrate an InstallShield MultiPlatform project to InstallAnywhere 2010 or later using a Project Manifest, a Build Configuration named [Migrated Configuration](#) is created.

Build Configuration Options in the InstallAnywhere User Interface

You define Build Configurations on the **Build Configurations** tab of the Advanced Designer **Build** task.

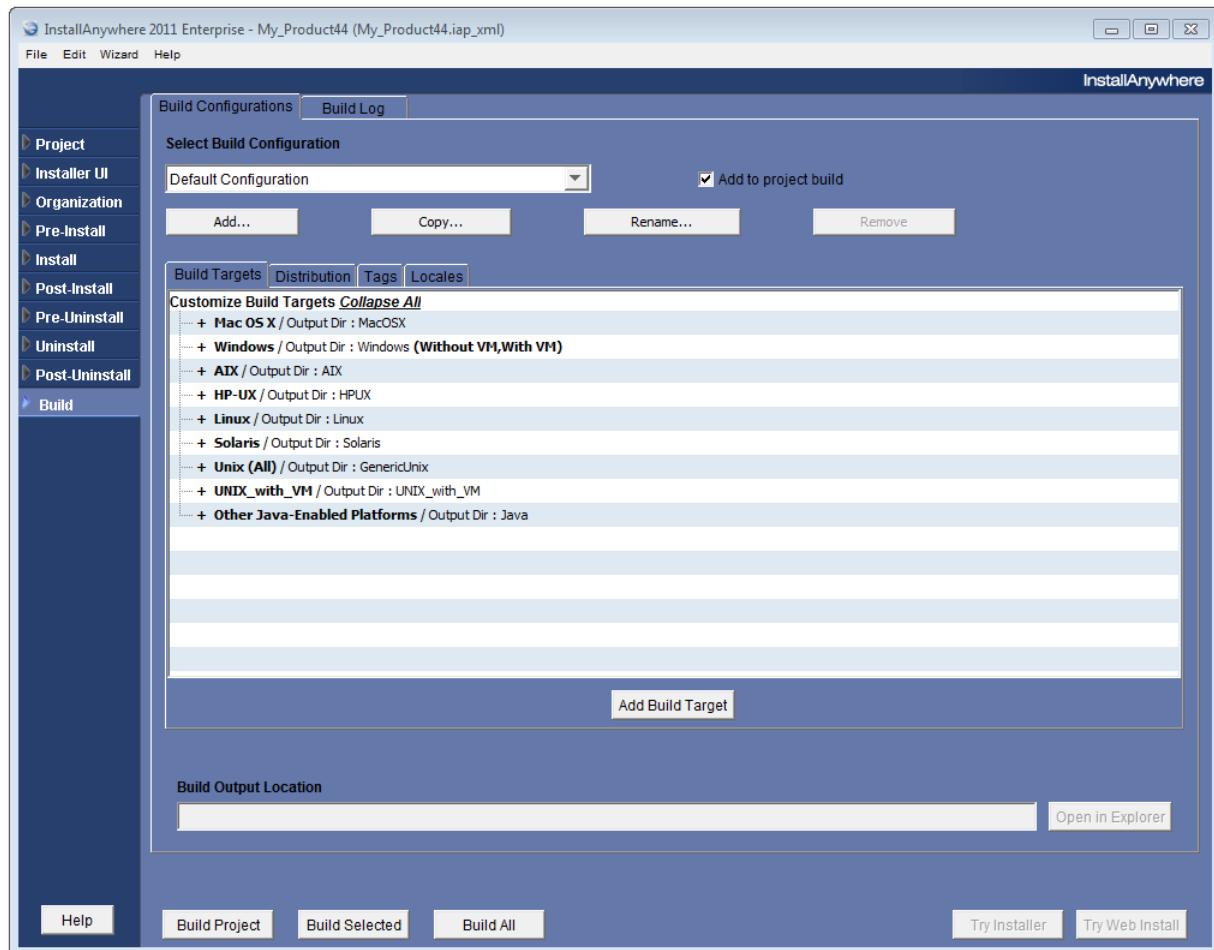


Figure 5-19: Build Configurations Tab of the Build Task

Chapter 5: Creating Basic Installers

Building Installers

When using the Project Wizard interface, you cannot create or edit Build Configurations, but you can select an existing Build Configuration to build by making a selection from the **Select Build Configuration** list on the **Build Installer** panel.

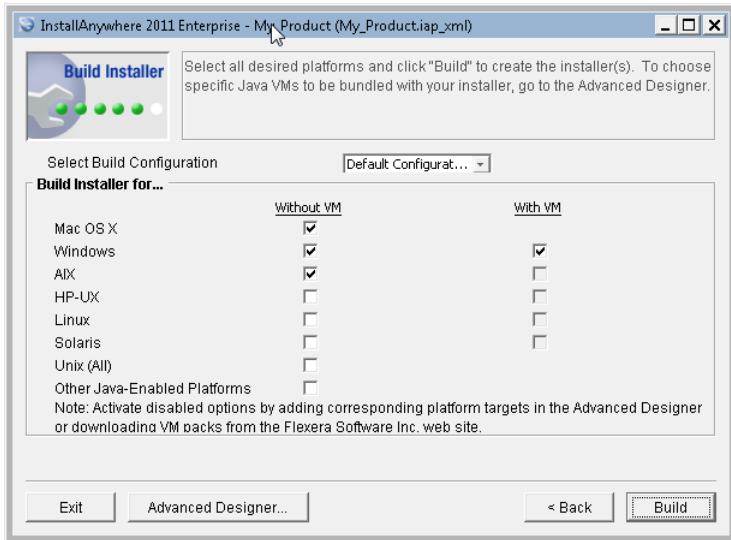


Figure 5-20: Select Build Configuration Option on the Build Installer Panel of the Project Wizard

Using Build Configuration Tags

You can use Tags to bundle different sets of actions, panels, features, and components into Build Configurations. Using Tags involves three main steps:

- **Create Tags**—First, create a set of Tags that you can use to group project elements.
- **Assign Tags to project elements**—Next, assign an appropriate Tag to all of those project elements that you want to include in some Build Configurations but exclude from others.
- **Associate Tags with Build Configurations**—Then, for each Build Configuration, specify which Tags you want to include and which Tags you want to exclude.

When a Build Configuration that has been customized using Tags is built, it results in an installer that includes:

- **All untagged project elements**—The installer will include all untagged project elements: project elements that do not have any Tags listed in the **Associated Tags** list on the **Tags** subtab of its customizer.
- **Project elements associated with a Tag that is also associated with the Build Configuration**—The installer will include those project elements that have been associated with one or more Tags, one of which is also associated with the selected Build Configuration.

All other project elements will be excluded. In other words, if a project element is associated with one or more Tags, none of which is associated with the selected Build Configuration, that project element will be excluded from the installer.



Note • For more information, see [Using Tags to Customize Build Configurations](#).



Important • If a project element is not associated with any Tag, then that project element is included in all Build Configurations by default.



Important • When you create a new Tag, it is automatically associated with all existing Build Configurations.

Locale Settings

For each Build Configuration, you can specify the languages for which the installer will be created. Languages are selected on the **Locales** subtab of the **Build Configurations** tab of the Advanced Designer **Build** task. A locale is enabled when it is checked.

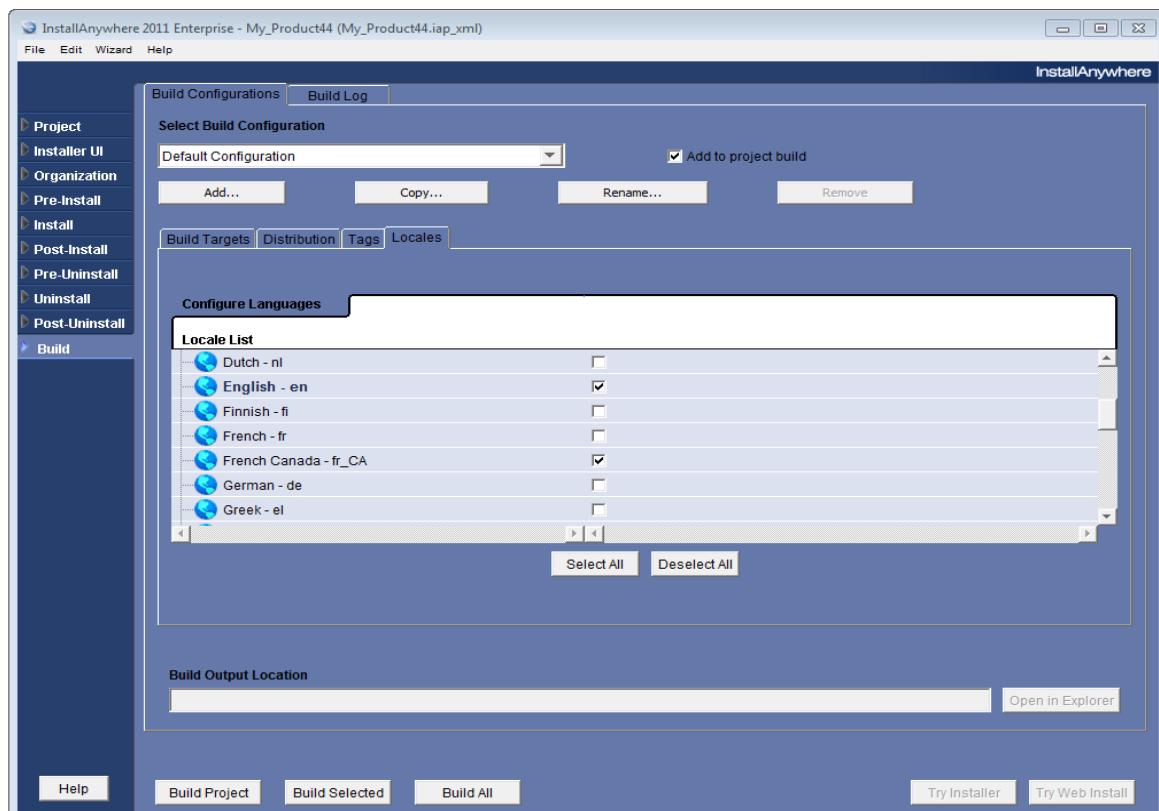


Figure 5-21: Build Configurations Tab of the Build Task / Locales Subtab



Important • In previous releases, Locales could only be specified on a project-level basis on the **Project > Locales** subtask, and these Locale settings were applied to all installers created by this project. Starting with InstallAnywhere 2010, you specify Locale settings on the **Locales** subtab of the **Build Configurations** tab, which enables you to specify different Locale settings for each Build Configuration.

Creating a New Build Configuration

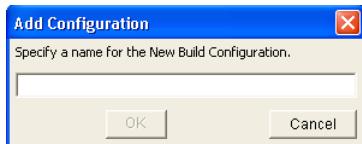
Each InstallAnywhere project can have multiple Build Configurations, each representing how the installer will be built for particular set of platforms, files, build distributions, JVMs, and other settings.

To add a new Build Configuration to a project, perform the following steps.



Task: *To add a new Build Configuration to a project:*

1. Open an installer project in the Advanced Designer.
2. Open the **Build** task.
3. On the **Build Configurations** tab, click **Add**. The **Add Configuration** dialog box opens, prompting you to enter a name for the new Build Configuration.



4. Enter a name and click **OK**.
 - The name must be 60 characters or less.
 - Do not use an asterisk (*), question mark (?), period (.), back slash (\), or forward slash(/) in the name.
- The new Build Configuration is created and is selected in the **Select Build Configuration** list, but it is not yet saved to the project.
5. Specify settings for this Build Configuration, as described in [Defining Build Targets](#), [Setting Build Distribution Options](#), [Using Tags to Customize Build Configurations](#), and [Specifying Locale Settings](#).
6. On the **File** menu, click **Save** to save this new Build Configuration into the project.

Creating Migrated Build Configurations

When a project created using a version of InstallAnywhere prior to InstallAnywhere 2010 is opened, all of the build settings of the earlier project are migrated to the new Build Configuration model. These migrated build settings are saved as a migrated Build Configuration named **Migrated Configuration**.

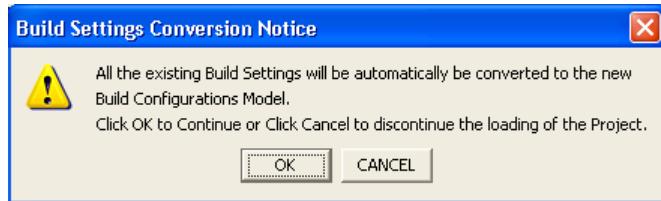
To create a migrated Build Configuration, perform the following steps:



Task: *To create a migrated Build Configuration:*

1. Launch InstallAnywhere. The InstallAnywhere **About** dialog box opens.
2. Click **Open Existing**. The **Open Project File** dialog box opens.
3. Locate and select the project that was created using an earlier version of InstallAnywhere.

4. Click **Open**. The **Build Settings Conversion Notice** dialog box opens, informing you that the existing build settings will automatically be converted to use the new Build Configurations model.



5. Click **OK**. The project opens in the Advanced Designer, and **Migrated Configuration** is selected in the **Select Build Configuration** list on the **Build Configurations** tab of the **Build** task.



Note • *The Add to project build option for the migrated Build Configuration is automatically selected.*

6. Save the project.
7. If you want to modify the settings for this migrated Build Configuration, perform the steps described in [Defining Build Targets](#), [Setting Build Distribution Options](#), [Using Tags to Customize Build Configurations](#), and [Specifying Locale Settings](#).

Renaming a Build Configuration

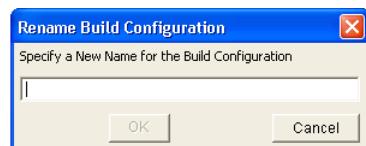
You can edit a Build Configuration's name by performing the following steps:



Task:

To rename a Build Configuration:

1. Open an installer project in the Advanced Designer.
2. Open the **Build** task.
3. On the **Build Configurations** tab, select the Build Configuration that you want to rename from the **Select Build Configuration** list.
4. Click **Rename**. The **Rename Build Configuration** dialog box opens.



5. Enter a new name for this Build Configuration and click **OK**. The Build Configuration is renamed.
6. Save the project.

Copying a Build Configuration

You can copy an existing Build Configuration and then make additional customizations. To copy an existing Build Configuration, perform the following steps.

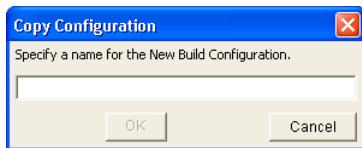


Tip • You could create one Build Configuration with your desired default settings to serve as a template, and then use the **Copy Configuration** feature to create multiple new Build Configurations based on those default settings.



Task: *To copy a Build Configuration:*

1. Open an installer project in the Advanced Designer.
2. Open the **Build** task.
3. On the **Build Configurations** tab, select the Build Configuration that you want to copy from the **Select Build Configuration** list.
4. Click **Copy**. The **Copy Configuration** dialog box opens.



5. Enter a name for the new Build Configuration and click **OK**. The new Build Configuration is now selected in the **Select Build Configuration** list.



Note • The original Build Configuration is not modified.

6. On the **File** menu, click **Save** to save the new Build Configuration into the project.
7. If you want to modify the settings for this copied Build Configuration, perform the steps described in [Defining Build Targets](#), [Setting Build Distribution Options](#), [Using Tags to Customize Build Configurations](#), and [Specifying Locale Settings](#).

Removing a Build Configuration

To delete an existing Build Configuration, perform the following steps.



Task: **To delete a Build Configuration:**

1. Open an installer project in the Advanced Designer.
2. Open the **Build** task.
3. On the **Build Configurations** tab, select the Build Configuration that you want to delete from the **Select Build Configuration** list.
4. Click **Remove**. You are prompted to confirm the deletion.
5. Click **Yes**. The Build Configuration is deleted.



Note • A project must have at least one Build Configuration. If a project contains only one Build Configuration, the **Remove** button on the **Build Configurations** tab will be disabled.

Adding a Build Configuration to the Project Build

You can add a Build Configuration to the project build by selecting its **Add to project build** option. If a Build Configuration is added to a project build, that Build Configuration will be built each time one of the following occurs:

- You click the **Build Project** or **Build All** buttons on the **Build Configurations** tab of the **Build** task.
- You build the project from the command line without specifying any Build Configurations, such as:
`build.exe MyProject.iap_xml`

To add a Build Configuration to the project build, perform the following steps.



Task: **To add a Build Configuration to the project build:**

1. Open an installer project in the Advanced Designer.
2. Open the **Build** task.
3. On the **Build Configurations** tab, select the Build Configuration that you want to add to the project build from the **Select Build Configuration** list.
4. Select the **Add to project build** option.
5. Select **Save** on the **File** menu to save the project.



Note • If you click the **Build All** button on the **Build Configurations** tab (or use the `-all` command line argument), all existing Build Configurations for that project will be built regardless of whether their **Add to project build** option has been selected.

Using Tags to Customize Build Configurations



Edition • This feature is included with InstallAnywhere Enterprise Edition.

You can use Tags to bundle different sets of actions, panels, features, and components with Build Configurations. Using Tags involves three main steps:

- **Step 1: Create Tags**—Define a set of Tags to use with Build Configurations. See [Creating New Tags](#).
- **Step 2: Assign Tags to Project Elements**—Assign an appropriate Tag to all of those project elements that you want to include in some Build Configurations but exclude from others. See [Assigning Tags to Project Elements](#).
- **Step 3: Associate Tags With Build Configurations**—For each Build Configuration, specify which Tags you want to include and which Tags you want to exclude. See [Associating Tags to Build Configurations](#).

To learn how to use Tags to customize Build Configurations, review the following topics:

Table 5-12 • Steps to Customize Build Configurations Using Tags

Step	Description
Determining Whether a Project Element is Included in a Build Configuration	Explains how to determine whether a project element (action, panel, feature, component) is included in a Build Configuration.
Creating New Tags	Define a set of Tags to use with Build Configurations.
Assigning Tags to Project Elements	Assign an appropriate Tag to all of those project elements that you want to include in some Build Configurations but exclude from others.
Associating Tags to Build Configurations	For each Build Configuration, specify which Tags you want to include and which Tags you want to exclude.
Searching for Tags	Search a project to locate all references to a given Tag.
Importing Tags from Merge Modules	Explains how to import Tags from static merge modules.

Determining Whether a Project Element is Included in a Build Configuration

Whether a project element (action, panel, feature, component) is included in a Build Configuration depends what is selected on the **Tags** subtab of each project element's customizer.

- [Tagging Project Elements](#)
- [Rules for Applying Build Tags to Build Configurations](#)

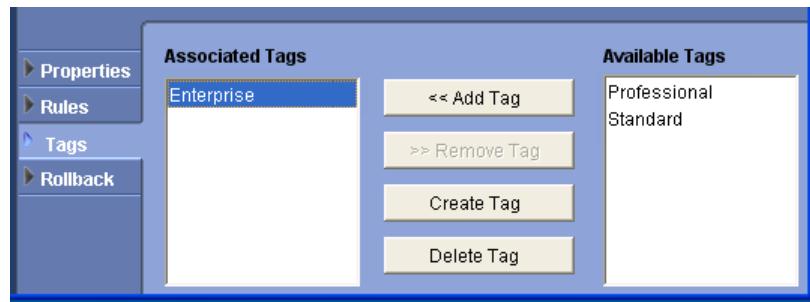
Tagging Project Elements

All project elements are either *tagged* or *untagged*:

- **Untagged**—A project element is untagged when it does not have any Tags listed in the **Associated Tags** list:



- **Tagged**—A project element is tagged when it has one or more Tags listed in the **Associated Tags** list:



Rules for Applying Build Tags to Build Configurations

When a Build Configuration is built, the project elements that are included depends upon both whether the Build Configuration has any associated Tags, and which (if any) Tags are associated with a project element.

When building a Build Configuration, follow these rules to determine which project elements are included in the resulting installer:

Table 5-13 • Rules for Including Project Elements in a Build Configuration

Build Configuration	Included Project Elements	Excluded Project Elements
No Associated Tags	<p>All project elements (both tagged and untagged).</p>  <p>Important • When a Build Configuration that has no associated Tags is built, it results in an installer that includes all project elements, both those with associated Tags and those without any associated Tags.</p>	None
With Tags	<ul style="list-style-type: none"> • All untagged project elements, AND • Project elements that are tagged with one or more of the Tags that are associated with the Build Configuration.  <p>Note • For example, if a project element is associated with a Tag named Standard, but the Build Configuration is associated with a Tag named Professional (but not Standard), the project element will be excluded from the installer.</p>	Project elements that are tagged with non-matching Tags (Tags that are not associated with the Build Configuration).  <p>Note • For example, if a project element is associated with a Tag named Professional, and the Build Configuration is also associated with a Tag named Professional, the project element will be included in the installer.</p>

To summarize, the following rules are enforced:

- A project element which has no Tags associated with it will be included in the installer for all Build Configurations.
- A project element which has Tags associated will be included in the installer if, and only if, the Build Configuration has at least one matching Tag.
- A Build Configuration which has no associated Tags includes all project elements.



Important • When you create a new Tag, it is automatically associated with all existing Build Configurations.

Creating New Tags



Edition • This feature is included with InstallAnywhere Enterprise Edition.

You can create new project Tags on the **Project > Advanced** subtask or on the **Tags** subtab of a project element customizer.

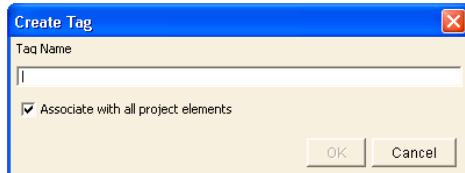


Note • Each Tag can be associated with multiple Build Configurations.



Task: **To create a new Build Configuration tag:**

1. Do one of the following:
 - Open the **Project > Advanced** subtask of the Advanced Designer.
 - Open the **Tags** subtab of a project element customizer.
2. In the **Manage Tags** area, click **Create Tag**. The **Create Tag** dialog box opens.



3. Enter a name for the new Tag.
4. If you want this new Tag to be automatically associated with all existing project elements, select the **Associate with all project elements** option.



Note • If you select the **Associate with all project elements** option when creating a new Tag, that Tag is automatically associated with all **existing** project elements, but it is not automatically associated with additional project elements that are added to the project from that point forward.

5. Click **OK** to close the **Create Tag** dialog box.
6. Select **Save** on the **File** menu to save the project.

Assigning Tags to Project Elements



Edition • This feature is included with InstallAnywhere Enterprise Edition.

You can use Tags to bundle different sets of actions, panels, features, and components with Build Configurations. After creating a set of Tags for a project, you need to assign an appropriate Tag to all of those project elements that you want to include in some Build Configurations but exclude from others.

When a group of project elements are associated with a given Tag, and then that Tag is associated with a Build Configuration, then all of those project elements are built for that Build Configuration.



Important • If a project element is not associated with any Tag, then that project element is included in all Build Configurations by default.

Considerations When Assigning Build Configuration Tags

In a project, Features take precedence over Components, which take precedence over Actions. When you are assigning Tags to project elements, to avoid conflicts, you should pick one project element level to filter by: Features, Components, or Actions. Mixing and matching is generally not recommended.

For example, if you assign a Tag to a Feature, do not assign a Tag to an action installed with that Feature. If you attempt to assign a Tag to a project element that “belongs” to a higher-level project element that is already assigned a different Tag, a warning message will appear on the **Tags** subtab of that project element’s customizer.

Assigning a Tag to a Project Element

To assign a Tag to a project element, perform the following steps:

**Task:****To assign a Tag to a project element:**

1. Open a project in the Advanced Designer.
2. Select a project element in the **Pre-Install**, **Install**, **Post-Install**, **Pre-Uninstall**, **Uninstall**, or **Post-Uninstall** tasks.
3. In the project element customizer, open the **Tags** subtab. All of the Tags defined in this project are listed. Tags associated with this project element are listed in the **Associated Tags** list, while those Tags not associated with this project element are listed in the **Available Tags** list.
4. To associate a Tag with this project element, select a Tag in the **Available Tags** list and then click **Add Tag** to move it to the **Associated Tags** list.
5. To remove a Tag from the **Associated Tags** list, select it then click **Remove Tag**.
6. Save the project.

Associating Tags to Build Configurations



Edition • This feature is included with InstallAnywhere Enterprise Edition.

You associate Tags with Build Configurations on the **Tags** subtab of the **Build** task’s **Build Configurations** tab.



Task: *To associate Tags to a Build Configuration:*

1. Open an installer project in the Advanced Designer.
2. Open the **Build** task.
3. On the **Build Configurations** tab, select the Build Configuration that you want to edit from the **Select Build Configuration** list.
4. Open the **Tags** subtab. All of the Tags defined in this project are listed. Tags associated with this Build Configuration are listed in the **Associated Tags** list, while those Tags not associated with this Build Configuration are listed in the **Available Tags** list.
5. To associate a Tag with a Build Configuration, select a Tag in the **Available Tags** list and then click **Add Tag** to move it to the **Associated Tags** list.
6. To remove a Tag from the **Associated Tags** list, select it then click **Remove Tag**.
7. Save the project.

Searching for Tags



Edition • This feature is included with InstallAnywhere Enterprise Edition.

You can search a project to locate all references to a given Build Configuration Tag. You can choose to search for a selected Tag in any of the installation tasks/phases (Pre-Install, Install, Post-Install, Pre-Uninstall, Uninstall, Post-Uninstall) and can choose to search Features and/or Components. Search results are displayed on the **Search Results** dialog box.



Task: *To search for a Build Configuration Tag:*

1. Open a project in the Advanced Designer.
2. On the **File** menu, click **Search** (or press Ctrl + F). The **Search Results** dialog box opens.
3. Open the **Tags** tab.
4. From the **Select Tag** list, select a Build Configuration Tag.
5. Under **Search for Selected Tag in**, select the installation phases (**Pre-Install**, **Install**, **Post-Install**, **Pre-Uninstall**, **Uninstall**, and/or **Post-Uninstall**) and product elements (**Features** and/or **Components**) that you want to search.
6. Click **Search**. Project elements associated with the selected Tag are listed, grouped by category.

Importing Tags from Merge Modules

You have the option of importing Tags from static merge modules. To import Tags from a static merge module, select the **Import Tags** option on the **InstallAnywhere Merge Module Import Assistant** dialog box.

- **Selected**—If this option is selected, the merge module's associated Tags will be imported into the parent project.
- **Not Selected**—If this option is not selected, the merge module's associated Tags will be ignored.



Note • The **InstallAnywhere Merge Module Import Assistant** dialog box opens when you click **Import Merge Module** on the **Organization > Modules** subtask and select a merge module to import.

Specifying Locale Settings

InstallAnywhere can generate localized installers when you select the locales you want to support. You specify locale settings for each Build Configuration on the **Locales** subtab of the **Build** task's **Build Configurations** tab.



Task: *To generate multi-language installers*

1. Open the **Build Configurations** tab of the **Build** task.
2. Select the Build Configuration you want to edit from the **Select Build Configuration** list.
3. Open the **Locales** subtab.
4. Use the check boxes to select languages.
5. Save the project.



Note • Additional localization may be required for resources, customized panels or consoles, and custom code.

Defining Build Targets

Build targets specify the different installers your project will build. Use the **Build Targets** subtab of the **Build Configurations** tab of the **Build** task to define build targets for your InstallAnywhere project.



Task: *To define a build target:*

1. In the Advanced Designer, click the **Build** task. The **Build Targets** subtab of the **Build Configurations** tab opens.
2. Under **Customize Build Targets**, click the plus sign next to the platform that you want to customize. The customizer for that platform opens.

3. For **Windows** and **UNIX** build targets, specify the following information in the customizer:

Control	Description
Target Name / Output Folder	<p>Enter the name for the build target with a description (typically the platform name) that distinguishes it from other installers the project builds.</p> <ul style="list-style-type: none">• Each build target label must have a unique name.• If you have two build targets for the same platform with the same label, InstallAnywhere appends <u>_1</u> to the label of the duplicate build target.
Platform Type	The platform type is automatically selected for each of the listed build targets, but if you are adding a new build target, you can select a different platform from the list.

Control	Description
Without VM	<p>Select this option if you do not want to include a Java virtual machine with this build target. If you select this option, you also need to select one of the following options to specify how the installer will find the VM it will use:</p> <ul style="list-style-type: none"> • Search for VM; if not found exit—Select this option to instruct the installer to search the target machine for a VM. If one is not found, the installer will exit. • Search for VM; if not found, download from URL—Select this option to instruct the installer to search the target machine for a VM. If one is not found, the installer will download a VM from the URL specified in the text box. <p> Optionally, you can also select the Verify downloaded VM with MD5 hash and enter the MD5 in the text box. After the installer is built and run, you will then be able to check the integrity of the downloaded VM.</p> <ul style="list-style-type: none"> • Don't search for VM; download from URL—Select this option to instruct the installer to always download a VM from the URL specified in the text box. <p> Optionally, you can also select the Verify downloaded VM with MD5 hash and enter the MD5 in the text box. After the installer is built and run, you will then be able to check the integrity of the downloaded VM.</p> <p></p> <p>Important • You can obtain VM packs by downloading them from the Flexera Software Web site (from the page that opens when you select Download Additional VM Packs from the Help menu). However, these VM packs are provided for your convenience only and should not be referenced by your installation project.</p> <p>Therefore, if you select the Search for VM; if not found, download from URL or Don't search for VM; download from URL options, do not specify a URL pointing to one of the VM packs on the www.flexerasoftware.com website. If you do, the build will fail. Instead, you should host the VM pack on your own file server or FTP server and specify the URL to that location.</p>

Control	Description
With VM	<p>Select this option if you want to include a Java virtual machine with this build target. If you select this option, you also need to select one of the following options to specify whether the installer should use the bundled VM if a VM is found on the target machine.</p> <ul style="list-style-type: none"> • Search for VM; if not found, use bundled VM—Select this option to instruct the installer to search the target machine for a VM. If one is not found, the installer will use the bundled VM. • Don't search for VM; use bundled VM—Select this option to instruct the installer to always use the bundled VM, even if a VM exists on the target machine.
VM to Bundle with Installer	<p>If you select the With VM option, select a bundled VM from this list.</p>  <p>Note • For more information, see Bundling a VM Pack.</p>
JVM Search Settings	<p>If the installer searches for a VM, it uses the JVM search instruction files specified in this list.</p> <ul style="list-style-type: none"> • Add—Click to open the Choose JVM Spec dialog box, where you are prompted to select a JVM spec from the list. • Remove—Click to remove the selected JVM spec from the list. • Up and Down—If more than one JVM spec is listed, use these buttons to list the specs in order of preference.  <p>Note • <i>InstallAnywhere provides several pre-written JVM Specs, which are listed on the Choose JVM Spec dialog box. You can also write your own JVM spec. For more information, see JVM Spec Files.</i></p>
Windows Installer Launcher Type	<p>(Windows only) Identify the type of launcher you want to use. Choose from Graphical Launcher and Console Launcher.</p>  <p>Note • <i>InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers will not run under those locales.</i></p>
Delete	<p>Click to delete the build target.</p>

For a **Mac OS X** build target, specify the following information in the customizer:

Control	Description
Target Name / Output Folder	<p>Enter the name for the build target with a description (typically the platform name) that distinguishes it from other installers the project builds.</p> <ul style="list-style-type: none"> Each build target label must have a unique name. If you have two build targets for the same platform with the same label, InstallAnywhere appends <u>_1</u> to the label of the duplicate build target.
Platform Type	<p>The platform type is automatically selected for each of the listed build targets, but if you are adding a new build target, you can select a different platform from the list.</p>  <p>Note • If you change the Platform Type from Mac OS X to another type, the controls displayed on the customizer will change.</p>
Without VM	<p>Select this option if you do not want to include a Java virtual machine with this build target.</p>
Delete	<p>Delete this build target from the project.</p>

- Click **File > Save** to store your changes in the project file.

Creating an Installer for a Customized Flavor of Unix

Some values in the **Platform Type** list of the platform customizers—**UNIX_with_VM**, **Unix (All)**, and **Other Java-Enabled Platforms**—function as customizable platforms that do not map to a specific OS.


Task:

To create an installer for a flavor of Unix that is not in the list of platforms:

- Under **Customize Build Targets**, click the plus sign next to **UNIX_with_VM**. The customizer for **UNIX_with_VM** opens.
- In the **Target Name / Output Folder** field, enter the name of the specific Unix platform for which you want to build.



Tip • This text field may also be used to differentiate between two variations of the same platform target.

- Under **VM Search Instructions**, select the **With VM** option and select a VM from the **VM to Bundle with Installer** list.

4. Specify other options, as described above.
5. Click **File > Save** to store your changes in the project file.

Setting Build Distribution Options

For each Build Configuration, you need to specify the types of installers that you want to create: Web installer, CD-ROM/DVD installer, or Merge Module. These distribution options are set in the Advanced Designer **Build** task on the **Distribution** subtab of the **Build Configurations** tab.

- [Creating Web Installers](#)
- [Creating CD-ROM/DVD Installers](#)
- [Creating Merge Modules](#)

Creating Web Installers

To set the Build Distribution options to create Web installers, perform the following steps.



Task: *To create Web installers:*

1. On the Advanced Designer **Build** task, open the **Build Configurations** tab.
2. From the **Select Build Configuration** list, select the Build Configuration that you want to specify distribution options for.
3. Open the **Distribution** subtab.
4. Under **Web Installer Options**, select **Build Web Installers**.
5. Specify whether or not to optimize installers for each build target. To produce a Web installer specific to each build target that exists on the **Build Targets** tab, select **Optimize Installer Size by Platform**.
6. From the **Web Page Displays In** list, choose the language in which the Web installer pages are rendered.
7. Save the project.

Creating CD-ROM/DVD Installers

To set the Build Distribution options to create CD-ROM/DVD installers, perform the following steps.



Task: *To create CD-ROM/DVD installers:*

1. On the Advanced Designer **Build** task, open the **Build Configurations** tab.
2. From the **Select Build Configuration** list, select the Build Configuration that you want to specify distribution options for.

3. Open the **Distribution** subtab.
4. Under **CD-ROM Installer Options**, select **Build CD-ROM Installers**.
5. Specify whether or not to optimize installers for each build target. To produce an installer specific to each build target that exists on the **Build Targets** tab, select **Optimize Installer Size by Platform**.
6. By default, the installer will be split to fit onto multiple 650 MB CDs. Optionally, if you want to change the media size, click **Change Disk Space and Name** to open the **Change Disk Space and Name** dialog box, and set media size and name options, as described in [Change Disk Space and Name Dialog Box](#).
7. Save the project.

Creating Merge Modules

For each Build Configuration, you can specify that you want to create a Merge Module installer by selecting the **Build Merge Module/Template** option on the **Distribution** subtab of the **Build Configurations** tab.

To set the Build Distribution options to create Merge Modules, perform the following steps.

**Task:****To create Merge Modules:**

1. On the Advanced Designer **Build** task, open the **Build Configurations** tab.
2. From the **Select Build Configuration** list, select the Build Configuration that you want to specify distribution options for.
3. Open the **Distribution** subtab.
4. Under **Merge Module/Template Options**, select **Build Merge Module/Template**.
5. If you want to produce Merge Modules specific to each build target that exists on the **Build Targets** tab, select the **Optimize Merge Module/Template Size by Platform** option. If you choose this option, separate Merge Modules will be created for each platform. Each will only contain the resources needed for that specific platform.



Important • *Do not use this option if Merge Modules will be imported into another installer. Importing a Merge Module requires a non-optimized Merge Module.*

6. If you want to create a Merge Module that can be installed but not altered, select the **Read Only** option.

By selecting the **Read Only** option, you are locking the Merge Module, which prevents it from being opened, used as a template, or being merged into an installer. The only way a read-only Merge Module can be added to an installer project is through the **Install Merge Module** action. Read-only Merge Modules can only be installed as a self-contained installer unit.

7. If you want to specify which variables in the Merge Module can be set by the parent installer and which variables in the parent installer can be set by the Merge Module, perform the steps in [Advertising Merge Module Installer Variables](#).
8. Save the project.

Advertising Merge Module Installer Variables

Advertised variables are used to inform the parent installer of settings required for a Merge Module's configuration, and to return information—such as status or return codes—back to the parent installer.

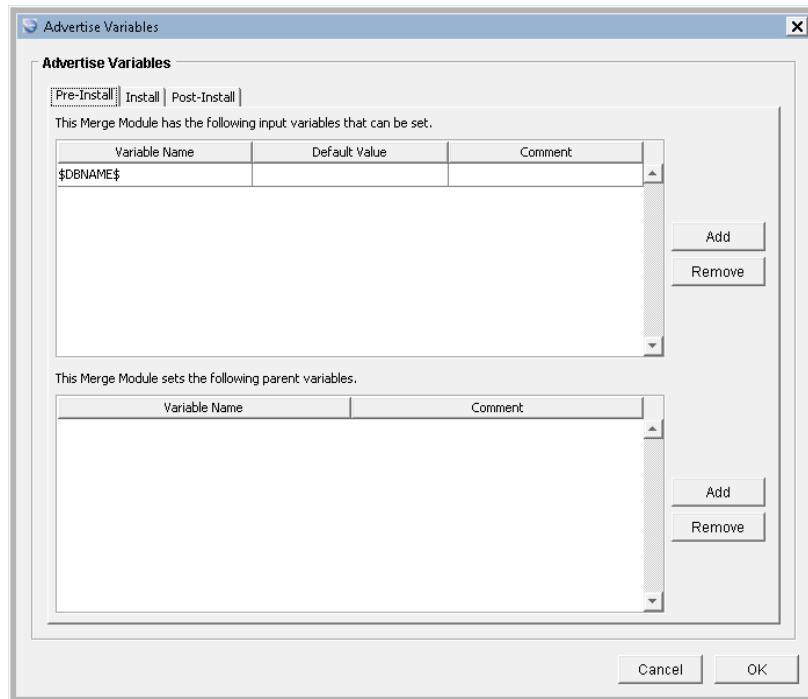
When building a Merge Module installer, you need to specify which variables in the Merge Module can be set by the parent installer and which variables in the parent installer can be set by the Merge Module. You do this on the **Advertise Variables** dialog box, which is opened by clicking **Advertise Variables** on the **Project > Variables** subtask.

To advertise the variables in a Merge Module installer that will be built from your InstallAnywhere project, perform the following steps:



Task: *To advertise Merge Module installer variables:*

1. Open the **Project > Variables** subtask.
2. Under **Advertise Variables**, click **Advertise Variables**. The **Advertise Variables** dialog box opens.



3. Select the appropriate tab to indicate the task that you want to advertise variables for: **Pre-Install**, **Install**, or **Post-Install**.



Note • Advertised variables are defined by task:

- Variables set as advertised as **Pre-Install** will only affect the Dynamic Merge Module in **Pre-Install**.
 - The **Install Merge Module** action only pays attention to Advertised Variables set in the **Install** task.
4. To advertise an input variable so that it will be visible in the parent project (the project that is going to install this Merge Module), click **Add** next to the input variables list to add an entry to the list, and then enter the following information:

Option	Description
Variable Name	Enter the name of a variable that has been defined in one of the panels of this project, using the syntax \$VARIABLE_NAME\$.
Default Value	If you want to hard code a value for this variable, enter text here. If you want the value of this variable to be entered by the end user during installation, leave this field blank.
Comment	Enter text to describe the purpose of this variable.

5. To advertise a variable in the parent project (the project that is going to install this Merge Module) so that it will be visible in this project, click **Add** next to the parent variables list to add an entry to the list, and then enter the following information:

Option	Description
Variable Name	Enter the name of a variable that you want to pass to the parent project using the syntax \$VARIABLE_NAME\$.
Comment	Enter text to describe the purpose of this variable.

6. Save the project.



Note • For each Build Configuration, you can specify if you want to create a Merge Module installer as part of the build output by opening the **Build** task and selecting the **Build Merge Module/Template** option on the **Distribution** subtab of the **Build Configurations** tab.

Variable Propagation from Merge Module to Parent Installer

Variables usually do not propagate from the merge module to the parent automatically. For this to occur, the variables have to be advertised in the respective phases of installation.

In general, \$INSTALL_SUCCESS\$ and \$RESTART_NEEDED\$ are two status variables which are used in the Install Complete panel.

- **\$INSTALL_SUCCESS\$**—The \$INSTALL_SUCCESS\$ variable cannot be set externally because it is a read-only variable, and therefore it cannot be advertised. But InstallAnywhere supports the automatic propagation of this

variable from the Pre-Install phase to the Post-Install phase (but not from the Install phase to the Post-Install phase) for a Dynamic Merge Module.

- **\$RESTART_NEEDED\$**—The \$RESTART_NEEDED\$ variable has to be advertised in the Post-Install phase in order for it to be propagated from the merge module to the parent.

Building Installers Using Build Configurations

Starting with InstallAnywhere 2011, you have multiple ways to build an installer. The following tables lists the methods you can use to build an installer.

Table 5-14 • Methods to Build an Installer

Method	Description
Build Selected Build Configurations	You can use the Build Selected button to build a selected Build Configuration. See Building a Selected Build Configuration .
Build All Project-Enabled Build Configurations	You can build all the Build Configurations that have been enabled for project build by using the Build Project button. See Building Build Configurations Enabled for Project Build .
Build All Build Configurations	You can build all of the Build Configurations available in a project by using the Build All button. See Building All Build Configurations .
Via Command Line	You can use the command line to specify which Build Configurations to build. See Running the Command-Line Build .

Building a Selected Build Configuration

A Build Configuration that is currently selected in the Build view of the InstallAnywhere Advanced Designer can be built using the Build Selected button, by performing the following steps.



Task:

To build the selected Build Configuration:

1. In the Advanced Designer, click the **Build** task. The **Build Configurations** tab opens, listing all of the project's defined Build Configurations.
2. From the **Select Build Configuration** list, select the Build Configuration that you want to build.
3. Click **Build Selected**. The selected Build Configuration is built.

Building Build Configurations Enabled for Project Build

Build Configurations can be enabled for Project build by selecting the **Add to project build** option associated with that particular Build Configuration.

To build only the Build Configurations in the project that have been enabled for Project Build, perform the following steps:

**Task: To build all Build Configurations in the project build:**

1. In the Advanced Designer, click the **Build** task. The **Build Configurations** tab opens.
2. Click **Build Project**. All of the Build Configurations that have been enabled for Project Build will be built.

Building All Build Configurations

To build all Build Configurations in the project, perform the following steps.

**Task: To build all Build Configurations:**

1. In the Advanced Designer, click the **Build** task. The **Build Configurations** tab opens.
2. Click **Build All**. All of the Build Configurations defined in the project are built.

Running the Command-Line Build

You can use the command-line build (build.exe) to build installers for the Build Configurations in an InstallAnywhere project. To build using the command line, use the following format:

```
build.exe MyProject BuildConf1 BuildConf2 BuildConf3
```

where:

Table 5-15 • Syntax of Command Line Build

Item	Description
MyProject	Name of InstallAnywhere project you are building.
BuildConf1, BuildConf2, BuildConf3	List of all of the Build Configurations defined in that project that you want to build, separated by commas.

You can also use the -all option to build all of the project's Build Configurations (even those that do not have their **Add to project build** option enabled), such as:

```
build.exe MyProject -all
```



Note • The command-line build can be called with many options to override build targets, distribution options, working directory, and more. See [Build Command-Line Arguments](#) for details.

Testing Installers

After the Build process is complete, you can test the installer by selecting either the **Try Web Install** or **Try Installer** button on the **Build Configurations** tab of the Advanced Designer.

- **Try Installer**—Click to run the default installer for your operating system.



Note • If two or more installers are built for your operating system, InstallAnywhere shows a control that lists the installers you can try. See [Multiple Target Installers](#).

- **Try Web Install**—Click to test the installation by launching a Web browser and the InstallAnywhere Web Install Page generated by the build process.

Multiple Target Installers

If, after the project has been built, there are more than one supported target installers available for the authoring platform, a selection list appears between the **Build All** and **Try Installer** buttons. If there are none or only one supported target installer, this list will be hidden.

The name that is displayed for each Target Installer uses one of the following formats:

- **Without VM**—OutputFolderName_BuildConfigurationName
- **With VM**—OutputFolderName_BuildConfigurationName - BundledVMName

Testing Installers for Other Platforms

When building for a platform other than that on which the installer is being developed, transfer that installer, and run it manually. By default, installers will be located in the `Build_Output` directory found in the same folder as the `.iap_xml` project file.

The `Build_Output` directory contains a subfolder for each Build Configuration in the project. Within each Build Configuration directory in the `Build_Output` folder, will be `Web_Installers`, `CDROM_installers`, or both. From within each of these directories, choose the platform to test. For the CDROM installer, transfer the entire contents of the `CDROM_Installers` folder.

Working with VM Packs

VM packs can be bundled with your installer to provide a Java virtual machine on the target system. This section covers some of the basic VM-related tasks.

Table 5-16 • VM Pack Tasks

Topic	Description
Downloading VM Packs	Explains how to download VM packs from the InstallAnywhere Web site.
Installing VM Packs	Discusses how to store VM packs in a location where InstallAnywhere can access them.
Bundling a VM Pack	Explains how to bundle a VM pack with the installer for a specific build target.



Note • For more advanced topics related to VM packs and Java virtual machines, see [Adding Advanced Procedures to Installers](#).

Downloading VM Packs

Build targets can specify the Java virtual machine to bundle with each target's installer. You can download additional VM packs for InstallAnywhere from the InstallAnywhere Web site. However, the VM packs you download are not available for selection on the **Build** task until you install them.

**Task:****To download a VM pack:**

1. In the Advanced Designer, select **Download Additional VM Packs...** on the **Help** menu. InstallAnywhere opens the VM pack download page in your system's default Web browser:

http://www.flexerasoftware.com/downloads/downloads_7150.htm

2. Locate the VM pack you want to download and save it to disk.



Note • These VM packs must be stored in <InstallAnywhere>/resource/installer_vms or another valid VM pack location. See [Installing VM Packs](#).

Installing VM Packs

The VM packs you download or create are not available for selection on the Build task until you install them in the `installer_vms` directory (commonly, `<InstallAnywhere>/resource/installer_vms`) or another valid VM path you define in the Preferences dialog box.



Task: To install a downloaded VM pack

1. Copy the VM pack to a valid VM pack location. The default location for VM packs is
`<InstallAnywhere>\resource\installer_vms\`
2. Exit and restart InstallAnywhere.



Note • The VM packs listed in the VM list on the Build Targets tab (**VM to Bundle with Installer**) are refreshed each time you edit preferences or start InstallAnywhere. Therefore, you cannot simply copy a VM pack into a valid VM pack path. You must first open and close the Preferences dialog box or restart InstallAnywhere.

For more information about valid VM pack locations, see [Resources](#) in [InstallAnywhere Preferences Dialog Box](#).

Bundling a VM Pack

The VM packs you download or create must be stored in a valid VM path before they become available to your InstallAnywhere project.



Note • For details on installing VM packs for use with your InstallAnywhere projects, see [Installing VM Packs](#).



Task: To bundle a VM pack:

1. In the Advanced Designer, click the **Build** task.
2. On the Build task, locate the build target for which you want to bundle a VM. Some build targets, such as Mac OS X and Other Java-Enabled Platforms (Pure Java), do not support VM bundling.
3. For the build target you want to include a bundled VM, click the **With VM** checkbox.
4. In the **VM to Bundle with Installer** list, click the VM you want to bundle with the build target.
5. Click **File > Save** to store your changes in the project file.

Creating Launchers for Java Applications



Note • For information on Advanced Settings options for the Create LaunchAnywhere for Java Application action, see [Customizing Individual Launcher Settings](#).

InstallAnywhere provides the **Create LaunchAnywhere for Java Application** action in the **Install** task. Click the **Add Launcher** button. This action creates a platform-native executable for launching a Java application (also known as a LAX executable). This LAX allows Java applications to be launched as if they were platform-native. For example, the LaunchAnywhere executable can be launched by double-clicking or by being called directly from the command line.

On Windows, InstallAnywhere appends .exe to the file. On Mac OS X, it creates a .app package.

A LaunchAnywhere executable is also responsible for invoking the Java runtime, setting the environment for the Java application, and passing any command-line arguments that may be required by the application.

Launcher Path and Name

The Path and Name text boxes describe the destination location and name of the launcher.

Table 5-17 • Launcher Path and Name Settings

Control	Description
Path	Identifies the location of the LaunchAnywhere executable to be created on the target system. You can use either the magic folder list or the Visual Tree's arrow buttons to change the path.
Name	The name of the LaunchAnywhere executable installed onto the target system. Modify the text to modify the Launcher name.

Main Class and Arguments

When adding a LaunchAnywhere action you will be given the option to search for the main class of your Java application. You can let InstallAnywhere find it for you or you can enter it yourself.

You can also define the command-line arguments that should be passed to your Java application when the LaunchAnywhere executable is run.

Launcher Type

Defining the Launcher Type allows developers to specify whether the Java application is either a GUI-based application calling javaw.exe (stdout and stderr suppressed by default), or a console application calling java.exe (stdout and stderr directed to the console by default). In addition, on Windows systems, console launchers will redirect stdout and stderr output to the same console window from which they were invoked.

Launcher Properties

Launcher properties are the LAX Properties associated with the launcher.



Task: *To edit LaunchAnywhere properties*

1. On the Create LaunchAnywhere for Java Application customizer, click **Edit Properties**. This opens the LaunchAnywhere Properties dialog box.
2. In the LaunchAnywhere Properties dialog box, **Add**, **Edit**, or **Remove** LAX properties.
3. Click **OK** to close the LaunchAnywhere Properties dialog box.

For more information on LaunchAnywhere properties, see [LAX Properties](#).

Launcher Icon

The Launcher Icon enables developers to select the icon to be used for the LAX executable on the Mac OS X and Windows platforms. Custom icons are not available on Unix installers. Mac OS X icons may be 128 x 128.



Note • *On Windows, the LaunchAnywhere executable itself will not display the specified icon directly. Instead, the LaunchAnywhere executable will have the generic .exe icon and an .ico file will appear in the same folder as the executable, but all shortcuts created during installation that target the LaunchAnywhere will inherit the custom icon. InstallAnywhere does not support interlaced GIF files.*

Windows Execution Level

LaunchAnywhere executables support Windows UAC (User Account Control).

Table 5-18 • Windows Execution Level Options

Option	Description
As Invoker	The launcher acquires the same execution level as its parent process.
Highest Available	The launcher requests the highest execution level (Windows privileges and user rights) available to the current user.
Administrator	The launcher requires local admin privileges to run. Depending on the privileges of the current user account and the configuration of the target system, this setting may result in a launcher that will not start.

Override Default UNIX/Mac OS X Permissions

This control allows you to provide pre-determined permissions for the launcher.

**Task:** *To override permissions*

1. Click **Override default UNIX/Mac OS X permissions**. This enables the Permissions text box.
2. Type the numeric permissions value (for example: the octal number mode, **755**) in the Permissions text box.

Improving Installation Performance

Follow the points below to optimize your installers.

Minimizing Size of Installers

Here are several suggestions to minimize the size of an installer.

- In **Build > Distribution**, select the **Optimize Installer Size by Platform** option.
- Avoid the background image used in GUI installers.
- Include a billboard that has a minimal file size.
- Exclude install panel labels or panel images.
- Use only the essential install steps.
- Do not include an uninstaller if one is not necessary.
- Use SpeedFolders to install groups of files together. SpeedFolders allow an entire folder of files to be treated as a single installation action, reducing space and increasing performance.
- Only build for the needed locales.
- Do not bundle a JRE. If you must include a JRE, use one that does not include international resources.

Optimizing Unix Installers

InstallAnywhere creates a single Unix installer without a bundled VM that can be run on most Unix platforms. The installers with bundled VMs are built specifically for each platform and may be optimized for the platform. The Unix installer without a bundled VM, however, contains all the resources that are needed for every Unix platform. Having all of the resources for the various Unix platforms can make this Unix without VM installer larger than the optimized installers for individual Unix platforms which actually include a VM. However, the Unix without VM installer will be smaller than the Unix installer built without platform optimizations. For similar reasons, the pure Java installer never receives any platform optimizations and is always the same size regardless of platform optimizations.

Launching Additional Installers

The **Execute Target File** action allows launching an installer from within an installer. Use this action when you need to launch another installer that is not an InstallAnywhere installer. You can also use Merge Modules to launch other InstallAnywhere installers to receive the maximum optimization and performance.

Installing Fonts

InstallAnywhere handles font installation with the help of the \$FONTS\$ magic folder.



Task: *To install fonts on a target system*

1. On the **Install** task, click **Add Files**. This opens the Add Files to Project dialog box.
2. Navigate to the font file you want to install, select it, and click **Add**. The font file you selected appears in the Files to Add list.
3. Click **Done**. This closes the Add Files to Project dialog box and activates the Install File customizer for the new font file.
4. In the Install File customizer, change the destination **Path** from **User Install Folder** (the default) to **Fonts Directory**. Notice that InstallAnywhere adds the \$FONTS\$ magic folder to the visual tree.

During installation, InstallAnywhere installers resolve the location for the \$FONTS\$ magic folder correctly for the target system.

Team Development

Source paths allow developers to reference file resources using variable paths instead of absolute paths. This allows you to share a project file with other team members, even when the file resources are located at different paths on their development systems. With source paths, you can even use the same project file on different types of operating systems, such as UNIX and Windows.

Table 5-19 • Team Development Tasks

Topic	Description
Enabling Source Paths	Shows how to enable source paths on the Source Paths tab of the InstallAnywhere Preferences dialog box.
Adding and Removing Source Paths	Explains how to add or remove source paths both in the Advanced Designer and through the use of system environment variables.
Updating the Location of Files and Resources	Discusses how to deal with issues that arise when source paths change.

Table 5-19 • Team Development Tasks

Topic	Description
Working with Source Control Management Software	Discusses the use of InstallAnywhere with source control management software.

Enabling Source Paths

Before you can add or remove source paths on the **Source Paths** tab of the **InstallAnywhere Preferences** dialog box, you must first enable source paths.



Task: *To enable source paths*

1. Click **Edit > Preferences** and then click the **Source Paths** tab.
2. Select **Enable Source Paths**.

Adding and Removing Source Paths

Source paths can be added through the use of source path variables in two different ways.

Adding Source Paths Using Preferences Dialog Box

To add a source path using the **Preferences** dialog box, perform the following steps:



Task: *To add source paths using the Preferences dialog box*

1. On the **Source Paths** tab of the **InstallAnywhere Preferences** dialog box, click **Add**. A blank box appears under **Access Path Name**.
2. Click the blank box under **Access Path Name** and enter a variable name for the source path. For example, **RESOURCE**.
3. Click the blank box under **Folder**. The Choose Folder dialog box appears.
4. Browse to the source path and click **Select**. InstallAnywhere adds the full path to the **Folder** box.
5. Click **OK**. To refer to this source path elsewhere in your InstallAnywhere project, use the following notation: **\$<access_path_name>\$**. For example, **\$RESOURCE\$**.

Adding Source Paths Using System Environment Variables

To add a source path using system environment variables, perform the following steps:



Task: *To add source paths using System Environment Variables*

1. Access the environment variables on the development system:
 - On Windows, right-click on My Computer on the Desktop, choose properties, choose Advanced, and click Environment Variables.
 - On Unix or Mac OS X, modify the proper shell configuration file or set the variable directly using the shell.
2. Add an environment variable for the source path, prepended with IA_PATH_ tag. For example, to set the source path SOURCE_PATH, set the environment variable IA_PATH_SOURCE_PATH.

Removing Source Paths

To remove a customized source path, perform the following steps:



Task: *To remove a customized source path*

1. Click the source path name under **Access Path Name**.
2. Click **Remove**.
3. Click **OK**.

Updating the Location of Files and Resources

When the location of a file or folder has changed, simply change the folder location listed for the source path. By changing the source path to the new location, you cause InstallAnywhere to update the references to the resources automatically.

If you open a project and InstallAnywhere cannot find the resources, either because they have moved or they no longer exist, you will be asked to locate the resource or remove the resource from the project. If you want to open a project without updating the location of the resource, change the Project Loading preference to Never. (The Project Loading preference is available on the General Settings tab of the InstallAnywhere Preferences dialog box.) Projects will be opened without checking the location of each resource. Instead, resources will be checked only when you build.

Working with Source Control Management Software

InstallAnywhere supports the Advanced Designer and project files to be checked into Source Control Management (SCM) software like ClearCase, Perforce, or CVS. You can also launch InstallAnywhere builds without checking it out of source control. InstallAnywhere will write out temporary files in the system's temp folder, the user's home\InstallAnywhere folder, and the folder where the InstallAnywhere project file is located. You can optionally place VM packs and plugins in the user's home\InstallAnywhere folder on Windows or Unix. This is helpful if you change or update your VM packs and plugins often, but you do not want to check them into source control.

Referencing Developer Installation Manifests (DIMs)

Using InstallAnywhere Collaboration or InstallShield Collaboration, application developers create Developer Installation Manifests (DIMs). DIMs are XML files that describe specific product subsystems and their requirements. DIMs are named with the .dim extension and describe the installation requirements for one subsystem of the entire software product.

InstallAnywhere supports referencing DIMs in installation projects. The following sections explain how to work with DIM references:

- [Adding a DIM Reference](#)
- [Customizing a DIM Reference](#)
- [Removing a DIM Reference](#)
- [Creating a Shortcut to a File within a DIM Reference](#)

Adding a DIM Reference

There are two ways to add a DIM reference to a project. You can add a DIM reference using the **DIM References** subtask under the **Organization** task, which provides a global view of all DIM References added to a project. You can also add a DIM reference using the **Install** task, which provides the ability to order DIM references amongst other actions.



Note • If you add a DIM reference using the **DIM References** subtask, it also appears in the **Visual Tree** under the **Install** task. If you add a DIM reference using the **Install** task, it also appears in the **DIM References List** in the **DIM References** subtask. Any additions or changes made in either of these tasks are reflected both places in the Advanced Designer.

Adding a DIM Reference Using the DIM References Subtask

To add a DIM reference using the DIM References subtask, perform the following steps:



Task: *To add a DIM reference using the DIM References subtask*

1. On the Advanced Designer, click **Organization** then **DIM References**. The DIM Reference List appears.
2. Click **Add DIM Reference**. The **Add DIM Reference to Project** dialog box appears.
3. Browse to the .dim file and click **Open**. The DIM Reference appears in the DIM Reference List and the DIM Reference customizer appears.



Note • *DIM references can be reassigned to Features and Components like all other actions. By default, DIM references are associated with the first available component and the DIM reference is associated with the features for that component. Use the **Assign to** list to reassign DIM references.*

DIM references are always added at the same level as Magic Folders. You cannot make a DIM reference a child of another Magic Folder, but you can use the navigational arrows to change their order amongst the other Magic Folders within the Visual Tree.

4. Continue with [Customizing a DIM Reference](#) to specify or change properties associated with the DIM reference.

Adding a DIM Reference Using the Install Task

To add a DIM reference using the Install task, perform the following steps:



Task: *To add a DIM reference using the Install task*

1. On the Advanced Designer, click **Install**.
2. Click **Add Action**. The **Choose an Action** dialog box appears.
3. Click **Create DIM Reference** and click **Add**. A DIM reference appears in the Visual Tree and the DIM Reference customizer appears.



Note • *DIM references can be reassigned to Features and Components like all other actions. By default, DIM references are associated with the first available feature and are added at the same level as Magic Folders. Use the **Assign to** list and the navigational arrows to reassign DIM references and change their order within the Visual Tree.*

4. Continue with [Customizing a DIM Reference](#) to associate the DIM reference with a .dim file and specify or change properties associated with the DIM reference.

Customizing a DIM Reference

When you click a DIM reference on the **DIM References List** or on the **Visual Tree**, a customizer appears that allows you to view or configure specific properties that are defined with the DIM.



Note • Any changes you make in the DIM Reference customizer are reflected in both the **DIM References** subtask and the **Install** task.

The DIM Reference customizer contains the following tabs:

- General tab
- Build Variables tab
- Runtime Variables tab



Task: *To customize a DIM reference*

1. On either the **DIM References List** or the **Visual Tree**, click the DIM reference you want to customize. The **DIM Reference** customizer appears.
2. Click the **General** tab. The General tab properties appear.
3. Click **Choose DIM Reference** to associate the DIM reference with a .dim file. The **Add DIM Reference to Project** dialog box appears.
4. Browse to the .dim file and click **Open**. All of the General tab properties are updated to reflect the DIM that you selected.



Note • All of the General tab properties are read-only except for **Destination Path**. This property represents the location that the predefined variable \$[INSTALLDIR] is associated with according to the DIM.

5. To change location associated with \$[INSTALLDIR], click the **Destination Path** box and type the desired location.
6. Click the **Build Variables** tab. The Build Variables tab properties appear.
7. To change the value associated with the variable, click the **Value** box and either type the desired value or click the ellipses button to browse to a directory location.



Note • The **Name** and **Description** properties are read-only.

8. Click the **Runtime Variables** tab. The Runtime Variables tab properties appear.
9. To change the value associated with the variable, click the **Value** box and type the desired value.



Note • The **Name** and **Description** properties are read-only.

Removing a DIM Reference

There are two ways to remove a DIM reference from a project. You can remove a DIM reference using the **DIM References** subtask under the **Organization** task, which provides a global view of all DIM References added to a project. You can also remove a DIM reference using the **Install** task, which provides the ability to order DIM references amongst other actions.



Note • If you remove a DIM reference using the **DIM References** subtask, it no longer appears in the **Visual Tree** under the **Install** task. If you remove a DIM reference using the **Install** task, it no longer appears in the **DIM References List** in the **DIM References** subtask. Any additions or changes made in either of these tasks are reflected both places in the Advanced Designer.

Removing a DIM Reference Using the DIM References Subtask

To remove a DIM reference using the DIM References subtask, perform the following steps.



Task: **To remove a DIM reference using the DIM References subtask**

1. On the Advanced Designer, click **Organization** then **DIM References**. The **DIM Reference List** appears.
2. On the **DIM Reference List**, click the DIM reference you want to remove.
3. Click **Remove**. The DIM reference no longer appears in the list.

Removing a DIM Reference Using the Install Task

To remove a DIM reference using the Install task, perform the following steps.



Task: **To remove a DIM reference using the Install task**

1. On the Advanced Designer, click **Install**.
2. On the **Visual Tree**, click the DIM reference you want to remove.
3. Click **Remove**. The DIM reference no longer appears in the Visual Tree.

Creating a Shortcut to a File within a DIM Reference

Once a DIM reference is added to your project, you can create a shortcut to a file defined within a DIM. The shortcut appears in the **Visual Tree** under the **Install** task like other actions.



Task: *To create a shortcut to a file within a DIM reference*

1. On the Advanced Designer, click **Install**.
2. Click **Add Action**. The **Choose an Action** dialog box appears.
3. Click **Create Alias, Link, Shortcut to DIM** and click **Add**. A generic shortcut appears in the Visual Tree and the **Create Alias, Link, Shortcut** customizer appears.
4. Click **Close** to close the **Choose an Action** dialog box.
5. Click **Choose Target**. The **Choose an Alias, Link, Shortcut Target** dialog box appears.
6. Double-click the magic folder under the DIM that contains the file you want to target. The magic folder expands displaying a list of files.
7. Click the file you want to target and click **OK**. The shortcut description in the Visual Tree reflects the name of the file you selected.



Note • *The target file must be present on the system before the Create Alias, Link, Shortcut action is executed. Use the navigational arrows as necessary to ensure the shortcut is ordered after the DIM reference that contains the target file.*

Using FlexNet Connect

InstallAnywhere uses FlexNet Connect to detect update notifications available for InstallAnywhere. The following sections explain how to use InstallAnywhere's FlexNet Connect functionality to check for notifications and install updates to InstallAnywhere:

- [Checking for InstallAnywhere Update Notifications](#)
- [Installing InstallAnywhere Updates](#)

You can also integrate FlexNet Connect with your application so update notifications are automatically available to your end users. For information and instructions on integrating FlexNet Connect functionality in your products and installers, see [Deploying Your Product with FlexNet Connect](#).

Checking for InstallAnywhere Update Notifications

You can configure InstallAnywhere to automatically detect when update notifications are available or you can check for update notifications manually.



Task: **To configure InstallAnywhere to automatically detect when update notifications are available**

1. On the **Edit** menu, click **Preferences**. The InstallAnywhere Preferences dialog box opens.
2. Click the **Updates** tab.
3. Select the **Check for updates automatically** check box.
4. To specify a specific HTTP proxy setting, select the **Use the following HTTP proxy setting** check box and enter the proxy address and port in the boxes provided.
5. Click **OK**.



Task: **To manually check for update notifications**

1. On the **Help** menu, click **Check for Updates**.
 - If a FileSync update is available, the **Update Available** dialog box appears. A FileSync update represents a critical update to the product.
 - If a non-critical update or message is available, a **FlexNet Connect** window appears.
2. Continue with the appropriate procedure in the [Installing InstallAnywhere Updates](#) section depending on what kind of update you are installing.

Installing InstallAnywhere Updates

This section explains how to install a critical InstallAnywhere update, and also how to install a non-critical update or message.



Note • Select the **Check for updates automatically** check box if you want InstallAnywhere to check for update notifications automatically.

Installing a Critical Update

To install a critical update, perform the following steps:

**Task:** *To install a critical update*

1. On the **Update Available** dialog box, choose when you want to install the update:
 - Click **Now** to immediately install the available update. If you have unsaved changes in your project, a dialog box appears prompting you to save the project. Once you save your changes, InstallAnywhere closes, and a dialog appears displaying the progress of the update installation.
 - Click **On Exit** to install the available update the next time you close InstallAnywhere. Once you close InstallAnywhere, a dialog appears displaying the progress of the update installation.
 - Click **Later** to dismiss the dialog box. The update is not installed, but this update is still available next time you check for updates. If you choose to install this update next time you check for updates, a dialog appears displaying the progress of the update installation.
2. On the progress dialog, click the arrow to view more details about the status of the update installation.
3. Click **Finish** when the update is downloaded and installed according to the progress dialog.



Note • The update installation process includes three phases: download, installation, and registration. You cannot cancel the update installation during the registration phase. If you click **Cancel** during the download or installation phase, the download completes but the installation and registration phase of the installation process is terminated.

Installing a Non-Critical Update or Message

To install a non-critical update or message, perform the following steps:

**Task:** *To install a non-critical update or message*

1. On the **FlexNet Connect** window, select the available update in the list.
2. Click **Install**. The update installation starts.

Adding Advanced Procedures to Installers

This section explains how to add more advanced procedures to an InstallAnywhere installer:

Table 6-1 • Adding Advanced Procedures to Installers

Topics	Description
Working with Variables	Includes topics related to setting InstallAnywhere variables, checking variable values, and encrypting variable values in your installers.
Creating VM Packs	Describes how to create your own VM packs for InstallAnywhere. (InstallAnywhere freely provides many pre-built VM packs on the Web.)
Controlling the VM Used to Run the Installer	Explains how to customize the criteria used for the VM search performed by the installer's launcher. (This determines the VM under which the installer runs.)
Controlling the Install of Bundled VMs	Explains how to control when a VM bundled with your installer is installed and where to install it.
Controlling the VM Your Launchers Use	Explains how to use project-level settings for the installer's VM search as well as launcher-specific settings that can leverage, or operate independently from, the installer's VM search. (Launcher-specific settings use the Advanced Settings on the Create LaunchAnywhere for Java Application customizer.)
Using the Choose Java VM Panel	Describes how to use the Choose Java VM customizer to specify the functions your end users may use to select a VM for the software your installer deploys.
Generating Response Files	Explains how to generate response files for use with silent installers.
Generating Log Files	Explains how to generate installation logs.

Table 6-1 • Adding Advanced Procedures to Installers

Topics	Description
Getting User Input	Shows examples of the Get User Input - Simple and Get User Input - Advanced panels.
Using Command-Line Arguments with Installers and Uninstallers	Lists the command-line arguments you can use with InstallAnywhere installers.
Setting Project Version at Build Time	Provides several methods you can use to change the version of an InstallAnywhere project at build time.
Checking Disk Space During Installation	Explains how to create an installer that performs a disk space check at various points in the installation life cycle.
Deploying Your Product with FlexNet Connect	Describes how to add, customize, or remove FlexNet Connect with the Enable Update Notifications action.
Localizing Projects and Installers	Includes several topics about how to localize projects, installers, and installer resources.
Packaging and Executing Custom Code	Discusses how to use custom code with your InstallAnywhere installers.
Calling InstallShield MultiPlatform APIs in InstallAnywhere	Explains how to import InstallShield MultiPlatform APIs (Services) into a custom code action.
Packaging Custom Code as a Plug-in	Explains how to package custom code for use as an InstallAnywhere plug-in action.
Internationalizing Custom Code	Describes how to localize custom code.
Digitally Signing Installers	Describes how to apply a digital signature to your Windows installer.
Setting Installation Rollback Options	Explains how to select the Enable Rollback option to avoid application corruption if an end user cancels an installation before it is completed or if an installation encounters a fatal error.
Determining a Successful Installation	Describes how to use the \$INSTALL_SUCCESS\$ variable to determine the result of an install process.
Troubleshooting	Contains several topics about debugging installers and provides details about supported application servers and database servers.

Working with Variables

In the Advanced Designer, you can set variable values, check the values of existing variables, and define which variables must be encrypted.

Table 6-2 • Working with Variables

Topic	Description
Setting Variables in the Advanced Designer	Describes the process of setting variable values with the Set InstallAnywhere Variable actions.
Checking the Value of a Variable	Discussed methods of discovering the value of a variable.
Encrypting Variable Values	Explains how to encrypt variable values in your installers.

Setting Variables in the Advanced Designer

The Advanced Designer provides two actions specifically for setting InstallAnywhere variables: **Set InstallAnywhere Variable - Single Variable** and **Set InstallAnywhere Variable - Multiple Variables**. Both actions require you to provide a variable name and value for each variable. In addition, you can choose to set the variable to be evaluated at assignment or resolved each time the variable is referenced.

- [Adding a Set InstallAnywhere Variable - Single Variable Action](#)
- [Adding a Set InstallAnywhere Variable - Multiple Variables Action](#)

Adding a Set InstallAnywhere Variable - Single Variable Action

To add a **Set InstallAnywhere Variable - Single Variable** action, perform the following steps:



Task: *To add a Set InstallAnywhere Variable - Single Variable action:*

1. In the Pre-Install, Post-Install, Pre-Uninstall, or Post-Uninstall task, click **Add Action**. This opens the **Choose an Action** dialog box.
2. In the Choose an Action dialog box, double-click **Set InstallAnywhere Variable - Single Variable** action. InstallAnywhere inserts the action in the active task.
3. In the **Variable Name** box on the **Set InstallAnywhere Variable - Single Variable** customizer, enter a name to identify this variable.
4. In the **Set Value To** box, enter the value you want to set for this variable.
5. To set this variable to evaluate at assignment, select the **Evaluate any variables at assignment** option.

6. To avoid the substitution of unknown variables for this action by instructing InstallAnywhere to only resolve InstallAnywhere variables which are listed in the project under **Project > Variables** (the known variables), select the **Do not substitute unknown variables** option. For more information, see [Preventing the Substitution of Unknown Variables](#).

Adding a Set InstallAnywhere Variable - Multiple Variables Action

To add a **Set InstallAnywhere Variable - Multiple Variables** action, perform the following steps:

**Task:****To add a Set InstallAnywhere Variable - Multiple Variables action:**

1. In the Pre-Install, Post-Install, Pre-Uninstall, or Post-Uninstall task, click **Add Action**. This opens the **Choose an Action** dialog box.
2. In the Choose an Action dialog box, double-click **Set InstallAnywhere Variable - Multiple Variables** action. InstallAnywhere inserts the action in the active task.
3. On the **Set InstallAnywhere Variable - Multiple Variables** customizer, enter text in the **Comment** box to identify the purpose of this set of variables.
4. Click **Edit Variables**. This opens the **Edit Multiple Variables** dialog box.
5. In the **Edit Multiple Variables** dialog box, click **Add**.
6. Enter a variable name in the **Variable Name** column.
7. Enter the value you want to set for this variable in the **Value** column.



Note • To set this variable to evaluate at assignment, select the check box in the **Evaluate at Assignment** column.

8. Click **Add** again and repeat the above steps to add additional variables.
9. When you have finished adding variables, click **OK** to close the **Edit Multiple Variables** dialog box. The variables that you added are now listed in the Variables list.
10. To avoid the substitution of unknown variables for this action by instructing InstallAnywhere to only resolve InstallAnywhere variables which are listed in the project under **Project > Variables** (the known variables), select the **Do not substitute unknown variables** option. For more information, see [Preventing the Substitution of Unknown Variables](#).

Preventing the Substitution of Unknown Variables

When setting variables in your installation project using any of the following actions, a **Do not substitute unknown variables** option is available:

- Set InstallAnywhere Variable - Multiple Variables
- Set InstallAnywhere Variable - Single Variable
- Execute Command

- Execute Script/Batch File
- Execute Target File

If your installation project includes a nested variable which could possibly contain multiple \$ characters, this could result in the creation of an unknown variable. Therefore, select the **Do not substitute unknown variables** option to instruct InstallAnywhere to only resolve InstallAnywhere variables which are listed in the project under **Project > Variables** (the known variables).

The following table provides a scenario where it would be beneficial to select this option.

Table 6-3 • Variable Scenario

Variable	Defined As	Set To
\$USER\$	Defined by end user.	te\$t
\$PWD\$	Defined by end user.	pa\$sword
\$COMMAND_LINE\$	-u \$USER\$ -p \$PWD\$	If the Do not substitute unknown variables option is selected, the \$COMMAND_LINE\$ variable will be set correctly to: -u te\$t -p pa\$sword However, if the Do not substitute unknown variables option is not selected, the text between the two \$ characters is read as a variable—which does not exist—and is therefore replaced with an empty value, resulting in an invalid value: -u tesword

Checking the Value of a Variable

The values of many variables may change throughout the progress of the installer. For predefined variables, the points in the installer where their values change are documented in [Standard InstallAnywhere Variables](#). The best way to check the values of variables during the install is the **Output Text to Console** and **Output Debug Information** actions. Both of these can be used to print out useful debugging information to either the console or text files. Developers can also use the **Display Message** panel to show the variable and its value.

Encrypting Variable Values



Caution • Encryption is not fully supported for Merge Modules. Sensitive information that requires encryption should not be included in a Merge Module.

Encrypting your variable values can make your installers more secure. The values of encrypted variables are stored only in their encrypted form. InstallAnywhere decrypts encrypted variables when the installer needs to retrieve or set their values, but records only the encrypted values in installer output, such as install scripts, install logs, response files, and installer debug output.



Tip • Use the **Security** settings on the **Project > Variables** subtask to specify the type of encryption you want your installers to use. Using JCE encryption, along with a FIPS-compliant JCE library and supported algorithm, you can ensure your installers are FIPS 140-2 compliant. See **Security** settings on the **Variables** subtask for more information about the encryption options InstallAnywhere provides.

**Task:** **To encrypt a variable value**

1. On the **Project > Info** subtask, click **Configure**. This opens the Configure Variables dialog box.
2. On the Configure Variables dialog box, click **Add**.
3. In the Variable Name column, enter the variable name for the variable value you want to encrypt.
4. In the corresponding Options column, choose **Encrypt Variable Value**.
5. Click **OK**.



Note • To remove encryption from a variable in your project, select that variable in the Configure Variables dialog box and click **Remove**.

Creating VM Packs

Many VM packs are available for download on the InstallAnywhere Web site.

http://www.flexerasoftware.com/downloads/downloads_7150.htm

If none of the VM packs available on the Web site meet your requirements, you can create your own VM pack. Information about creating VM packs is presented in the following topics:

- [Using the Create VM Pack Utility](#)
- [Creating a VM Pack Manually](#)
- [VM Pack Structure](#)
- [VM Pack Properties](#)
- [Making a VM Pack FIPS-Compliant](#)

Using the Create VM Pack Utility

You can use the **Create VM Pack** dialog box, which is opened when you select **Create VM Pack** on the **File** menu of the Advanced Designer, to create a VM pack for any platform.



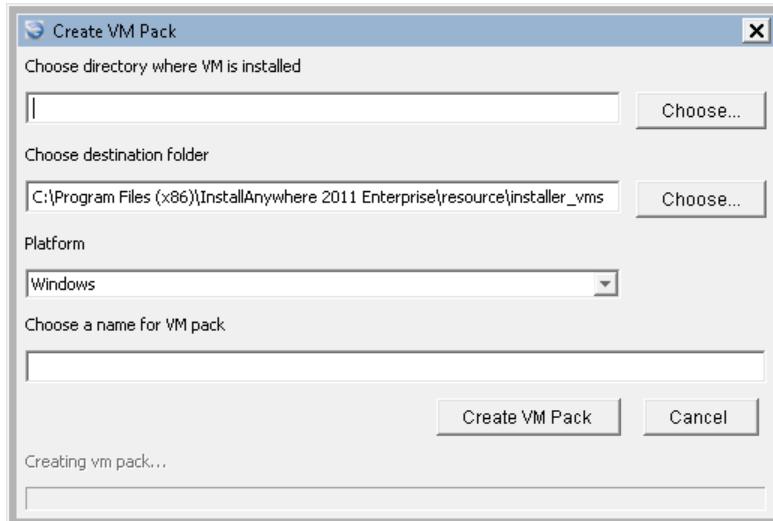
Note • You can also use the VM Pack Utility as a standalone tool on a machine where InstallAnywhere is not installed. See [Running the VM Pack Utility as a Standalone Tool](#).

To create a VM pack using the Create VM Pack utility, perform the following steps:



Task: To create a VM pack using the Create VM Pack utility:

1. On the **File** menu of the Advanced Designer, select **Create VM Pack**. The **Create VM Pack** dialog box opens.



2. Click **Choose** next to **Choose directory where VM is installed** and select the directory containing the VM. This directory must be named **jre**.



Important • The VM must be located in a directory named **jre**. See [Preparing a VM Pack for Use: “jre” Directory Name Requirement](#) for more information.

3. Click **Choose** next to **Choose destination folder** and select the directory where you want to store the new VM pack.
4. From the **Platform** list, choose the platform of this new VM pack. Available options are:
 - Windows
 - Solaris

- HP-UX
- AIX
- Linux



Note • If the **Platform** that you select does not match the platform of the selected JRE/JDK, the files used to create the VM pack are not cleaned up correctly, and the destination folder will contain extra files which are not part of the VM pack.

5. In the **Choose a name for VM pack** field, enter a name to uniquely identify the new VM pack.
6. Click **Create VM Pack** to create the VM pack using the specified information.

Running the VM Pack Utility as a Standalone Tool

You can also use the VM Pack utility as a standalone tool on a machine where InstallAnywhere is not installed.

**Task:****To run the Create VM Pack utility as a standalone tool:**

1. Copy the VMPackUtility directory, which is found in the InstallAnywhere installation directory, to the machine on which the JDK/JRE installation is present:
2. Execute the utility using the following command line:

```
$ > java -jar Utility.jar
```

A wizard which automates the process of creating a VM pack opens.

Creating a VM Pack Manually

As an alternative to using the **Create VM Pack** utility (as described in [Using the Create VM Pack Utility](#)), you can also manually create a VM pack.

**Task:****To create a VM pack:**

1. Create an archive of the JRE directory.
 - a. For Windows, create a `vm.zip` archive. Do not use compression and do not retain full path info.
 - b. For Unix, create a `vm.tar.Z` archive. Create a compressed tarball. For example,
`tar -czvf vm.tar.Z jre`.
2. Create a `vm.properties` file. See [VM Pack Properties](#) for details on properties.
3. Create a ZIP archive of the VM archive (`vm.zip` or `vm.tar.Z`) and the `vm.properties` file. For best results, use maximum compression on this archive.
4. Rename the resulting ZIP archive: `<CustomVMPack>.vm`.

VM Pack Structure

VM packs are stored as archives with a .vm extension. VM packs contain a VM archive (vm.zip or vm.tar.Z) and a vm.properties file.

The structure and contents of the JVM archive inside the VM pack vary between Windows and Unix systems.

Structure of Windows VM Packs

```
Windows.vm
|-vm.properties
|-vm.zip
  |-bin
    |-java
  |-lib
```

Structure of Unix VM Packs

```
unix.vm
|-vm.properties
|-vm.tar.Z
  |-jre
    |-bin
      |-java
    |-lib
```

VM Pack Properties

Every VM pack requires a `vm.properties` file at the root of the VM pack archive.

Table 6-4 • VM Pack Properties

Property	Description
<code>vm.platform</code>	Sets the general platform. Choose from <code>unix</code> or <code>windows</code> .
<code>vm.platform.flavor</code>	<p>Identifies the specific platform the VM supports. Choose from</p> <ul style="list-style-type: none"> • AIX • HP-UX • Linux • solaris • win32 <p>These values are case sensitive and must be used exactly as shown. InstallAnywhere uses the <code>vm.platform.flavor</code> value to determine which VMs to list for each platform. If the value you set doesn't match one of the expected values, your VM pack will not appear in the VM pop-up list on the Build Targets tab.</p>  <p>Note • There are no Mac OS X VM packs. Apple provides JVMs for all OS X systems.</p>
<code>vm.name</code>	Records the VM name as it appears in the VM pop-up list on the Build Targets tab. For example, <code>SunJRE160_01iWin32.vm</code> .
<code>vm.exe.path</code>	<p>Sets the path to the VM pack. Below are several examples of setting this property for different platforms:</p> <pre> vm.platform=unix vm.platform.flavor=solaris vm.name=JavaSoft_jre1.4_Sparc/Solaris vm.exe.path=bin/java</pre> <p>or</p> <pre> vm.platform=windows vm.platform.flavor=win32 vm.name=JavaSoft_jre1.4_Win32 vm.exe.path=bin\java.exe</pre> <p>or</p> <pre> vm.platform=classic mac os vm.platform.flavor=mac os vm.name=MRJ 2.2.5 vm.exe.path=</pre>

Making a VM Pack FIPS-Compliant

Most VM packs are not pre-configured to use a FIPS-compliant JCE (Java Cryptography Extension) security provider. On the **Project > Variables** task, you can enable Java Cryptography Extension encryption and specify an encryption algorithm. Together with a FIPS-compliant JCE library, these settings allow you to create FIPS 140-2 compliant installers.



Task: To make a VM pack FIPS-compliant

1. Extract the contents of the VM pack.
2. Extract the resulting VM archive (`vm.zip` or `vm.tar.Z`).
3. Obtain a FIPS-compliant security provider library that works with the JRE in the VM pack.
4. Place the security provider library in the `lib/ext` directory of the extracted VM archive contents.
5. Modify the security provider list in `lib/security/java.security`. For example, after modifying the IBM JVM security provider it lists

```
security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS  
security.provider.2=com.ibm.crypto.provider.IBMJCE  
security.provider.3=com.ibm.jsse.IBMJSSEProvider  
security.provider.4=com.ibm.jsse2.IBMJSSEProvider2  
security.provider.5=com.ibm.security.jgss.IBMGSSProvider  
security.provider.6=com.ibm.security.cert.IBMCertPath  
...
```

6. Follow the steps in [Creating VM Packs](#) to recreate the VM pack archive.



Note • While modifying the VM pack, you can also add a setting to the `vm.properties` file to set the type of algorithm used by default when **Use JCE Encryption** is checked (in the **Project > Variables** subtask). For example, `vm.algorithm=DES`.

Controlling the VM Used to Run the Installer

This topic explains how to customize the criteria used for the VM search performed by the installer's launcher. When the installer's launcher finds a VM that fits the criteria in the Valid VM List (and no VM is bundled with the installer), it uses that VM to run the installer.

The default VM selection criteria InstallAnywhere uses will accept any JVM version 1.4 or newer. However, you can customize the selection criteria for different versions, different vendors (IBM, SUN, HP, APPLE), and different types (JDK or JRE).



Tip • For a detailed discussion of JVM selection criteria settings, see [About Java VM Selection Criteria](#).

Chapter 6: Adding Advanced Procedures to Installers

Controlling the Install of Bundled VMs



Note • The installer's **Valid VM list** is specified on the **Installer Settings** tab of the **Project > JVM Settings** task.



Task: **To customize the installer's VM search:**

1. Open the **Project > JVM Settings** task.
2. Open the **Installer Settings** tab.
3. Edit the entry in the **Valid VM list** to change the version, vendor, and type criteria to meet your installer's JVM requirements.
4. Click **File > Save**.



Note • There is commonly no need to alter these settings for InstallAnywhere installers themselves, but you may use this criteria to provide a global VM setting that applies to the installer as well as all the LaunchAnywhere launchers the installer deploys. This does not limit your ability to further customize subsequent VM search criteria or launcher-specific VM settings.

Controlling the Install of Bundled VMs

If you bundled one or more VMs in your project, you can control when the VM is deployed and where the installer deploys it.

By default, InstallAnywhere installs any VM you bundle with your installer and uninstalls that VM if the user uninstalls your product. But the VM you bundle with your installer need not be installed to be used to run the installer. (The installer is capable of extracting the JVM to a temporary location during the install and then deleted when the installation is complete.)

The default location to which InstallAnywhere installers store a bundled VM (when installed) is in a `jre` subdirectory of the product's install location. On the **Installer Settings** tab of the **Project > JVM Settings** task, the VM install location is set by the **VM Install Folder** list and text box controls. Default: `User Install Folder/jre`.

If necessary, you can change the magic folder in which the `jre` subdirectory is created or change the name of the subdirectory in which the bundled VM is installed.

Table 6-5 • Controlling the Install of Bundled VMs

Topic	Description
Preventing the Install of the Bundled VM	Describes how to use the Bundled Virtual Machine settings on the Installer Settings tab of the Project > JVM Settings task to prevent your installers from deploying the bundled VM.

Table 6-5 • Controlling the Install of Bundled VMs

Topic	Description
Installing the Bundled VM When the Project Includes a Launcher	Describes how to use the Bundled Virtual Machine settings on the Installer Settings tab of the Project > JVM Settings task to install the bundled VM only when your project includes one or more launchers.
Installing the Bundled VM Only When No Valid VM Exists	Describes how to use the Bundled Virtual Machine settings on the Installer Settings tab of the Project > JVM Settings task to install the bundled VM only when no valid VM exists on the target system.
Preventing the Uninstall of the Bundled VM	Describes how to use the Bundled Virtual Machine settings on the Installer Settings tab of the Project > JVM Settings task to prevent your project's uninstallers from removing the bundled VM when your software is uninstalled.
Changing the Bundled VM Install Folder	Describes how to use the Bundled Virtual Machine settings on the Installer Settings tab of the Project > JVM Settings task to change the magic folder location for the install location of the bundled VM.
Changing the Bundled VM Install Subfolder	Describes how to use the Bundled Virtual Machine settings on the Installer Settings tab of the Project > JVM Settings task to set the subdirectory name for the install location of the bundled VM.

Preventing the Install of the Bundled VM

To prevent the installation of the bundled VM, perform the following steps:



Task: *To prevent your bundled VM from being installed in all cases:*

1. Open the **Installer Settings** tab of the **Project > JVM Settings** task.
2. Clear the selection of the **Install Bundled/Downloaded Virtual Machine** option.



Note • If you deselect the **Install Bundled/Downloaded Virtual Machine** option, it will not be possible to allow users to choose the bundled VM in the **Choose Java VM** panel. The bundled VM, although still available for the installer to use, will not be presented in the **Choose Java VM** panel.

Installing the Bundled VM When the Project Includes a Launcher

To install the bundled VM only when the project includes a launcher, perform the following steps:



Task: *To install your bundled VM only when your project includes a LaunchAnywhere launcher:*

1. Open the **Installer Settings** tab of the **Project > JVM Settings** task.
2. Select the **Only when installing a LaunchAnywhere** option.

Installing the Bundled VM Only When No Valid VM Exists

In some cases, you may want the VM you bundled with your installer to be installed only when the installer fails to find a VM on the target system that matches the VM search criteria in your project.



Task: *To install your bundled VM only when no compatible VM exists on the target system:*

1. Open the **Installer Settings** tab of the **Project > JVM Settings** task.
2. Select the **Only when a compatible VM is not found in the system** option.

This approach gives priority to the installer's VM search—the search the installer performs to find a suitable VM for the launchers the installer deploys. The criteria for this search depend on the **VM Search Settings** you provide on the **General**, **Windows**, and **UNIX** subtabs of the **Search Panel Settings** tab of the **Project > JVM Settings** task.



Note • For more information on these settings, see [Customizing the VM Search Settings for Your Launchers](#) and [Customizing the VM Search Paths and Patterns](#).

In the event that the VM search succeeds, the installer does not install the bundled VM. If the search fails to locate a compatible VM, the installer does install the bundled VM.

Preventing the Uninstall of the Bundled VM

Typically, if an InstallAnywhere installer deploys a bundled VM to an end user's system, the uninstaller removes that VM when the product with which it was bundled is uninstalled. To keep the bundled VM on the end user's system even when the product is uninstalled, you must alter the **Bundled/Downloaded Virtual Machine** settings on the **Installer Settings** tab of the **Project > JVM Settings** task.



Task: *To prevent your bundled VM from being removed during the uninstall:*

1. Open the **Installer Settings** tab of the **Project > JVM Settings** task.
2. Select the **Do not remove bundled/downloaded VM when uninstalling** option.

Changing the Bundled VM Install Folder

To change the VM installation folder of the bundled/downloaded virtual machine, perform the following steps:



Task: *To change the VM Install Folder*

1. Open the **Installer Settings** tab of the **Project > JVM Settings** task.
2. Select a Magic Folder from the **VM Install Folder** list.



Note • It is possible to select one of the user-customizable magic folders (\$USER_MAGIC_FOLDER_n\$) and define the location value of that magic folder with a **Set InstallAnywhere Variable** action in the **Pre-Install** task.

Changing the Bundled VM Install Subfolder

To change the bundled VM installation subfolder, perform the following steps:



Task: *To change the subdirectory:*

1. Open the **Installer Settings** tab of the **Project > JVM Settings** task.
2. In the text box next to the **VM Install Folder** field, enter the name of the subdirectory to which you want the bundled VM installed. The default value is `jre`.



Note • Users can change the name of the subdirectory in which the bundled JVM is installed but cannot create a multi-level subdirectory structure under the magic folder you choose for VM Install Folder.

Controlling the VM Your Launchers Use

The Java virtual machine used by a launcher your installer deploys depends on the target environment as well as various VM-related project settings. The tasks related to the settings the installer uses to select a VM for your launchers have been collected in the following topic groups.

Table 6-6 • Controlling the VM Your Launchers Use

Topic	Description
Customizing the VM Search Settings for Your Launchers	Explains how to customize the VM search settings on the Search Panel Settings tab of the Project > JVM Settings task.
Customizing the VM Search Paths and Patterns	Describes how to change the default Search Path and Java Executable Pattern settings for Windows and Unix target systems.
Customizing Individual Launcher Settings	Describes how to use the options on the Advanced Settings tab of a launcher's customizer (the Create LaunchAnywhere for Java Application customizer).

Customizing the VM Search Settings for Your Launchers

The **VM Search Settings** on the **General** subtab of the **Search Panel Settings** tab of the **Project > JVM Settings** task are the main controls that govern the criteria used in the installer's VM search: the search the installer does for the launchers it deploys. There are three options for customizing the VM Search settings for your launchers:

- **Installer's valid VM list**—You can use the criteria expressed in the installer's valid VM list. See [Using the Installer Valid VM List](#).
- **Valid VM list of all LaunchAnywhere actions**—You can use the synthesis of criteria specified in the `1ax.n1.valid.vm.list` property of all your project's launchers. See [Using a Valid VM List of All LaunchAnywhere Actions](#).
- **Specific valid VM list**—You can use a custom valid VM list that you provide specifically for the installer's VM search. See [Using a Specific Valid VM List](#).



Note • This section addresses the configuration of the **VM Search Settings** on the **General** subtab of the **Search Panel Settings** tab of the **Project > JVM Settings** task. For information on configuring the `1ax.n1.valid.vm.list` property for individual launchers, see [Customizing Individual Launcher Settings](#).

Using the Installer Valid VM List

To use the Installer Valid VM List to govern the criteria used in the installer's VM search, perform the following steps:



Task: *To use the installer Valid VM List criteria:*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **General** subtab under **VM Search Settings**, select the **Use installer's valid VM list** option. Selecting this option causes the installer to use the criteria the installer's launcher uses in the "bootstrap" phase.



Note • *The installer's Valid VM list is specified on the **Installer Settings** tab of the **Project > JVM Settings** task. The currently specified value is listed next to the **Use installer's valid VM list** option.*

Using a Valid VM List of All LaunchAnywhere Actions

To use a synthesis of the criteria specified in the individual launchers included in your project, perform the following steps:



Task: *To use a synthesis of the criteria specified in the individual launchers included in your project:*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **General** subtab under **VM Search Settings**, select the **Use the valid VM list of all LaunchAnywhere actions** option. Selecting this option causes the installer to combine the `1ax.n1.valid.vm.list` property values of all your project's launchers. The installer uses a synthesis of these settings to find a VM that is valid for all your project's launchers.

For example, if your project includes three launchers—two using the default 1.4+ criteria but one using SUN 1.5+—the installer's search will require the vendor SUN and the version 1.5+ for all launchers.

Using a Specific Valid VM List

To use a custom valid VM list for the installer's VM search, perform the following steps:



Task: *To use a custom valid VM list for the installer's VM search:*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **General** subtab under **VM Search Settings**, select the **Use a specific valid VM list** option.
3. In the text box, enter the VM search criteria that you want the installer to use.



Tip • *For a detailed discussion of JVM selection criteria settings, see [About Java VM Selection Criteria](#).*

Customizing the VM Search Paths and Patterns

The conventional locations InstallAnywhere installers search on Windows and Unix systems for VMs are reflected in the default settings on the **Windows** and **UNIX** tabs of the **Project > JVM Settings** task. Likewise, the default **Java Executable Patterns** settings in these locations reflect the search patterns InstallAnywhere installers use to identify Java executables. InstallAnywhere allows you to customize both the search paths and the Java executable patterns your project's installers use.



Note • These settings do not directly apply to your project's LaunchAnywhere launchers. They are potentially indirectly applied to your project's Windows and Unix launchers when the launcher's **Advanced Settings** indicate it will use either the **VM Selected by the installer or by the end user via Choose VM Panel or First VM found in the system matching the VM Search Settings defined under Project > JVM Settings**. However, should a LaunchAnywhere launcher perform an independent JVM search or use the VM used by the installer, these custom search paths and Java executable patterns will not be used.

The following table lists topics that explain how to customize the VM search paths and patterns:

Table 6-7 • Customizing the VM Search Paths and Patterns

Topic	Description
Customizing Windows Search Paths	Describes how to change the default Windows JVM Search Paths on the Windows subtab of the Search Panel Settings tab of the Project > JVM Settings task.
Customizing Java Executable Patterns for Windows Installs	Explains how to customize the default patterns the installer uses to identify Java executables on Windows target systems.
Customizing Unix Search Paths	Describes how to change the default Unix JVM Search Paths on the UNIX subtab of the Search Panel Settings tab of the Project > JVM Settings task.
Customizing Java Executable Patterns for Unix Installs	Explains how to customize the default patterns the installer uses to identify Java executables on Unix target systems.

Customizing Windows Search Paths

For installers that target Windows systems, you can customize search path settings in the following ways:

- Adding a Search Path for Windows Systems
- Adding a Windows Registry Search to the Search Paths List.
- Adding a Path Environment Variable Search to the Search Paths List
- Removing a Search Path from Windows Systems
- Editing an Existing Windows Search Path
- Changing the Directory Depth for a Windows Search Path

Adding a Search Path for Windows Systems

To add a search path for Windows systems, perform the following steps:



Task: *To add a search path for Windows systems:*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Windows** subtab, open the **Search Paths** tab.
3. On the **Search Paths** customizer, click **Add**. A new entry to the **Search Paths** list.
4. Click the new blank row of the **Search Paths** list to place your cursor in the text box.
5. Enter the absolute path to the directory you want the installer to search for Java executables.



Note • You cannot use InstallAnywhere variables or Magic Folders in the path statement.



Note • By default, new search paths only search the top level of the directory you specify. No subdirectories are searched.

Adding a Windows Registry Search to the Search Paths List

To add a Windows Registry search to the Search Paths list, perform the following steps:



Task: *To add a Windows registry search to the Search Paths list:*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Windows** subtab, open the **Search Paths** tab.

Chapter 6: Adding Advanced Procedures to Installers

Controlling the VM Your Launchers Use

3. On the **Search Paths** customizer, click **Add Windows Registry**. The Windows Registry Search is added to the Search Paths list.



Note • The Windows Registry Search examines standard keys in the Windows registry. Users cannot customize the Windows Registry Search.

Adding a Path Environment Variable Search to the Search Paths List

To add a path environment variable search to the Search Paths list, perform the following steps:



Task:

To add Path Environment Variable Search to the Search Paths list

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Windows** subtab, open the **Search Paths** tab.
3. On the **Search Paths** customizer, click **Add Path Environment Variable**. The Path Environment Variable Search is added to the Search Paths list.



Note • The Path Environment Variable Search scans the top-level of any directories in the PATH environment variable.

Removing a Search Path from Windows Systems

To remove a search path from Windows systems, perform the following steps:



Task:

To remove a search path from Windows systems:

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Windows** subtab, open the **Search Paths** tab.
3. Click on the entry in the Search Paths list you want to delete and click **Remove**.

Editing an Existing Windows Search Path

To edit an existing Windows search path, perform the following steps:



Task:

To edit an existing Windows search path:

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Windows** subtab, open the **Search Paths** tab.

3. In the Search Paths list, click the path you want to edit. This activates the entry and places the cursor at the end of the path statement.
4. Enter your path changes.



Note • You cannot edit search path settings for Path Environment Variable Search and Windows Registry Search but you can add or remove them as necessary.

Changing the Directory Depth for a Windows Search Path

To change the directory depth for a Windows search path, perform the following steps:



Task: **To change the directory depth for a Windows search path:**

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Windows** subtab, open the **Search Paths** tab.
3. In the Search Paths list, double-click the level and choose one of the following options:
 - **Top Level**—Only searches within the immediate directory. No subdirectories.
 - **First Level**—Searches the immediate directory and its first-level subdirectories.
 - **All Levels**—Searches the immediate directory and all subdirectories.



Note • Searching **All Levels** results in a very thorough search but can significantly increase the time it takes to perform the search. However, the installer stops searching in any directory as soon as it finds a VM in that directory.

Customizing Java Executable Patterns for Windows Installs

The default search patterns InstallAnywhere installers use on Windows systems include

- java.exe
- bin\java.exe
- jre\bin\java.exe

You can add, remove, or edit patterns in the **Java Executable Patterns** list.



Note • Once a pattern matches against any directory, that directory is considered a VM home directory and none of its subdirectories are searched.

Chapter 6: Adding Advanced Procedures to Installers

Controlling the VM Your Launchers Use

Adding a New Search Pattern

To add a new search pattern, perform the following steps:



Task: *To add a new search pattern to the Java Executable Patterns list*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Windows** subtab, open the **Java Executable Patterns** tab.
3. Click **Add**, and enter the new search pattern.

Removing a Search Pattern

To remove a search pattern, perform the following steps:



Task: *To remove a search pattern from the Java Executable Patterns list:*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Windows** subtab, open the **Java Executable Patterns** tab.
3. Click the search pattern you want to delete and click **Remove**.

Editing an Existing Search Pattern

To edit an existing search pattern, perform the following steps:



Task: *To edit an existing search pattern in the Java Executable Patterns list:*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Windows** subtab, open the **Java Executable Patterns** tab.
3. Click the search pattern you want to edit and enter your changes.

Customizing Unix Search Paths

For installers that target Unix systems, you can customize search path settings in the following ways:

- [Adding a Search Path for Unix Systems](#)
- [Adding a Path Environment Variable Search to the Unix Search Paths List](#)
- [Removing a Search Path from Unix Systems](#)
- [Editing an Existing Unix Search Path](#)
- [Changing the Directory Depth for a Unix Search Path](#)

Adding a Search Path for Unix Systems

To add a search path for Unix systems, perform the following steps:



Task: *To add a search path for Unix systems*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **UNIX** subtab, open the **Search Paths** tab.
3. On the Search Paths customizer, click **Add**. A new entry is added to the **Search Paths** list.
4. Click the new blank row of the **Search Paths** list to place your cursor in the text box.
5. Enter the absolute path to the directory you want the installer to search for Java executables.



Note • You cannot use *InstallAnywhere variables* or *Magic Folders* in the path statement.



Note • By default, new search paths only search the top level of the directory you specify. No subdirectories are searched.

Adding a Path Environment Variable Search to the Unix Search Paths List

To add a path environment variable search to the Unix search paths list, perform the following steps:



Task: *To add a Path Environment Variable Search to the Unix Search Paths list*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **UNIX** subtab, open the **Search Paths** tab.
3. On the **Search Paths** customizer, click **Add Path Environment Variable**. A Path Environment Variable Search is added to the **Search Paths** list.



Note • The Path Environment Variable Search scans the top-level of any directories in the PATH environment variable.

Removing a Search Path from Unix Systems

To remove a search path from Unix systems, perform the following steps:

**Task:** *To remove a search path from Unix systems*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **UNIX** subtab, open the **Search Paths** tab.
3. Click on the entry in the **Search Paths** list you want to delete and click **Remove**.

Editing an Existing Unix Search Path

To edit an existing Unix search path, perform the following steps:

**Task:** *To edit an existing Unix search path:*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **UNIX** subtab, open the **Search Paths** tab.
3. In the **Search Paths** list, click the path you want to change. This activates the entry and places the cursor at the end of the path statement.
4. Enter your path changes.



Note • You cannot edit search path settings for Path Environment Variable Search but you can add or remove it as necessary.

Changing the Directory Depth for a Unix Search Path

To change the directory depth for a Unix search path, perform the following steps:

**Task:** *To change the directory depth for a Unix search path:*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **UNIX** subtab, open the **Search Paths** tab.
3. In the **Search Paths** list, double-click the level and choose one of the following options:
 - **Top Level**—Only searches within the immediate directory. No subdirectories.
 - **First Level**—Searches the immediate directory and its first-level subdirectories.
 - **All Levels**—Searches the immediate directory and all subdirectories.



Note • Searching **All Levels** results in a very thorough search but can significantly increase the time it takes to perform the search. However, the installer stops searching in any directory as soon as it finds a VM in that directory.

Customizing Java Executable Patterns for Unix Installs

The default search patterns InstallAnywhere installers use on Unix systems include

- java
- bin\java
- jre\bin\java

You can add, remove, or edit patterns in the **Java Executable Patterns** list.



Note • Once a pattern matches against any directory, that directory is considered a VM home directory and none of its subdirectories are searched.

Adding a New Search Pattern

To add a new search pattern, perform the following steps:



Task:

To add a new search pattern to the Java Executable Patterns list:

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Unix** subtab, open the **Java Executable Patterns** tab.
3. Click **Add**, and enter the new search pattern.

Removing a Search Pattern

To remove a search pattern, perform the following steps:



Task:

To remove a search pattern from the Java Executable Patterns list:

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Unix** subtab, open the **Java Executable Patterns** tab.
3. Click the search pattern you want to delete and click **Remove**.

Chapter 6: Adding Advanced Procedures to Installers

Controlling the VM Your Launchers Use

Editing a Search Pattern

To edit a search pattern, perform the following steps:



Task: *To edit an existing search pattern in the Java Executable Patterns list:*

1. Open the **Search Panel Settings** tab of the **Project > JVM Settings** task.
2. On the **Unix** subtab, open the **Java Executable Patterns** tab.
3. Click the search pattern you want to edit and enter your changes.

Customizing Individual Launcher Settings

The primary controls for individual launchers appear on the **Advanced Settings** tab of the **Create LaunchAnywhere for Java Application** action's customizer.



Note • For a detailed discussion of how the launcher's **Advanced Settings** options affect VM selection, see [About the Launcher's VM Selection Behavior at Run-Time](#).

The following topics provide instructions on using the **Advanced Settings** options to achieve specific goals.

Table 6-8 • Customizing Individual Launcher Settings

Topic	Description
Causing a Launcher to Use the VM That Ran the Installer	Explains how to use the launcher's Advanced Settings to force the launcher to use the same VM used to run the installer. This is either the bundled VM or the VM selected by the installer's launcher.
Causing a Launcher to Use the VM Selected During the Installer's VM Search	Explains how to use the launcher's Advanced Settings to force the launcher to use the VM selected by the VM search the installer performs.
Causing a Launcher to Perform an Independent VM Search	Explains how to use the launcher's Advanced Settings to prevent the installer from assigning a VM selection to the launcher.

Causing a Launcher to Use the VM That Ran the Installer

This process forces the launcher to use the VM under which the installer was run. When the VM bundled with the installer is present, the launcher will use that VM. If no VM was bundled with the installer, the launcher uses the Installer Valid VM List (from the **Installer Settings** tab of the **Project > JVM Settings** task) as its value for `lax.nl.valid.vm.list`.



Task: **To use the VM under which the installer was run:**

1. Open the **Install** task.
2. Select the launcher you want to modify. The **Create LaunchAnywhere for Java Application** customizer opens.
3. On the **Advanced Settings** tab, select the **VM used by the installer** option.
4. Click **File > Save**.

Causing a Launcher to Use the VM Selected During the Installer's VM Search

This process forces the launcher to use the first VM found that meets the criteria specified in the **Project > JVM Settings** task.



Task: **To use the VM selected during the installer's VM search**

1. In the Advanced Designer, open the **Install** task.
2. Select the launcher you want to modify. The **Create LaunchAnywhere for Java Application** customizer opens.
3. Open the **Advanced Settings** tab.
4. Select **The first VM found in the system matching the VM Search Settings defined under Project > JVM Settings**.
5. Click **File > Save**.



Note • The launcher's VM selection criteria are set at install time. Therefore, the launcher itself does not use any customized **Search Paths** or **Java Executable Patterns** you set on the **Windows** and **UNIX** subtabs of the **Search Panel Settings** tab of the **Project > JVM Settings** task.

Causing a Launcher to Perform an Independent VM Search

The final option on the **Advanced Settings** tab of the **Create LaunchAnywhere for Java Application** action's customizer prevents the installer from assigning a VM to the launcher. The benefit of this option is that it allows the launcher to perform an independent VM search at the time it is executed.

When you select this option, the launcher uses the value in its `lax.n1.valid.vm.list` property (default value: 1.4+) as the VM search criteria.



Task: *To perform an independent VM search:*

1. In the Advanced Designer, open the **Install** task.
2. Select the launcher you want to modify. The **Create LaunchAnywhere for Java Application** customizer opens.
3. Open the **Advanced Settings** tab.
4. Select **No specific VM**.
5. Click **File > Save**.



Note • You can customize the default value (1.4+) of a launcher's `tax.n1.valid.vm.list` property. To do so, click the **Edit Properties** button on the **General Settings** tab of the **Create LaunchAnywhere for Java Application** customizer. This opens the **LaunchAnywhere Properties** dialog box. Locate the `tax.n1.valid.vm.list` property, double-click its value, and edit the text. Click **OK** to close the **LaunchAnywhere Properties** dialog box.

For a detailed discussion of JVM selection criteria settings, see [About Java VM Selection Criteria](#).

Using the Choose Java VM Panel

The **Choose Java VM** panel provides your end users with the option to choose a VM they want your software to use.

Table 6-9 • Using the Choose Java VM Panel

Topic	Description
Adding a Choose Java VM Panel	Describes the basic steps for adding a Choose Java VM panel to your project.
Allowing Users to Choose Only the VMs the Installer's Search Finds	Explains how to customize the Choose Java VM panel so that VMs found by the installer's VM search are the only available choices.
Allowing Users to Choose the Bundled VM	Explains how to add the bundled VM to the VM choices available to your end users.
Allowing Users to Search for VMs in Non-Standard Locations	Explains how to add the Search Another Location button to your Choose Java VM panel.
Allowing Users to Browse for a VM on the Local System	Explains how to add the Choose Java Executable button to your Choose Java VM panel.

Adding a Choose Java VM Panel

The **Choose Java VM Panel** is available to the **Pre-Install** task.



Task: ***Adding a Choose Java VM panel***

1. In the Advanced Designer, click the **Pre-Install** task.
2. On the Pre-Install task, click **Add Action**.
3. In the Choose an Action dialog box, click the **Panel** tab.
4. On the Panel tab, double-click **Choose Java VM**.

Allowing Users to Choose Only the VMs the Installer's Search Finds

The **Choose Java VM** panel can be used to provide end users with a choice of Java VMs that match the installer's search criteria. This means that the VM search settings on the **General** subtab of the **Search Panel Settings** tab of the **Project > JVM Settings** task provide the main criteria set and that, on Windows and Unix systems, the search is conducted in the search paths and search patterns specified on the **Windows** and **UNIX** subtabs, respectively.



Task: ***To configure the Choose Java VM panel to show only VMs found by the installer's VM search:***

1. Open the **Pre-Install** task.
2. Click **Add Action**. The Choose an Action dialog box opens.
3. On the **Panel** tab, select **Choose Java VM**. The **Choose Java VM** panel is added to the list and its customizer opens.
4. In the **Choose Java VM** panel customizer, de-select the following check boxes:
 - **Allow the end user to install the bundled VM**
 - **Allow the end user to search for locations other than the paths defined under Project > JVM Settings > Search Panel Settings**
 - **Allow the end user to choose a specific Java executable**
5. Click **File > Save**.

The **Choose Java VM** panel enables these check boxes by default, but to restrict a user's options to only those VMs found by the installer's VM search, you must disable them. The resulting **Choose Java VM** panel shows only a list of the VMs on the target system that match the installer's VM search criteria.



Note • You can exercise tighter control over what VMs appear in the **Choose Java VM** panel by altering the **VM Search Settings** on the **General** subtab of the **Search Panel Settings** tab of the **Project > JVM Settings** task and the **Search Paths** and **Java Executable Patterns** defined on the **Windows** and **UNIX** subtabs.

Allowing Users to Choose the Bundled VM



Note • The option to allow users to choose the bundled VM is enabled, by default, when you first add a **Choose Java VM** panel. These instructions are only necessary to re-enable this feature in an existing **Choose Java VM** panel that had previously had the option disabled.

Often setup authors who bundle a VM with their installer want their end users to be able to choose the bundled VM during the install. This procedure adds an **Install a Java VM Specifically for this Application** option to the **Choose Java VM** panel. When an end user selects that option, the installer configures its launchers to use the bundled VM.



Task: **To allow users to choose the bundled VM:**

1. Open the **Pre-Install** task.
2. Select the **Choose Java VM** panel. The **Choose Java VM** customizer opens.
3. On the **General Settings** tab of the customizer, select the **Allow the end user to install the bundled VM** option.
4. Click **File > Save**.

When you build the project and test an installer that includes a bundled VM, the **Choose Java VM** panel shows both the VM search results and the **Install a Java VM Specifically for this Application** option.



Note • The **Choose Java VM** panel never determines whether or not to install the bundled VM; instead this is something that is determined by the **Bundled/Downloaded Virtual Machine** settings on the **Installer Settings** tab of the **Project > JVM Settings** task. The **Choose Java VM** panel provides the option of allowing the launchers the installer deploys to run under the bundled VM.

Allowing Users to Search for VMs in Non-Standard Locations



Note • This option is enabled, by default, when you first add a **Choose Java VM** panel. These instructions are only necessary to re-enable searching for VMs in non-standard locations in an existing **Choose Java VM** panel that had previously had the option disabled.

To avoid long wait times, InstallAnywhere installers only search the most common locations for Java executables. Within the **Choose Java VM** panel, you can provide users with the option of searching for VMs in locations other than the most common ones.



Tip • For Windows and Unix installers, you can customize the search paths and search patterns this search uses. See [Customizing the VM Search Paths and Patterns](#) for details.

This process adds a **Search Another Location** button to the **Choose Java VM** panel. An end user can click the **Search Another Location** button and select a folder in which to search for a VM. The search criteria you set in your InstallAnywhere project applies to this search as well.



Task: **To allow users to search for VMs in non-standard locations:**

1. Open the **Pre-Install** task.
2. Select the **Choose Java VM** panel. The **Choose Java VM** customizer opens.
3. On the **General Settings** tab, select the **Allow the end user to search for locations other than the paths defined under Project > JVM Settings > Search Panel Settings** option.
4. Click **File > Save**.

Allowing Users to Browse for a VM on the Local System



Note • This option is enabled, by default, when you first add a **Choose Java VM** panel. These instructions are only necessary to re-enable browsing for VMs in an existing **Choose Java VM** panel that had previously had the option disabled.

For maximum flexibility, you can also permit your end users to browse their systems and select the VM they want to use. This process adds a **Choose Java Executable** button to the **Choose Java VM** panel. An end user can click the **Choose Java Executable** button and browse the local file system for the JVM they want.



Task: *To allow users to browse for a VM:*

1. Open the **Pre-Install** task.
2. Select the **Choose Java VM** panel. The **Choose Java VM** customizer opens.
3. On the **General Settings** tab, select the **Allow the end user to choose a specific Java executable** option.
4. Click **File > Save**.

JVM Spec Files

InstallAnywhere provides a set of JVM spec files which target all the supported platforms. However, you can customize these files as well as add new files to the existing set.

- [About JVM Spec Files](#)
- [Guidelines for Writing a JVM Spec File](#)

About JVM Spec Files

A JVM spec file, or JVM search instructions file, is a flat file with an extension of `.jvm`. Each operating system has its own set of search files catering to different versions of the JVM, as well as belonging to different vendors such as IBM, Sun Microsystems, HP-UX, .etc. Examples of JVM spec file names are:

```
ibm_win32_14.jvm  
ibm_aix_14X.jvm  
sun_linux_jre15.jvm
```

JVM spec files contain a set of JVM search hints in the form of key-value pairs. The following is an example of a JVM spec file:

```
DESC: Sun Microsystems Java Runtime Environment (JRE) 1.5 for Linux  
JVM_EXE:bin/java  
PLATFORM_HINT:  
JDK_HOME  
JAVAHOME  
JAVA_HOME  
/:  
JVM_PROPERTIES:  
java.vendor=Sun...  
java.version=1.5...  
/:  
PATH_HINT:  
/usr/jre1.5.0  
/usr/local/jre1.5.0  
/usr/java/jre1.5.0  
/:
```

The text `/:` is used to separate those keys which can have multiple values.

During the launch of the installer, the JVM spec file will be read by the InstallAnywhere installer launcher and the hints are used to determine the location where the JRE is available on the target system.

You can instruct the installer to use more than one JVM spec file at the same time by selecting more than one file in the **JVM Search Settings** list on the **Build Targets** subtab of the **Build Configurations** tab of the **Build** task. The order in which the JVM spec files are listed in the **JVM Search Settings** list is important. The installation will attempt to find a matching JRE using the first JVM spec file specified in the list, then work its way down the list.

InstallAnywhere provides a set of JVM spec files which target all the supported platforms. However, you can customize these files as well as add new files to the existing set.

Guidelines for Writing a JVM Spec File

If you write your own JVM spec file, keep in mind the following guidelines:

Table 6-10 • Guidelines for Writing a JVM Spec File

Guideline	Description
Use a Unix-compliant format	JVM spec files should be written on a Unix compliant format (without Ctrl-M characters).
Maintain the proper format for the JVM_EXE key	InstallAnywhere restricts the JVM_EXE key in the JVM spec file to have a value of: <ul style="list-style-type: none"> Windows platforms: bin\java.exe Unix-based platforms: bin/java You should maintain the same format when customizing the spec file.
PATH_HINT should contain the JRE path only until the JRE folder	The PATH_HINT in the JVM spec file should contain the JRE path only until the JRE folder. For example, if the JRE is present at the following location: <ul style="list-style-type: none"> Windows: C:\Program Files\jre1.5.07 Unix: /usr/java5_64/jre the PATH_HINT should be: <ul style="list-style-type: none"> Windows: \Program Files\jre1.5.07 Unix: /usr/java5_64/jre  <p>Note • For Windows, you should not mention the drive letter (such as C: or D:) in the PATH_HINT section.</p>
Wildcard (*) character can be used in the PATH_HINT section	The wildcard (*) character can be used in the PATH_HINT section.

Table 6-10 • Guidelines for Writing a JVM Spec File

Guideline	Description
Regular expressions limited to ending ellipsis or wildcard character	Regular expressions in the JVM_PROPERTIES section are limited to ending ellipsis (...) or wildcard (*) character.
Locations where JVM spec files can be stored	<p>By default, all JVM spec files are installed in the <IA_HOME>\resource\jvms directory. However, these files can be present at any location on the machine where InstallAnywhere is installed.</p> <p>In cases where a JVM spec file is located in a directory other than <IA_HOME>\resource\jvms, you need to identify the location of those JVM spec files by doing one of the following:</p> <ul style="list-style-type: none"> • InstallAnywhere Preferences dialog box—You can specify the location of the JVM spec files in the JVM Specs Resource Paths field on the Resources tab of the InstallAnywhere Preferences dialog box. In this field, enter a semicolon-separated path list where JVM spec files are located. InstallAnywhere scans the paths listed here and populates the Choose JVM Spec dialog box with the JVM spec files it finds on these paths as well as those stored at the default JVM spec file location (<IA_HOME>\resource\jvms). • Environment variable—You can specify the location of the JVM spec files by setting the IA_JVM_SPECS_PATHS environment variable to a semicolon-separated list of paths. <p>Instead of adding each platform-specific directory to this environment variable, such as:</p> <pre>IA_JVM_SPECS_PATH=C:\IDEs\IA2011RC1\resource\jvms\aiX; C:\IDEs\IA2011RC1\resource\jvms\generic-unix; C:\IDEs\IA2011RC1\resource\jvms\hpux; C:\IDEs\IA2011RC1\resource\jvms\linuX....</pre> <p>you should point this environment variable to the directory which contains all of the platform-specific directories, such as:</p> <pre>IA_JVM_SPECS_PATH=C:\IDEs\IA2011RC1\resource\jvms</pre> <p></p> <p>Note • <i>The organization of the JVM spec files must be similar to that of \$IA_HOME\$/resource/jvms directory; each spec file must be placed in its own platform folder, such as:</i></p> <ul style="list-style-type: none">  aiX  generic-unix  hpux  linuX  solaris  unix  windows

Table 6-10 • Guidelines for Writing a JVM Spec File

Guideline	Description
Restart is required if Spec file is added or modified	If a new JVM spec file is added or an existing spec file is modified, the InstallAnywhere interface needs to be restarted for the changes to take effect.

Generating Response Files

You can generate a response file (commonly named `installer.properties`) by clicking the **Always Generate Response Files** check box on the **Project > Info** task or by specifying `-r` at the command line. By default, all of the variables defined in your installation are recorded.

- [Generating Response Files By Selecting the “Always Generate Response Files” Option](#)
- [Generating Response Files Using the -r Command Line Option](#)



Note • By default, a response file is named `installer.properties` or `[installername].properties` and it is created in the same directory as the installer.

Generating Response Files By Selecting the “Always Generate Response Files” Option

The following procedure explains how to automatically generate a response file by selecting the **Always Generate Response Files** option and to exclude selected variables from that file.



Task: *To automatically generate a response file and exclude variables:*

1. On the **Advanced Designer**, click **Project**. The **Project** task opens.
2. On the **Project** task, click **Info**. The **Info** subtask opens.
3. Click the **Always Generate Response Files** check box.
4. On the **Project** task, click **Variables**. The **Variables** subtask opens.
5. Under **Configure Variables**, click **Configure**. The **Configure Variables** dialog box opens.
6. Click **Add**. A blank row appears in the list.
7. In the **Variable Name** column, enter the name of the variable you want to exclude.
8. Double-click the blank box in the **Options** column and then select either:
 - **Exclude Variable Entirely** to exclude the variable name and value in the response file.
 - **Exclude Value Only** to exclude only the value for the specified value in the response file.
9. Click **OK**.

Generating Response Files Using the -r Command Line Option

You can force the creation of a response file without using the **Always Generate Response File** option by using the -r command line option, and you can use it in combination with other options, as shown in the following examples:

- Specifying a Different Response File Location
- Specifying a Different Response File Location and Response File Name



Important • The -r command line option requires at least one argument.

Specifying a Different Response File Location

To generate a response file in a different location than the default location, you can specify the path in the command line:

```
install.exe -r <C:\destinationPathToResponseFile>
```

such as:

```
install.exe -r C:\IAProjects\ResponseFiles
```

In this scenario, the `installer.properties` response file will get generated at the specified location, provided the directory or path is accessible to the user.

Specifying a Different Response File Location and Response File Name

To generate a response file in a different location than the default location using a different name, you can specify the path and new name in the command line:

```
install.exe -r <C:\PathToResponseFile\ResponseFileName.txt>
```

such as:

```
install.exe -r C:\IAProjects\ResponseFiles\MyResponseFile.txt
```

In this scenario, a response file named `MyResponseFile.txt` will be generated at the specified location, provided the directory or path is accessible to the user. Also, if a file with the same name exists at the specified location, the user must have permission to overwrite that file.



Note • If you have the **Always Generate Response File** option selected on the **Project > Info** subtask but you want to specify a different location and/or file name by using the command line, the command line arguments will be honored.

Generating Log Files

Logging is one of the ways to record the results during runtime, reporting the success and/or failure execution details that occur. Log files are not generated by default. You can set your project to build installers that do create log files.

On the **Log Settings** subtask, you can enable logging during installation and also during the Maintenance Mode options of **Uninstall**, **Add Features**, **Repair Installation**, and **Remove Features**.



Edition • Logging during the Maintenance Mode phases of Add Features, Repair Installation, Remove Features, and Uninstall is only available in InstallAnywhere Enterprise Edition.



Task:

To enable logging:

1. On the **Project > Log Settings** subtask, select **Enable Logging**.
2. Under **Mode**, select the phases for which you want to enable logging: **Install**, **Add**, **Repair**, **Remove**, and/or **Uninstall**.
3. In the **Path** field for each selected phase, the default path for the log file is:

\$USER_INSTALL_DIR\$\$\\$_\$PRODUCT_NAME\$_installation\$\\$Logs\$/

You can customize this path by editing these fields. Click **Restore Default Paths** to reset all log paths to the default settings.



Note • You can also enter absolute paths as the log directory path. In the case of an invalid path, the log will be generated in the default location.

4. Under **Log Format**, select **Plain text format** (default) or **XML format**.
5. By default, the **Uninstall all logs** option is selected. Clear this option to prevent the uninstaller from removing the logs when your software is uninstalled. The most recent uninstall log will not be removed.
6. To include debug information in the log files, select **Include debug output (stderr and stdout)**.
7. To direct debug output to a specific location, enter one of the following in the **Send stderr to** and **Send stdout to** fields:
 - **Console**—Enter **console** to send stderr or stdout output to a console window.
 - **File**—Enter a file name to send stderr or stdout output to a file. For example, **installer_stderr.txt** causes the installer to create a file with that name, in the same location as the installer, and record the stderr output there.

- **Platform-neutral paths**—Use `$/` or $\$, Java properties ($prop.*$), or LAX environment variables ($lax.n1.env.*$ or $lax.n1.env.exact_case.*$) to specify platform-neutral paths for the debug output. (LAX environment variables are not supported for Pure Java installers or Mac OS X installers.)`
- **No output**—Leave blank to discard stderr or stdout output.



Note • If the **Include debug output (stderr and stdout)** option under **Uninstall & Debug Options** is selected, these options are disabled.

Getting User Input

This section provides examples on the use of InstallAnywhere's Get User Input panels.

Table 6-11 • Getting User Input

Topic	Description
Using the Get User Input - Simple Panel	Walks through an example of the Get User Input - Simple panel.
Using the Get User Input - Advanced Panel	Illustrates the use of the Get User Input - Advanced panel with an example that acquires a username and password.

Using the Get User Input - Simple Panel



Edition • This panel is available only for InstallAnywhere Enterprise edition users.

This action provides you with the ability to create a simplified custom panel that uses either text fields, check boxes, radio buttons, pop-up menus, or lists.



Note • Only one type of control can be used in each **Get User Input - Simple** panel. To employ a mix of controls, use a **Get User Input - Advanced** panel.



Task: [To create a Get User Input - Simple panel](#)

1. On the Pre-Install, Post-Install, Pre-Uninstall, or Post-Uninstall task, click **Add Action**.
2. From the Choose an Action dialog box, click **Panel** and then click **Get User Input - Simple**.
3. Choose an **Input Method**. Choose from **Textfields**, **Checkboxes**, **Radio Buttons**, **Popup Menu**, or **List**.

4. Click **Configure** to add, edit, or remove controls.
5. Click **OK** to close the Configure Input Items dialog box.



Tip • Click **Preview** at any point to view the controls you have added to the panel so far.

Example: Obtaining Server Setup Information

In this example, you use text field controls on a **Get User Input - Simple** panel to create a Server Setup panel that gets the IP address, port, and a contact email address.



Task: To create a Server Settings (Get User Input - Simple) panel

1. On the Pre-install task, click **Add Action**.
2. From the Choose an Action dialog box, click **Panel** and then click **Get User Input - Simple**.
3. On the Get User Input - Simple customizer, type the Title, **Server Setup**.
4. Click **Configure**. (The default Input Method is Textfields.)
5. In the Configure Input Items dialog box, click **Add**.
6. Type the Label, **IP Address**, and press the **[Tab]** key.
7. Type the Default Value, **127.0.0.1**.
8. Click **Add**.
9. Type the Label, **Port**, and press the **[Tab]** key.
10. Type the Default Value, **8080**.
11. Click **Add**.
12. Type the Label, **Server Contact**, and press the **[Tab]** key.
13. Type the Default Value, **yourname@yourdomain.com**.
14. Click **OK**.
15. Click **Preview** to show the panel with the current controls.



Tip • The Preview feature does not resolve variable values. To see your panel in a truly live setting, build the project and run the installer.

Using the Get User Input - Advanced Panel



Edition • This panel is available only for InstallAnywhere Enterprise edition users.

This action allows you to create a custom panel with a variety of configurable controls you can use to obtain input from users. You may add an unlimited number of varied input items to a single panel.



Task: *To create the Get User Input - Advanced panel example*

1. On the Pre-Install, Post-Install, Pre-Uninstall, or Post-Uninstall task, click **Add Action**.
2. From the Choose an Action dialog box, click **Panel** and then click **Get User Input - Advanced**.
3. Add the panel components. Choose from **Textfields**, **Choice Groups**, **Labels**, or **File Choosers**.
4. Choose a component and click **Configure Selection** to customize its settings.
5. Click **Preview Panel** to see the panel with the current components.



Tip • Click **Preview Panel** at any point to view the controls you have added to the panel so far.

Example: Verify Username and Password Information



Note • InstallAnywhere includes a **Get Password** panel specifically for collecting and verifying passwords. Still, this exercise provides a good example to illustrate the use of the **Get User Input - Advanced** panel for a similar custom panel.

This example includes three text fields. One text field is for the username. The two password panels have shadowed Input Echo Characters so an end user's password is hidden as it is entered.



Task: *To create a Verify Username and Password (Get User Input - Advanced) panel*

1. On the Get User Input - Advanced customizer, type the Title, **Verify Username and Password**.
2. Click **Add Textfield**. The text field appears in the component table.
3. Click **Configure Selection**. The Configure Textfield dialog box appears.
4. In the Configure Textfield dialog box, type a **Caption**, such as **Please type your username**.
5. In the Label text box, type **Username**.
6. Click **OK**.
7. Click **Add Textfield** to add a new text field for the password.

8. In the Configure Textfield dialog box, type a Caption, such as **Please type your password**.
9. For Input Echo Character, choose **Shadowed**. This hides the characters the user types in the installer panel.
10. In the Label text box, type **Password**.
11. Click **OK**.
12. Click **Add Textfield** to add a new text field for the password confirmation.
13. In the Configure Textfield dialog box, type a **Caption**, such as **Please retype your password**.
14. For Input Echo Character, choose **Shadowed**. This hides the characters the user types in the UI.
15. In the label text box, type **Confirmation**.
16. Click **OK**.
17. Click **Preview Panel** to show a preview with your panel's current controls.



Tip • For the text fields controls you add to this panel, the **InstallAnywhere** variable that records the user input appears in the **Results Variable** text box on the **Configure Textfield** dialog box.

Using Command-Line Arguments with Installers and Uninstallers

InstallAnywhere installers and uninstallers can be run with command-line arguments that can dictate, alter, or override the function of the installer. InstallAnywhere includes installer command line arguments that perform the following tasks:

- [Setting the Interface Mode](#)
- [Setting the Installer Locale](#)
- [Generating a Response File](#)
- [Using a Response File or installer.properties File](#)
- [Passing Custom Variables](#)
- [Showing Installer Help](#)

Setting the Interface Mode

To set the interface mode from the command line, perform the following steps:



Task: *To set the interface mode from the command line:*

At the command line, enter <installer_name> -i <mode>

For example, to use console mode with an installer that includes support for console mode:

```
install.exe -i console
```

Setting the Installer Locale

To set the installer locale from the command line, perform the following steps:



Task: *To set the installer locale from the command line:*

At the command line, enter <installer_name> -l <language_code>[_OPTIONAL_COUNTRY_CODE]

For example, to run the installer with the Simplified Chinese locale:

```
install.exe -l zh_CN
```

Generating a Response File

To generate a response file from the command line, perform the following steps:



Task: *To generate a response file from the command line:*

At the command line, enter <installer_name> -r "path_and_file_name"

For example, to create a response file, you could enter the following at the command line:

```
install.exe -r "/Users/James/myresponse.properties"
```



Note • If you do not enter a path and file name for the response file, the file will be named *installer.properties* or *[installername].properties* and it will be created in the same directory as the installer.



Tip • Be aware that non-standard properties file names require a more complicated call to be used later. This is because any properties file or response file that uses either the default name (*installer.properties*) or the installer name (*[installer_name].properties*) can be automatically used by an installer when the installer and the properties file are in the same directory. When you use a non-standard name, you must reference the properties file (using a *-f* argument), specifically, when you run the installer.

Using a Response File or installer.properties File

To use a response file or `installer.properties` file from the command line, perform the following steps:



Task: *To use a response file or `installer.properties` file from the command line:*

At the command line, enter `<installer_name> -f "path_and_file_name"`

For example, to use property settings from a file named `installer.properties` stored in the All Users\Desktop directory, enter the following:

```
install.exe -f "C:\Documents and Settings\All Users\Desktop\installer.properties"
```

Passing Custom Variables

To pass custom variables from the command line, perform the following steps:



Task: *To pass custom variables from the command line:*

At the command line, enter `<installer_name> -D<myvar=myvalue>`

For example, to pass a new value for install directory (\$USER_INSTALL_DIR\$):

```
install.exe -D$USER_INSTALL_DIR$="/Users/jane/Applications Folder"
```

Showing Installer Help

To show installer help from the command line, perform the following steps:



Task: *To show installer help from the command line:*

At the command line, enter `<installer_name> -?`

Setting Project Version at Build Time

There are several methods you can use to change the version of an InstallAnywhere project at build time:

- Using `productVersion` Parameter in `build.exe` Command Line
- Using `ProjectVersion` Attributes in `buildproperties.xml` File
- Using `product.version` Attributes in `buildproperties.properties` File
- Using `ProjectVersion` Attributes in `iaant.jar` File

Using productVersion Parameter in build.exe Command Line

You can use the productVersion parameter in the command line of build.exe to change the version of an InstallAnywhere project at build time. The following example changes the version to 2.1.3.24:

```
build.exe MyProduct.iap_xml productVersion=2.1.3.24
```



Note • Versions are conventionally represented in the following format: [Major].[Minor].[Revision].[Subrevision], such as: 2.1.3.24.

Using ProjectVersion Attributes in buildproperties.xml File

In the buildproperties.xml file, you can specify the following attributes in the <build> tag to customize the project version at build time:

```
<build
    ProjectVersionMajor="2"
    ProjectVersionMinor="3"
    ProjectVersionRevision="1"
    ProjectVersionSubrevision="24">
    ...
    ...
</build>
```

Using product.version Attributes in buildproperties.properties File

Using the buildproperties.properties file, you can specify the product.version attributes to customize the project version at build time:

```
product.version.major=2
product.version.minor=3
product.version.revision=1
product.version.subrevision=24
```

Using ProjectVersion Attributes in iaant.jar File

You can use the ProjectVersion attributes in the iaant.jar file to customize the project version at build time:

```
<taskdef name="buildinstaller" reverseloader="true"
    classname="com.zerog.ia.integration.ant.InstallAnywhereAntTask">
    <classpath>
        <fileset dir="${env.IA_HOME}/resource/build">
            <include name="iaant.jar"/>
        </fileset>
    </classpath>
</taskdef>
<buildinstaller IAProjectFile="BasicProject.iap_xml"
    IALocation="${env.IA_HOME}"
    ProjectVersionMajor="2"
    ProjectVersionMinor="3"
    ProjectVersionRevision="1"
    ProjectVersionSubrevision="24" />
```

Checking Disk Space During Installation

You can create an installer that performs a disk space check at various points in the installation life cycle, and you can customize how that disk space is displayed.

To display disk space on the Pre-Installation Summary panel of the installer, select the **Disk Space Information** in setting on the customizer of the Pre-Install Summary panel, and select the magnitude you wish to use (**Bytes**, **KiloBytes**, **MegaBytes**, or **GigaBytes**) from the list.

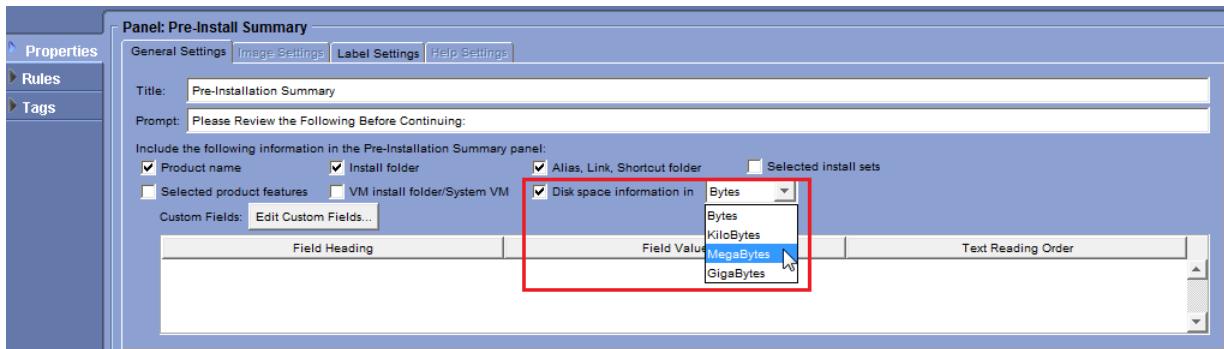


Figure 6-1: Disk Space Information Option on the Pre-Install Summary Panel Customizer

In the following example, **MegaBytes** was selected from the list:

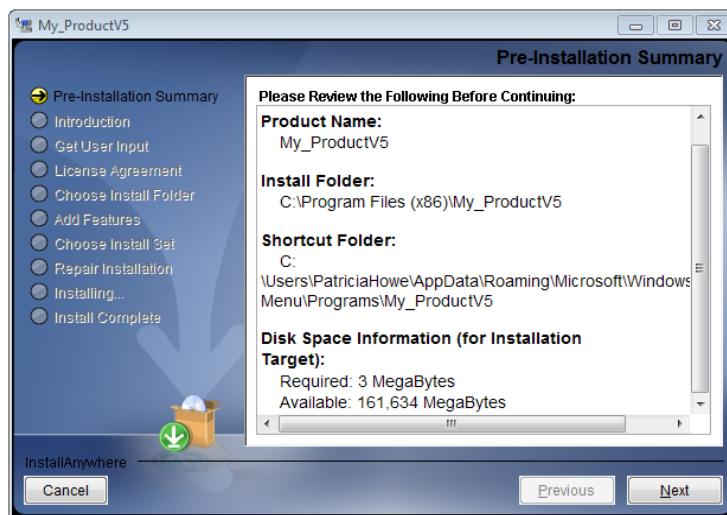


Figure 6-2: Pre-Installation Summary Panel Displaying Disk Space Information

Chapter 6: Adding Advanced Procedures to Installers

Checking Disk Space During Installation

If you want to use more than one magnitude to display disk space information (such as to displaying Free Disk Space in GBs while showing Required Disk Space in MBs), you can use the following variables.

Table 6-12 • Disk Space Variables

Type	Variables
Free Disk Space	\$FREE_DISK_SPACE_BYTES\$ \$FREE_DISK_SPACE_KILOBYTES\$ \$FREE_DISK_SPACE_MEGABYTES\$ \$FREE_DISK_SPACE_GIGABYTES\$
Required Disk Space	\$REQUIRED_DISK_SPACE_BYTES\$ \$REQUIRED_DISK_SPACE_KILOBYTES\$ \$REQUIRED_DISK_SPACE_MEGABYTES\$ \$REQUIRED_DISK_SPACE_GIGABYTES\$



Note • Insufficient space messages will always be displayed in megabytes.

Considerations Regarding Disk Space Checks

When configuring your installation project to perform disk space checks, keep the following in mind:

- The **Disk Space Check** action is always the first action executed as part of Pre-Install phase.
- Disk space is governed by the user installation directory (\$USER_INSTALL_DIR\$ and **Choose Folder** action) and the chosen install set (\$CHOSEN_INSTALL_SET\$ and **Choose Install Set** action).
- The **Disk Space Check** action will be also be run on demand when there is change in installation folder or install sets using either actions or variables.

Deploying Your Product with FlexNet Connect

InstallAnywhere includes an **Enable Update Notifications** action. This action deploys the FlexNet Connect Java agent along with your product. The Java agent periodically checks for notifications about your product and allows your users to view messages as well as to download and install any updates it finds. (By default, the Enable Update Notifications action also creates a launcher for the Java agent so your end users have the option of checking for updates manually.)

The following sections describe how to integrate the FlexNet Connect client with your InstallAnywhere installer and the product it installs:

Table 6-13 • Deploying Your Product With FlexNet Connect

Topic	Description
Adding an Enable Update Notifications Action	Explains how to add the Enable Update Notifications action to your installers.
Customizing an Enable Update Notifications Action	Describes the process of defining update notification settings in the Enable Update Notifications customizer.
Connecting Your Product with the Publisher Site	Shows how to use the Product ID from your install project to correlate the product deployed by your installer with the product on the Publisher site. Without connecting the Publisher-side product with your installer's product, notification delivery will fail.
Removing an Enable Update Notifications Action	Explains how to remove an Enable Update Notifications action from your installers.

Adding an Enable Update Notifications Action

You can integrate with FlexNet Connect and enable update notifications by adding an action to the **Install** task. You can order the FlexNet Connect integration amongst other actions.



Task: *To add an Enable Update Notifications action*

1. On the Advanced Designer, click **Install**.
2. Click **Add Action**. The **Choose an Action** dialog box appears.
3. Click **Enable Update Notifications** then click **Add**. An Enable Update Notifications action appears in the Visual Tree and the **Enable Update Notifications** customizer appears.



Note • This action can be reassigned to Features and Components like all other actions. Use the **Assign to** list and the navigational arrows to reassign actions and change their order within the Visual Tree.

4. Continue with [Customizing an Enable Update Notifications Action](#) to specify or change properties associated with this action.

Customizing an Enable Update Notifications Action

The **Enable Update Notifications** customizer contains the following tabs:

- [General Settings Tab](#)
- [Advanced Settings Tab](#)



Important • You must register your product with FlexNet Connect before you can successfully integrate it with your installation project. If you have not already registered your product, click **Click here** on the **General Settings** tab. A FlexNet Connect window appears. Enter your username and password and click **Sign In**. Register your product using the screen prompts. Click the quick help icons for more information.

General Settings Tab

The steps in the task, below, explain how to customize the General Settings tab. For details about the controls on this tab, see [Enable Update Notifications](#).



Task: *To customize the General Settings tab*

1. In the Install task, click the Enable Update Notifications action you want to customize. The **Enable Update Notifications** customizer appears.
2. Click the **General Settings** tab.
3. At the **Path** label, click the drop-down list to specify the location you want the FlexNet Connect files to be installed. Notice that the location of the Enable Update Notifications action in the Visual Tree changes to reflect the new destination.



Note • *The text box associated with the FlexNet Connect file destination is read-only; however, you can place the Enable Update Notifications action in any folder in the Visual Tree and that change is automatically reflected in the customizer.*

4. Click the **Create LaunchAnywhere to execute update check** to enable or disable the creation of a launcher. By default, the **Enable Update Notifications** customizer is configured to create a LaunchAnywhere executable. If you do not want to include a LaunchAnywhere executable for this action, clear the **Create LaunchAnywhere to execute update check** check box. (This choice disables all subsequent settings on the **General Settings** tab.)
5. In the **Launcher Name** box, type a description for this launcher.
6. If your project builds installers for Windows, click a **Windows Execution Level** option: **As Invoker, Highest Available**, or **Administrator**.
7. Click **Create Alias, Link, Shortcut** to generate a link, alias, or shortcut for the update-check launcher.
8. Click **Change** to select a custom icon for this launcher. To revert to the default LaunchAnywhere icon, click **Default**.

Advanced Settings Tab

The steps in the task, below, explain how to customize the settings on the Advanced Settings tab. For details about the controls on this tab, see [Enable Update Notifications](#).



Task: *To customize the Advanced Settings tab*

1. Click the **Enable Update Notifications** action you want to customize on the Virtual Tree. The Enable Update Notifications customizer appears.
2. Click the **Advanced Settings** tab.

Chapter 6: Adding Advanced Procedures to Installers

Deploying Your Product with FlexNet Connect

3. By default, FlexNet Connect messages appear in the same language as the installer runtime. To select another language, click the **Custom** radio button and click the desired language on the list.



Important • *The following languages supported in InstallAnywhere are not available for FlexNet Connect: Czech, Catalan, Icelandic. If you do not specify a custom language in this case, the FlexNet Connect language defaults to English.*

4. By default, the version number InstallAnywhere uses to register your product with FlexNet Connect is the product version you provide in the **Project > Description** subtask. To register your product under a different version number, click the **Custom** radio button and specify the alternative version in the box.
5. Specify the amount of days between the times that FlexNet Connect checks for notifications for your product in the **Update check interval (days)** box.



Note • *This check-interval setting applies only to installation authors using ShowNotification API calls (or the deprecated AppUpdate and AutoUpdate API calls) in the FlexNet Connect agent.*

6. By default, InstallAnywhere uses the server hosted by Flexera Software to communicate with FlexNet Connect. To designate another server, click the **Self-hosted URL** radio button and enter the URL in the box.
7. By default, InstallAnywhere uses the FlexNet Connect library deployed by the InstallAnywhere installer. If you want to specify another version of the FlexNet Connect library, click the **Custom location** radio button and either type the path and filename in the box or click **Choose File** to browse to the file.

Connecting Your Product with the Publisher Site

Before your installed products can receive notifications, you must make sure the product ID of the product in your installer matches the product code of the product on the Publisher site.



Task:

To verify that your product ID matches the product code:

1. In the InstallAnywhere Advanced Designer, click **Project > Description**.
2. On the Description task, make a note of the **Product ID** value.
3. In a Web browser, open the FlexNet Connect Publisher site.
4. Log in to the Publisher site (FlexNet Connect username and password required), and open the **Products** view.

5. In the Products view, select the **Product** your installers deploy; then click **View/Edit**. This opens the Edit Product page.



Note • If your installer project's product does not exist as a product on the Publisher site, you must first enter the product there. Remember to use the product ID from your installer as the product code.

6. On the Edit Product page, locate the product code and compare it to the product ID from the **Project > Description** task of your installer project.



Note • The product code appears right after the **What is the Product Code for your product?** prompt on the **Edit Product** page.

These values must be an exact (albeit *not* case-sensitive) match.



Tip • It is also critically important that the version number in your installer project exactly matches the version number of the product on the Publisher site. If the version numbers do not match, or are only a partial match, the check for notifications fails and the agent returns an error.

Removing an Enable Update Notifications Action

You can also remove unwanted Enable Update Notifications actions in the **Install** task.



Task: **To remove an Enable Update Notifications action**

1. On the Advanced Designer, click **Install**.
2. On the Visual Tree, click the **Enable Update Notifications** action you want to remove.
3. Click **Remove**. The Enable Update Notifications action is deleted from the Visual Tree.

Localizing Projects and Installers

InstallAnywhere's Enterprise edition allows developers to build installers for up to 31 different languages. The Standard edition enables developers to build for up to 9 languages. The following topics provide instructions for specific localization tasks.

Table 6-14 • Localizing Projects and Installers

Topic	Description
Generating Multi-Language Installers	Describes the very basic steps to designate the locales your installer will support.
Localizing Resources	Describes the process for localizing resources like License Agreements, side panels, billboards, and custom icons.
Localizing Custom Installer Labels	Describes how to localize custom installer labels that are not automatically localized.
Localizing the Splash Screen	Describes how to localize the title, image, and confirmation button of the splash screen.
Localizing the Get User Input Panels	Describes how to internationalize elements of the Get User Input panels.
Adding an External Resource Bundle	Explains how to add an external resource bundle to your project.
Referencing an External Resource Key	Shows how to reference the keys from your external resource bundle.



Note • For information on localizing custom code, see [Internationalizing Custom Code](#).

Generating Multi-Language Installers

InstallAnywhere can generate localized installers when you select the locales you want to support. You specify locale settings for each Build Configuration on the **Locales** subtab of the **Build** task's **Build Configurations** tab. (Additional localization may be required for resources, customized panels or consoles, and custom code.)



Task: *To generate multi-language installers*

1. Open the **Build Configurations** tab of the **Build** task.
2. Select the Build Configuration you want to edit from the **Select Build Configuration** list.
3. Open the **Locales** subtab.
4. Use the check boxes to select languages.
5. Save the project.

Localizing Resources

Resources (such as License Agreements, side panels, billboards, and custom icons) can be localized for specific countries. Actions such as the License Agreement Panel and LaunchAnywhere serialize the paths and filenames to their resources as well as their dynamic strings to the locale files. You can then change these paths and the filenames.



Task: *To localize the License Agreement*

1. Make sure to include every resource in the Install task. Installers will not have access to resources not specified in this task.
2. Find the line in the locale file that contains the text `LicenseAgr.#.FileName`.
3. Specify the filename of the file that contains the localized license agreement (for instance, `License_fr.html`). Do not type the fully qualified absolute pathname to the file—just the filename itself.
4. Find the line that contains the text `LicenseAgr.#.Path`.
5. Specify the path name to the file that contains the localized license agreement (on the local file system).

Localizing Custom Installer Labels

Custom labels that match the installer panels are not automatically localized.



Task: *To match labels for localization*

1. Build the installer.
2. Locate the installer's Locale directory.
3. Open the custom_en file in WordPad or another text editor.
4. Search for the `Installer.1.installLabelsAsCommaSeparatedString` variable. This variable should contain the added installer labels.
5. Copy and paste this variable into the other locale files.
6. Finally, provide translations for these labels in their respective files.

Localizing the Splash Screen



Edition • The ability to localize the splash screen is available only in InstallAnywhere Enterprise Edition.

The startup splash screen shows a splash screen title, image, and confirmation button. All of these elements are customizable by locale.

When you localize these splash screen elements, they respond to the locale of the target machine and to the locale setting command-line option: -l <language code>.

- Localizing the Splash Screen Title
- Localizing the Splash Screen Image
- Localizing the Splash Screen Confirmation Button



Note • See [Using Command-Line Arguments with Installers and Uninstallers](#).



Tip • You can use a variable for these elements, but be aware that the only variables that are automatically resolved in the splash screen are \$INSTALLER_TITLE\$ and \$PRODUCT_NAME\$. The value of any other variable used here must either be passed to the installer at the command line or be included in the `installer.properties` file for the installer. (It is not possible to localize the splash screen with references to keys in an external resource bundle.)

Localizing the Splash Screen Title

To localize the splash screen title, perform the following steps:



Task: *To localize the splash screen title:*

1. Locate your project's `locales` directory. This directory is in the same location as your project file and named to match your project name:

`<project_name>locales_<build_configuration_name>`

such as:

`My_Productlocales_MacOSConfiguration`



Important • Prior to InstallAnywhere 2010, the `locales` directory was named `<project_name>locales`. Therefore, if you are migrating from InstallAnywhere 2009 to InstallAnywhere 2011, you need to rename the `<project_name>locales` directory to `<project_name>locales_<build_configuration_name>` and then rebuild the project.

2. Open each non-English locale file in a text editor, and find the `splashScreenGUITitle` element. This element typically uses an `Installer.#.splashScreenGUITitle` key, where # represents the reference ID for the element.
3. Provide the translated title text as the value for the `Installer.#.splashScreenGUITitle` key.
4. Save your changes.

Localizing the Splash Screen Image

To localize the splash screen image, perform the following steps:



Task: *To localize the splash screen image:*

1. In the Install task, add all localized splash screen image files to the **DO NOT INSTALL** magic folder. This ensures the image resources for the localized splash screens are available to your installer but not installed to the target system.
2. Locate your project's `locales` directory. This directory is in the same location as your project file and named to match your project name:

`<project_name>locales_<build_configuration_name>`

such as:

`My_Productlocales_MacOSConfiguration`



Important • Prior to InstallAnywhere 2010, the `locales` directory was named `<project_name>locales`. Therefore, if you are migrating from InstallAnywhere 2009 to InstallAnywhere 2011, you need to rename the `<project_name>locales` directory to `<project_name>locales_<build_configuration_name>` and then rebuild the project.

3. Open each non-English locale file in a text editor and find the `splashScreenGUImageName` and `splashScreenImagePath` elements. These elements appear in the locale files as `Installer.#.splashScreenGUImageName` and `Installer.#.splashScreenImagePath`, where # represents the reference ID for the element.
4. Enter the correct locale-specific file name and path (if necessary) values for the `Installer.#.splashScreenGUImageName` and `Installer.#.splashScreenImagePath` keys, respectively.
5. Save your changes.

Localizing the Splash Screen Confirmation Button

To localize the splash screen confirmation button, perform the following steps:

**Task:****To localize the splash screen confirmation button:**

1. Locate your project's `locales` directory. This directory is in the same location as your project file and named to match your project name:

`<project_name>locales_<build_configuration_name>`
such as:
`My_Productlocales_MacOSConfiguration`



Important • Prior to InstallAnywhere 2010, the `locales` directory was named `<project_name>locales`. Therefore, if you are migrating from InstallAnywhere 2009 to InstallAnywhere 2011, you need to rename the `<project_name>locales` directory to `<project_name>locales_<build_configuration_name>` and then rebuild the project.

2. Open each non-English locale file in a text editor, and find the `splashScreenGUICConfirm` element. This element typically uses an `Installer.#.splashScreenGUICConfirm` key, where # represents the reference ID for the element.
3. Provide the translated confirmation button text as the value for the `Installer.#.splashScreenGUICConfirm` key.
4. Save your changes.



Tip • You can also localize related console-installer elements on the **Choose Locale** console using the locale elements `splashScreenConsoleTitle` and `splashScreenConsolePrompt`.

Localizing the Get User Input Panels

You can easily internationalize choices in the **Get User Input** panel by editing a project's locale files. Each choice or option in the Get User Input panel is localizable whether it is a label, text field, option button, or check box. Here are some examples using English and French:

Get User Input - Simple Panel Example



Task: *To localize a Get User Input - Simple panel:*

1. Create your project as you normally would. In this example, we have added a Get User Input - Simple panel that asks the question: "Are you ready?" This panel offers the user option buttons with Yes and No as possible choices.
2. Build for all intended platforms and locales.
3. Navigate to the <project_name>locales_<build_configuration_name> directory.
4. As we're customizing a French installer for this example, use a simple text editor (notepad.exe,TextEdit, VI, or emacs for example) to open the custom_fr and custom_en files.
5. In the custom_en file, search for the text you want to customize (the text string used in your **Get User Input - Simple** panel, in this case—"Are you ready?") It will look something like this (though the ID may be different):

```
# CreateDialog.c586fa408eaf.prompt=Are you ready?  
CreateDialog.c586fa408eaf.prompt=Are you ready?  
# CreateDialog.c586fa408eaf.title=Get User Input  
CreateDialog.c586fa408eaf.title=Get User Input
```

If you want to change the Yes/No, just do a search for Yes or No in the custom_en. You find something like this:

```
# EntryAtom.c8ca91558eaf.label=Yes  
# EntryAtom.c58723868eaf.label=No
```

6. Find the matching entries in the custom_fr file.
7. In the custom_fr file, make changes to the matching entries to reflect the customizations for this locale. For example:

```
# CreateDialog.c586fa408eaf.prompt=Are you ready?  
CreateDialog.c586fa408eaf.prompt= tes Vous prít?  
  
# CreateDialog.c586fa408eaf.title=Get User Input  
CreateDialog.c586fa408eaf.title=Obligez L'Utilisateur à entrer  
  
# CreateDialog.ef23914cb8fd.ValidInputsAsCommaSeparatedString=yes,no  
CreateDialog.ef23914cb8fd.ValidInputsAsCommaSeparatedString=oui,non  
  
# EntryAtom.c8ca91558eaf.label=Yes  
EntryAtom.c8ca91558eaf.label=Oui  
  
# EntryAtom.c58723868eaf.label=No  
EntryAtom.c58723868eaf.label=Non
```

8. Rebuild your project. The proper entries should appear in the proper locale.

Get User Input - Advanced Example

The process is similar for the **Get User Input - Advanced** panels. Here is the same example, but using the Get User Input - Advanced panel.

**Task:** *To localize a Get User Input - Advanced panel:*

1. Create your project as you normally would. In this example, we've added a User Input Panel that asks the question: "Are you ready?" This panel offers option buttons with Yes and No as possible choices.
2. Build for all intended platforms and locales.
3. Navigate to the <project_name>locales_<build_configuration_name> directory.
4. As we're customizing a French installer for this example, use a simple text editor (notepad.exe,TextEdit, VI, or emacs for example) to open the custom_fr and custom_en files.
5. In the custom_en file, search for the text you want to customize (the text string used in your User Input Panel, in this case - "Are you ready?") It will look something like this (though the ID may be different):

```
# GetUserInput.c8e4659f8f03.title=Get User Input
 GetUserInput.c8e4659f8f03.title=Get User Input

# GUIGroupData.c8e471658f03.caption=Are you ready?
 GUIGroupData.c8e471658f03.caption=Are you ready?
```

If you want to change the Yes/No, just do a search for Yes or No in the custom_en. You find something like this:

```
# GUIComponentData.c8e477e48f04.label=Yes
 GUIComponentData.c8e477e48f04.label=Yes

# GUIComponentData.c8e4c9558f04.label=No
 GUIComponentData.c8e4c9558f04.label=No
```

6. Find the matching entries in the custom_fr file.
7. In the custom_fr file, make changes to the matching entries to reflect the customizations for this locale. For example:

```
# GetUserInput.c8e4659f8f03.title=Get User Input
 GetUserInput.c8e4659f8f03.title=Obligez L'Utilisateur à entrer

# GUIGroupData.c8e471658f03.caption=Are you ready?
 GUIGroupData.c8e471658f03.caption=  Vous prít?

# GUIComponentData.c8e477e48f04.label=Yes
 GUIComponentData.c8e477e48f04.label=Oui

# GUIComponentData.c8e4c9558f04.label=No
 GUIComponentData.c8e4c9558f04.label=Non
```

8. Rebuild your project. The proper entries should appear in the proper locale.

Adding an External Resource Bundle



Edition • External resource bundles are available only in InstallAnywhere Enterprise edition.

External resource bundles contain custom locale files (one per locale) you can use to provide additional localized strings in your installer or replace InstallAnywhere's built-in dynamic locale files. Each custom locale properties file includes a set of key/value pairs that provide localized values for installer resources.

When you add an external resource bundle, your installers use the custom locale properties files from that bundle to resolve the value of keys referenced in the installer, based upon the value of \$INSTALLER_LOCALE\$.



Task: To add an external resource bundle

1. In the Advanced Designer, open the **Project > Locales** task.
2. Double-click the **Bundle Name** column of the **External Resource Bundle Settings** table.
3. Enter a **Bundle Name** for the external resource bundle you want to use.
4. Double-click the **Resource Bundle Path** column. The **Choose a File** dialog box opens.
5. Locate the directory in which your custom locale properties file are stored and select one of the locale properties files.
6. Click **Select**.



Note • Verify that the external resource bundle contains a properties file for each locale you plan to support. Installers built for locales without a properties file will default to English values for each localized resource that references the external resource bundle.

Referencing an External Resource Key



Edition • External resource bundles are available only in InstallAnywhere Enterprise edition.

The keys you create in the locale properties files of your external resource bundle can be referenced in your installers using the following syntax.

`$L{<bundle_name>.<key>}`

- `<bundle_name>` is the name you provide in the **External Resource Bundle Settings** table on the **Project > Locales** task.
- `<key>` is the key for the localized value you want to show in your installer.

External resource bundle keys function just like InstallAnywhere variables—except they always acquire their value from the locale files in your external resource bundle (according to the value of `$INSTALLER_LOCALE$`). Hence, if the installer locale is FR (French), the value of a key reference is set by that key's value in the French locale properties file within your external resource bundle. (The installer locale is commonly set by the locale of the target system, the default installer locale, or the locale the user selects on the splash screen.)



Tip • Many of the most common panels have dynamic text elements that include default values. For example, the Introduction panel provides default text for the panel title and message. InstallAnywhere provides localizations in all supported locales for these default values. If you enter a reference to an external resource key in place of the default text, InstallAnywhere uses the locale files in your external resource bundle instead of InstallAnywhere's built-in dynamic locale files.

Packaging and Executing Custom Code

Custom code can extend the functionality of your installer. Follow the steps below to ensure your custom code compiles and executes correctly for use with InstallAnywhere.



Task:

To create and use custom code in an installer

1. Add `IAClasses.zip` to the CLASSPATH. The Java compiler will need to reference the classes in this file when compiling the code. `IAClasses.zip` is located in the root installation directory of InstallAnywhere.
2. Create the action, panel, console or rule. A good starting point is to use the Java source file templates found in the `CustomCode/Templates` folder inside the InstallAnywhere installation directory.
3. Compile the source files.
4. Decide which additional files and resources will be needed. For example, images, text files, or other resources may be required.
5. Create a Java Archive (JAR) file that contains the compiled class files and resource files.



Note • For information on using signed JARs as dependencies, see [Support for Signed JARs as Dependencies](#).



Note • Make sure that the selected archive has full path information in it. (Each file in the archive should be stored with its proper package path.) Without this JAR, Java will not be able to properly find the packaged class files.

6. Add an action or rule that executes custom code (such as **Execute Custom Code**).
7. Choose the JAR file created in the previous step.
8. Type the fully qualified package and class name, such as: `com.acme.MyAction`.
9. Set options and add necessary dependencies.



Note • Custom code that uses the `InstallShieldUniversalRegistry` service needs to add `hsqldb.jar` as a dependency. Click **Add jar or zip**, locate `hsqldb.jar` in `<InstallAnywhere>\resource\dbclients`, and click **Open**.

10. Test and debug the action or panel.



Note • A frequent source of trouble with custom code is incompatibility between the JDK with which you compile your code and the JVM you bundle with your installer. Ensure your Java compiler is compatible with your installers' bundled JVMs.

Since InstallAnywhere cannot be run from within an integrated development environment, the best ways to debug the custom code are to use the **Output Debug Information** action or `System.out.println()` statements to print debug output to the console during testing.



Note • It is critical that the Custom Code JAR does not include the classes and resources from `IAClasses.zip` as this will seriously affect the behavior of the installer and uninstaller.

Support for Signed JARs as Dependencies

Starting with InstallAnywhere 2011, InstallAnywhere now supports adding signed JARs as dependencies to Custom Code Actions, Custom Code Panels, Custom Code Consoles, and Custom Rules. Also, a JAR or ZIP file that contains custom code can now be signed.

In previous versions of InstallAnywhere, although you could build a project that contained signed JARs, the installer that was generated would not launch. InstallAnywhere would extract all entries of the signed JAR, including the manifests, and would add them to the installer ZIP files (such as `execute.zip`). But when the installer ZIP files were added to the classpath of the installer, the installer would fail to launch because Java Security Exceptions would be thrown when initializing the JAR.

Starting with InstallAnywhere 2011, in the case of a signed JAR, InstallAnywhere will not unpack the JAR and the JAR entry will be left intact inside the `execute.zip` file during build time. Therefore, during runtime, the signed JARs are extracted to temporary directories and are added to the InstallAnywhere classpath, making them ready to be consumed for the custom codes or any relevant actions requiring the contents of this signed JAR.

Signed JARs can now be added as dependencies in the following scenarios:

- [Execute Custom Code Action, Panel, or Console](#)
- [Evaluate Custom Rule](#)
- [Maintenance Mode, Instance Management, Plug-Ins](#)

Execute Custom Code Action, Panel, or Console

When configuring an Execute Custom Code action, a Custom Code panel, or a Custom Code console, specify the custom code archive location that contains all the classes required for execution, including the fully qualified class name to execute.

In addition, specify the location of the signed JAR archive files which the custom code is dependant upon. When the installer is built, the signed jar along with the signing information will be kept intact.

During runtime, the dependant signed jars will be placed on the IA classpath to be consumed while the relevant action execution happens.

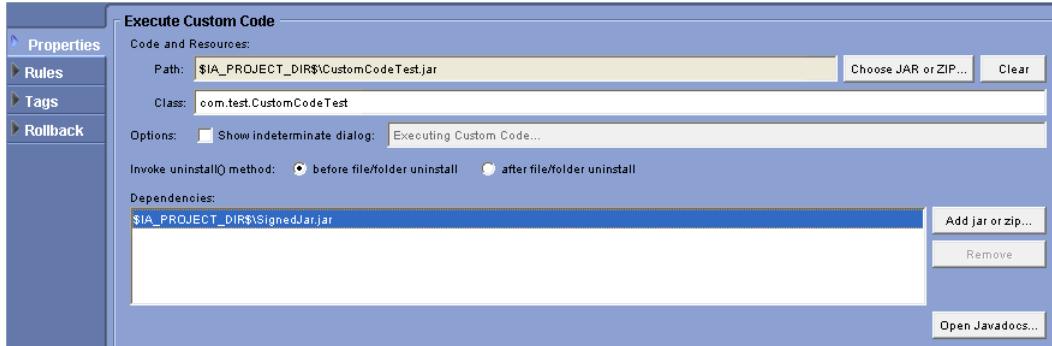


Figure 6-3: Selecting a Signed JAR as a Dependency on Execute Custom Code Customizer

Evaluate Custom Rule

When adding an **Evaluate Custom Rule**, you can click **Configure Dependencies** and select a signed JAR on the Custom Rules Dependencies dialog box.

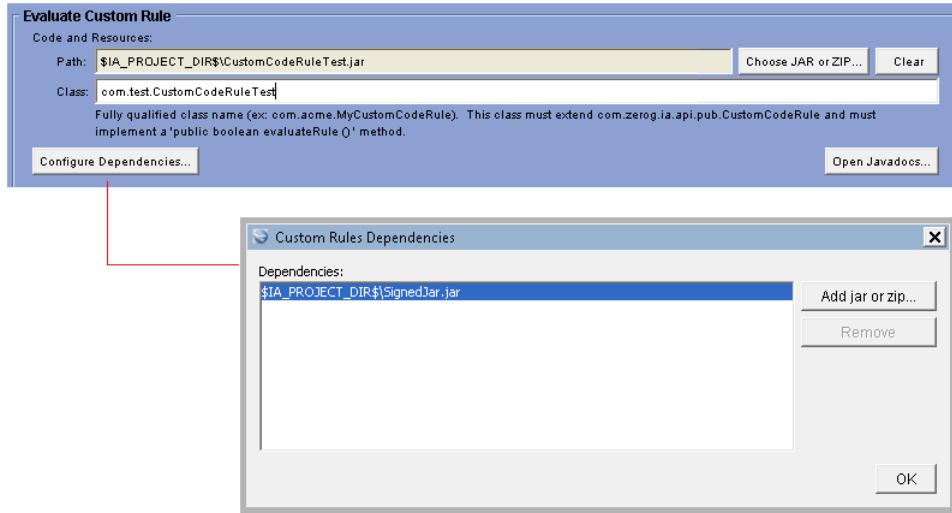


Figure 6-4: Selecting a Signed JAR as a Dependency on an Evaluate Custom Rule

Maintenance Mode, Instance Management, Plug-Ins

You can also include signed JARs for some Maintenance Mode options (such as Add/Repair) and also for Instance Management. Also, custom codes that are packaged in a signed JAR can be used as Plug-Ins.

Calling InstallShield MultiPlatform APIs in InstallAnywhere

InstallAnywhere includes a set of InstallShield MultiPlatform APIs (Services), which you can import into a custom code action. The following service interfaces are available:

Table 6-15 • InstallShield MultiPlatform Service Interfaces

Interface	Description
SecurityService	Provides the ability to work with users and user groups on the target system.
SystemUtilService	Performs system-level operations such as setting up environment variables, rebooting the system, startup commands, and OS properties.
FileService	Provides the ability to manipulate and query files, directories, and partitions at runtime.

Chapter 6: Adding Advanced Procedures to Installers

Calling InstallShield MultiPlatform APIs in InstallAnywhere

Table 6-15 • InstallShield MultiPlatform Service Interfaces

Interface	Description
Win32Service	Provides a variety of methods for interacting with Win32 capabilities including Windows NT Services, and Windows specific file system APIs.
Win32RegistryService	Provides access to the Win32 Registry for reading and writing registry data.

To call InstallShield MultiPlatform APIs in InstallAnywhere, perform the following steps:



Task: *To call InstallShield MultiPlatform APIs in InstallAnywhere:*

1. To access to these services, use the following syntax:

```
InstallerProxy.getService(<ClassName>.class)
```

For example, to get access to the InstallShield MultiPlatform FileService, use the following code:

```
FileService fservice = (FileService) ip.getService(FileService.class);  
//where ip is the InstallerProxy Object
```

2. When compiling code which uses these services, all JAR files in the <IAhome>/resource/services directory and the <IAhome>/resource/services/ppk directory should be in the compilers classpath.
3. In addition, you need to select the **Add service support for custom code** option under **Classpath Settings** on the **General Settings** tab of the **Project > JVM Settings** subtask in the Advanced Designer interface. This ensures that these services are built into the installer and made available at runtime.

For additional information on Custom Code, please see the following resources:

Table 6-16 • Additional Information on Custom Code

Subject	Location in InstallAnywhere's Installation Directory
Java Docs	/javadoc/index.html
Templates	/CustomCode/Templates
Samples	/CustomCode/Samples

Packaging Custom Code as a Plug-in

In order to make Custom Code available as a plug-in action in InstallAnywhere, you must first package it properly.



Task:

To package a custom code as a plug-in

1. Package the custom code and all of its resources in a JAR (just like for regular custom code).
2. Create a properties file called `customCode.properties`. In this properties file will be specified all of the information InstallAnywhere needs to integrate the plug-in with the Advanced Designer. Place the properties file in the JAR with no stored path information (at the root level of the JAR).

The properties file *must* contain the following properties:

- **plugin.main.class=<classname>** The class that implements the proper member of the InstallAnywhere API for a custom code action, panel or console (such as a class that implements `com.zerog.ia.api.pub.CustomCodeAction`)
- **plugin.name=<plug-in name>** The name the plug-in will be displayed as in the **Plug-Ins** tab of the InstallAnywhere Advanced Designer's **Choose an Action** dialog box.
- **plugin.type=<action | panel | console>** A property that helps InstallAnywhere determine when the plug-in can be used and which icon to use to represent it in the Advanced Designer.

The following properties *may* also be used:

- **property.<propertname>=<propertydefault>** Tells the plug-in to populate the action's customizer with a property named `<propertname>` and set to the default value of `<propertydefault>`.
- **plugin.icon.path=<relative path to .png or .jpg file in JAR>** Sets a custom 32x32 icon for the custom code plug-in in the Advanced Designer
- **plugin.available=<preinstall | install | postinstall | preuninstall | postuninstall>** A comma separated value list that defines the tasks (in the Advanced Designer) in which the plug-in should be available.

The following is an example of a properties file for a plug-in:

```
plugin.main.class=com.zerog.ia.customcode.util.fileutils.ExtractToFile
plugin.name=Extract to File
plugin.type=action
plugin.icon.path=myicon.gif
plugin.available=preinstall,install,postinstall
property.ExtractToFile_Source=path/to/file/in.zip
property.ExtractToFile_Destination=$USER_INSTALL_DIR$$/$myfile.txt
```

3. Additionally, plug-ins can offer help to InstallAnywhere users on how to properly use the plug-in. Help is displayed in HTML, and is launched by the user pressing a button on the plug-ins customizer. The button only appears if a help file is provided in the plug-in. To utilize installer help:
 - a. Create a file called `help.htm`.
 - b. Package it in the plug-in JAR (without stored path information).

4. Place the properly packaged JAR (with the custom code, its resources and the properties file) in <InstallAnywhere>/plugins.
5. Launch InstallAnywhere. The plug-in will now be visible in the InstallAnywhere **Choose an Action** dialog box.

Internationalizing Custom Code

The InstallAnywhere API provides a simple means for localizing custom code actions, panels and consoles. Here's an example of how to localize a java.awt.Label inside a custom code panel. The custom code panel's setup UI method will look something like this:

```
public boolean setupUI(CustomCodePanelProxy ccpp)
{
    Label myLabel = new Label();
    myLabel.setText(ccpp.getValue("MyCustomCodePanel.myLabel"));
}
```

CustomCodePanelProxy, InstallerProxy, CustomCodeConsoleProxy, and UninstallerProxy provide access to the getValue method. This method takes a String as a parameter that represents the key portion of the key/value pair as defined in InstallAnywhere's international resource files. Developers can create any name for the key that they would like, as long as it does not conflict with previously defined keys. They can even use a pre-existing key to obtain a string that has already been translated in InstallAnywhere's resource files.

However, `_ia` must be added to the beginning of each key. For example, if the custom code utilized the key `MyCode.foo.flotsam`, it would need to be added as `_ia.MyCode.foo.flotsam`. This modification must be repeated for all custom code localized strings and for each supported locale.



Note • To accommodate existing custom code, installers first check for keys starting with `_ia`, but then make an additional set of passes, this time ignoring the `_ia` and searching for the remainder of the key name.

Locale key-value pairs are set directly in the project locales files (this is different from previous versions). They are all named `custom_xx` where `xx` is the two-letter locale code for each locale. For instance, the English resource file is `custom_en` and the Japanese resource file is named `custom_ja`. These files are in UTF-8 format. For more information of these files, see [Localization](#) and [Localization Reference](#).

To create/edit locale keys for every installer project, update the static text. To create/edit keys only in the current project, update the dynamic text. The dynamic text is regenerated every time you save your project, so update the files every time the project is changed.

Digitally Signing Installers

You can digitally sign installers on Windows if you have a valid digital certificate. Digital signatures give end users the added security that comes with knowing that an installer came from a trusted source. It also prevents your customer from seeing a “Unknown Publisher” warning when they launch installers on Windows XP Service Pack 2. To digitally sign installers you essentially need 3 files: a .pvk file (a private key), a .spc file (the code signing certificate), and signcode.exe (the code signing tool from Microsoft). You can download signcode.exe Microsoft’s download center. It is included in a package called codesigningx86.exe. Signcode has a GUI mode, but can also run from the command line. The command line can be as simple as:

```
signcode /spc myCert.spc /v mypkey.pvk "install.exe"
```

Signcode has numerous command line parameters. For more information about Signcode, search for signcode.exe in the [MSDN Library](#).

Setting Installation Rollback Options

If an end user cancels an installation before it has completed, or if a fatal error occurs during the installation, the result can be an incomplete, corrupt application. To avoid this problem, you can enable installation rollback by selecting the **Enable Rollback** option on the **Project > Advanced** subtask.

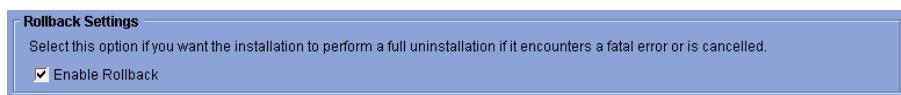


Figure 6-5: Rollback Settings on the Project > Advanced Subtask



Note • The **Enable Rollback** option is selected by default for all new projects.

If you select the **Enable Rollback** option, and the end user cancels an installation or a fatal error occurs, the installer will automatically revert what has been altered or added to the system. The installation log will contain all status messages, including the action that triggered the rollback.

Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback

On the **Rollback** subtab of action customizers in the **Install** task, you can specify rollback options on a file-by-file basis. You can fine tune how the installer treats individual project elements during a rollback and can specify which project elements would trigger a rollback if the installation of that element fails.

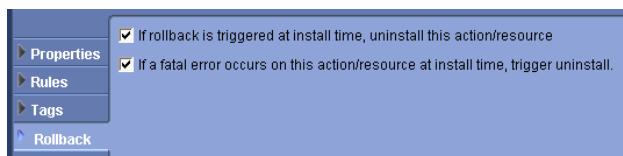


Figure 6-6: Rollback Subtab of Install Task

Chapter 6: Adding Advanced Procedures to Installers

Determining a Successful Installation

If the **Enable Rollback** option is selected on the **Project > Advanced** task, you can select the options on the **Rollback** tab of action customizers in the **Install** task to specify rollback options for the selected project element:

- **If rollback is triggered at install time, uninstall this action/resource**—Select this option if you want the selected project element to be uninstalled if the product installation is cancelled or encounters a fatal error. If this option is not selected and a rollback occurs, the project element will not be uninstalled.
- **If a fatal error occurs on this action/resource at install time, trigger uninstall**—Select this option if you want a rollback to be triggered if the selected project element (execution of the action results in a fatal error) fails to execute properly or to be installed properly. If this option is not selected and the execution of this action results in a fatal error during installation, a rollback will not be triggered.

By default, both options are selected for all project elements.



Note • In previous releases, InstallAnywhere had support for custom code that provided a Rollback Handler that would be called if an end user cancelled his installation. Starting with InstallAnywhere 2010, this new Rollback functionality has been added. Therefore, the Rollback Handlers custom code will be executed only when the **Enable Rollback** option on the **Project > Advanced** subtask is not selected.

Using Trigger Rollback Actions

You can also choose to add a **Trigger Rollback** action to the **Install** task to specifically initiate a rollback if a certain rule or condition is met. For more information, see the [Trigger Rollback](#) action.

Determining a Successful Installation

InstallAnywhere's built-in variable `$INSTALL_SUCCESS$` can be used to receive confirmation that an installation has been successful. There are four possible values for the `$INSTALL_SUCCESS$` variable:

- `SUCCESS`
- `WARNING`
- `NONFATAL_ERROR`
- `FATAL_ERROR`

Similarly, the corresponding variable, `$UNINSTALL_SUCCESS$`, can be used to determine the success of the uninstaller.

Troubleshooting

There are several methods available to debug InstallAnywhere installers. Deciding upon which method to use depends in part on the installer development cycle—during installer development or later if an end user has a problem with the installer.

Most InstallAnywhere installers utilize Flexera Software's LaunchAnywhere technology. Along with many convenient features for end users (double-clickable launchers, native-like user experience) LaunchAnywhere launchers provide a host of built-in debugging features.

Table 6-17 • Troubleshooting Topics

Topic	Description
Debugging During Installer Development	Provides information about debugging steps that setup authors can take during the installer development process and includes links to topics on the use of the Project > Log Settings task (Installer Debug Output), the Output Debug Information action, and the Display Message panel for debugging. This section also includes instructions for manually setting the lax.stderr.redirect and lax.stdout.redirect properties for LaunchAnywhere launchers.
Debugging During Post Development	Aggregates several topics on debugging installers on various supported platforms/build targets. All these options force debug output to the console.
Other Debugging Issues	Includes topics on various platform-specific issues that may be useful for debugging and topics that discuss reviewing debug information.
Exit Codes	Describes exit codes from the build and from InstallAnywhere installers.
Application Server Support	Lists supported application servers, versions, and required JVMs.
Database Server Support	Lists supported database servers and versions.

Debugging During Installer Development

Installer developers can make use of several InstallAnywhere debugging features available from the Advanced Designer. These topics describe debugging techniques that can be employed at design time. They range from reporting robust debug information (far beyond what appears in the install log) to using the Display Message panel to show the value of one or more variables during different steps of the install process.

InstallAnywhere provides two primary types of debug output:

- **Installer Debug Output**—This output provides a host of information relevant to debugging issues with the InstallAnywhere run-time on various platforms. This output can be obtained by setting **Send stderr to** and **Send stdout to** values under **Installer Debug Output** on the **Project > Log Settings** task to direct debug output to the console or to a file or by setting `lax.stderr` and `lax.stdout` properties for a launcher. It can also be obtained, post-development, in a number of platform-dependent methods explained in [Debugging During Post Development](#).
- **Output Debug Information Action Output**—The data this action returns is useful to debugging the logic of your installer. For example, you can use this feedback to determine the values of InstallAnywhere variables, Magic Folders, Java Properties, etc. This can help ascertain the cause of unexpected file deployment/panel flow events. It also allows you to choose what type of debug output you receive.

Topic	Description
Directing Installer Debug Output	Discusses the use of the Project > Log Settings task to direct debug output.
Using the Output Debug Information Action	Describes how to use the Output Debug Information action.
Debugging Using Display Message Panel	Shows how to render variable values in a Display Message panel.
Debugging LaunchAnywhere Launched Executables	Explains how to alter the LAX properties for a launcher to generate debug output.

Directing Installer Debug Output

On the project level, InstallAnywhere allows installer developers to write debug information to the console or to a file.

The **Output Debug Information** action has a similar capability but produces different output and allows developers to select the output data they want to include. Use an **Output Debug Information** action to obtain feedback on the logic of your installer. Use the settings on the **Project > Log Settings** task to debug errors and failures in the InstallAnywhere run time that may be dependent on the environment of the target system.



Task:

To direct debug output to the console

1. Open the **Project > Log Settings** task.
2. Under **Installer Debug Output**, enter `console` in the **Send stderr to** and **Send stdout to** text boxes.

**Task:** **To direct debug output to a file**

1. Open the **Project > Log Settings** task.
2. Under **Installer Debug Output**, enter a file name or a limited set of InstallAnywhere variables in the **Send stderr to** and **Send stdout to** text boxes.



Note • *The default location for debug output sent to a file is the directory in which the installer resides.*

The values you use for **Send stderr to** and **Send stdout to** can include three types of InstallAnywhere variables: environment variables (`$lax.nl.env.*$` and `$lax.nl.env.exact_case.*$`), java system properties (`$prop.*$`), and directory separators variables (`$/``). These variables can be useful when you intend to direct debug output to a custom file location.

Keep the following important points in mind when working with debug output options on the **Project > Log Settings** task:

- Use only the permitted types of InstallAnywhere variables (LAX environment variables, Java properties, and directory separator variables). If you use variables other than the types permitted, they will resolve to empty strings at run time.
- Mac OS X and Pure Java installers do not support LAX environment variables (`$lax.nl.env.*$` and `$lax.nl.env.exact_case.*$`). They do support Java properties (`$prop.*$`) and directory separator variables (`$/``).
- If the location to which the installer attempts to write the stderr and stdout is not writable, the installer creates these debug files in a temporary location. In this case, the installer writes `lax-<random_number>-err.txt` and `lax-<random_number>-out.txt` to the target system's temp directory.
- Because these files are not removed by the uninstaller, consider clearing the **Send stderr to** and **Send stdout to** fields prior to the final build of the product installer. Or leave the output intact to make post-development debugging easier.



Tip • *When your project includes one or more Merge Modules, the stderr and stdout settings of the parent project override the stderr and stdout settings in the Merge Modules.*

Using the Output Debug Information Action

The **Output Debug Information** action allows you to select the output data you want to include and set destination of the output (console or file). The values produced in the debug output depend in part on when the action is performed as part of the install.

The debug information is useful for investigating and resolving problems with your installer. If errors related to rules occur (or, for example an action is not occurring that should occur), use Output Debug Information to verify the values for each of the variables and Java properties used by the InstallAnywhere rules. For errors related to runtime launcher or installer, use the **Installer Debug Output** settings on the **Project > Log Settings** task instead. For more information, see [Directing Installer Debug Output](#).



Note • The **Installer Debug Output** controls on the **Project > Log Settings** task also produce debug information; however, the debug information obtained in that manner is different, in content, from the information provided in the **Output Debug Information** action.



Task: *To use the Output Debug Information action:*

1. In the Advanced Designer, click the **Post-Install** task.
2. Click **Add Action**. This opens the Choose an Action dialog box.
3. On the General tab, click **Output Debug Information**. This adds the action to the Post-Install Action list.
4. In the Output Debug Information customizer, click the output data items you want to include in the debug information output.
5. Set Output Destination options:
 - Check **Output to the console (stderr)** to send the debug output to the console.
 - Check **Output to a file** to send the debug output to the file you specify in the **Path** text box.
 - Customize the Path value as necessary. The value for Path can be a file name, a relative path, an absolute path, with or without InstallAnywhere variables.



Note • Without specifying a location for debug information output to a file (file name only), the debug information is stored at the current value of \$USER_INSTALL_DIR\$.

Debugging Using Display Message Panel

Often, it's desirable to debug some portions of an installation during installer development. One simple feature of InstallAnywhere Enterprise Edition allows developers to add a Display Message panel that can display specific InstallAnywhere variable values.

For example:

1. Add the rule: Install only if `$prop.os.name$=Solaris`
2. Run the installer. Notice that the action to which the rule was assigned does not execute.
3. Add a Display Message panel with the message: The prop.os.name is: `$prop.os.name$`
4. Rerun the installer.

Notice that the message in the Display Message panel shows the value of `prop.os.name` is SunOS, not Solaris. Knowing this, you can reformulate the rule to match the proper name.

Debugging LaunchAnywhere Launched Executables

Because InstallAnywhere installers use LaunchAnywhere executables, the platform-specific procedures are also useful for debugging installed applications that make use of the LaunchAnywhere Java launcher technology. Generally, it is simple to alter the LAX file to allow the launcher to always generate output. This behavior can then be changed upon qualification and final release.



Task:

To generate debug output for a LaunchAnywhere executable

1. In the InstallAnywhere Advanced Designer, select the launcher.
2. In the Create LaunchAnywhere for Java Application customizer, click **Edit Properties**.
3. Set the values for `lax.stderr.redirect` and `lax.stdout.redirect`:

Property	Value
<code>lax.stderr.redirect</code>	<code>output.txt</code>
<code>lax.stdout.redirect</code>	<code>output.txt</code>

4. After a normal installation, edit the .lax file as described in the preceding instructions.

Debugging During Post Development

There are several methods available to debug installers without access to the Advanced Designer (post-development debugging). The following topics provide tips for obtaining debug output from installers on various target platforms.

These methods override the debug settings of the current installer (not including the data from Output Debug Information actions). Any **Installer Debug Output** settings you provided in **Project > Log Settings** or `lax.stderr.redirect/lax.stdout.redirect` settings you provided for a launcher are supplanted by the override setting.



Note • None of these methods has any effect on the data generated by an **Output Debug Information** action.

Table 6-18 • Debugging During Post Development

Topic	Description
Debugging a Win32 Installer	Explains how to capture debug output from an installer running on a Windows machine.
Debugging a Unix/Linux or Pure Java Installer	Provides pointers on debugging Unix/Linux installers by setting a LAX_DEBUG environment variable and on debugging Pure Java installers by setting a LAX_DEBUG Java property.
Debugging a Mac OS X Installer	Shows how to direct stderr and stdout by setting values in the installer's Info.plist.

Debugging a Win32 Installer

To view or capture the debug output from a Win32 installer, hold down the <CTRL> key immediately after launching the installer and until a console window appears. Before exiting the installer, copy the console output to a text file for later review.

On some Windows NT systems, run the installer once with the <CTRL> key down, resetting the scroll back buffer for the console window, then quit and run the installation again.

If there are problems capturing the console output, try a slightly more convoluted method (this will often be the case on Win9x because of the limited ability of the console to capture output). First launch the installer and allow it to extract the necessary files. Once it reaches the **Preparing to Install** window, when given the opportunity to choose a language or to go to the Windows temp directory, look for a temp folder which starts with an `l` followed by many numeric digits (for example: `l1063988642`). Be sure it is the most recent directory by sorting the directories by their **modified** date. Open the directory, there should be a file called `sea_loc`. Delete this file. Now return to the installer, click **OK**, and at the first opportunity, cancel the installation.

Now go back to the directory inside the temp directory, where the file sea_loc was deleted. There should be another directory called Windows; open it. There should be an .exe file (most likely install.exe). There should also be another file with the same name except it will have a .lax extension. Open it with a text editor and set the lax.stderr.redirect and lax.stdout.redirect properties:

```
lax.stderr.redirect=output.txt  
lax.stdout.redirect=output.txt
```

After these changes have been made, save the file and launch the .exe. When the installation is complete there should be an output.txt file in the same directory as the .lax file. The output.txt file contains the same information as that sent to the console.

Debugging a Unix/Linux or Pure Java Installer

To debug Unix/Linux installers and Pure Java installers without access to the InstallAnywhere Advanced Designer, set LAX_DEBUG to True.



Task: To capture the debug output from the Unix command line

1. Enter one of the following (based on which shell) at the command line:

```
export LAX_DEBUG=true  
setenv LAX_DEBUG true  
LAX_DEBUG=true  
set LAX_DEBUG
```



Note • Use whatever syntax is appropriate for the Unix shell being used.

2. Run the installer. The output that is produced is sent to the console.



Important • The Choose Java VM Console action traces symbolic links for any data entered as a path to a VM. This can lead it to display an unfamiliar path.



Task: To capture debug output from a Pure Java installer

At the command line, start the Pure Java installer and pass the **-DLAX_DEBUG=true** option to the JVM. For example:

```
java -DLAX_DEBUG=true -jar install.jar
```

Like setting a LAX_DEBUG environment variable for Unix/Linux installers, passing the LAX_DEBUG Java property to a Pure Java installer sends debug output to the console.

Debugging a Mac OS X Installer

InstallAnywhere utilizes the standard output layers in Mac OS X to display output. To gather debugging output from an OS X installer, launch Console.app (found in /Applications/Utilities). To retain this information, cut and paste information from the console window to a file.

If you do not see debug output from an installer, check the Info.plist file inside the installer. To do this, Control-click (or right-click) on the installer and select Show Package Contents. Inside the Contents folder you will see an XML file named Info.plist. You'll need to change:

```
<key>lax.stderr.redirect</key>
<string></string>
<key>lax.stdout.redirect</key>
<string></string>
```

to

```
<key>lax.stderr.redirect</key>
<string>console</string>
<key>lax.stdout.redirect</key>
<string>console</string>
```

When you relaunch the installer, the installer output should be now listed in Console.app.

Other Debugging Issues

Review the following topics for issues related to specific platforms:

- [Mac OS X Magic Folder Behavior](#)
- [EBCDIC Encoding Issues](#)
- [Reviewing Debug Information](#)

Mac OS X Magic Folder Behavior

The following table describes the behavior associated with both authenticated end users and regular end users on Mac OS X.

Table 6-19 • Mac OS X Behavior

Magic Folders	User Without Administrative Privileges	User Without Administrative Privileges Post- Authentication
Cleanup at Startup	Trash	Trash
Desktop	Desktop	Desktop
Dock	User Dock	User Dock
Fonts	User Fonts	System Fonts

Table 6-19 • Mac OS X Behavior (cont.)

Magic Folders	User Without Administrative Privileges	User Without Administrative Privileges Post- Authentication
Group	The User Group	Admin Group
Installation Drive Root	/, /Volumes/*	/, /Volumes/*
JavaHome	JavaHome	JavaHome
Owner	The User	Root
Owners/Permissions	User without Administrative Privileges	User without Administrative Privileges Post Authentication
Preferences	User Preferences	User Preferences
Programs Folder	Applications	Applications
System	System	System
System Drive Root	/	/
User Applications	User Applications	User Applications
User Home	User Home	User Home



Note • An asterisk (*) means all mounted volumes (partitions) except Startup.



Tip • Using an authenticated installer allows the installer to be run as the Root user. Mac OS X Choose Folder panels have aliases to such locations as Desktop or Home. When an end user selects these from a native Mac OS X dialog box, the locations are set to the Root user's Desktop or Home. Consequently, an end user may not be able to access these files installed to these locations. In such cases, consider parsing the return paths using the Match Regular Expression Rule and alerting end users to this special case.

EBCDIC Encoding Issues

InstallAnywhere installers work with EBCDIC-native systems. You can build installers for these systems with InstallAnywhere on an ASCII-native system; however, some InstallAnywhere actions may yield unexpected results if you are not clear on how they work.

Installing Files

InstallAnywhere installers deploy files on the target system with a binary copy mechanism. Text files are installed the same way. When you build an installer for an EBCDIC-native target system on a Windows development system, for example, the installer deploys text files in ASCII format. To install EBCDIC-format files, first convert those files using a conversion tool like native2ascii (part of the J2SE SDK) and then add them to the installer separately. You can use Check Platform rules on both the ASCII and EBCDIC text files to control which files are installed on which platforms.



Note • The **System i (i5/OS) Install File** action contains text conversion controls in its customizer. Therefore, no manual text conversion steps are necessary. Users can employ the **Source CCSID** and **Target CCSID** text boxes to define the text conversion required for that action.

Modifying Existing Files

Actions, such as **Modify Text File - Single File** and **Modify Text File - Multiple Files**, perform their modifications in the target system's native encoding. To make ASCII-format changes to a text file on an EBCDIC-native target system, execute the Modify Text File action and then use a format-conversion tool, like native2ascii, to convert the file to ASCII format after the action has completed.

Reviewing Debug Information

InstallAnywhere produces debug information in two distinct ways:

- Installer Debug Output is generated when you direct debug output from the installer using the **Project > Log Settings** task.
- An alternate, customizable set of debug information is derived from the Output Debug Information action.

Reviewing Installer Debug Output

Installer debug output generally appears as in the following sample (truncated):

```
IAResourceBundle: create resource bundle: en
```

```
-----  
InstallAnywhere 2010 Enterprise  
Version: 10.0
```

```
-----  
Fri Jun 27 17:13:04 CDT 2008
```

```
Free Memory: 24575 kB  
Total Memory: 22581 kB
```

```
No arguments
```

```
java.class.path:
```

```
C:\Program Files\InstallAnywhere 2011 Enterprise\resource  
C:\Program Files\InstallAnywhere 2011 Enterprise\resource\swingall.jar  
C:\Program Files\InstallAnywhere 2011 Enterprise\resource\compiler.zip  
C:\Program Files\InstallAnywhere 2011 Enterprise\IAClasses.zip  
C:\Program Files\InstallAnywhere 2011 Enterprise\lax.jar  
C:\program files\InstallAnywhere 2011 Enterprise\jre\lib\rt.jar
```

```
ZGUtil.CLASS_PATH:
```

```
C:\Program Files\InstallAnywhere 2011 Enterprise\resource  
C:\Program Files\InstallAnywhere 2011 Enterprise\resource\swingall.jar  
C:\Program Files\InstallAnywhere 2011 Enterprise\resource\compiler.zip  
C:\Program Files\InstallAnywhere 2011 Enterprise\IAClasses.zip  
C:\Program Files\InstallAnywhere 2011 Enterprise\lax.jar  
...  
java.version      = 1.5.0  
java.vm.vendor   = Sun Microsystems Inc.  
java.class.version = 45.3  
java.home        = c:\program files\InstallAnywhere 2011 Enterprise\jre\bin\..  
...
```

The debug information shows vital information such as the VM in use, the VM version, the locale, the system architecture, the OS, and other features.

Reviewing Data from the Output Debug Information Action

The Output Debug Information action produces output data according to the data options you select in the action's customizer.

InstallAnywhere Variables

The content changes depending on settings made within the install. Output is sorted alphabetically and shows the contents of the all available InstallAnywhere Variables, including special variables (such as \$USER_INSTALL_DIR\$), properties read from the end user's environment, Java properties (denoted by variable names beginning with prop.), InstallAnywhere installer properties (denoted by variable names beginning with lax.), and variables that are defined by you.

```
InstallAnywhere Variables:  
/=$prop.file.separator$  
:=${prop.path.separator$}  
EXTRACTOR_DIR=/local0/home/test/proj/Test_Installers/InstData/UNIX/Solaris  
EXTRACTOR_EXECUTABLE=/local0/home/test/proj/Test_Installers/InstData/UNIX/Solaris/install.bin  
INSTALL_DRIVE_ROOT=/  
INSTALLER_LOCALE=en  
JAVA_HOME=/local0/home/test/Test/jre...
```

Magic Folders

This section lists Magic Folders, their InstallAnywhere variable names, and their values.

```
Magic Folders:  
Installer Temp Directory ($INSTALLER_TEMP_DIR$)=/private/var/tmp/folders.501/Cleanup At Startup/  
032038  
Temp Directory ($TEMP_DIR$)=/private/var/tmp/folders.501/Cleanup At Startup  
User Applications Folder ($MACX_USER_APPLICATIONS$)=/Users/  
...
```

Java Properties

Output includes InstallAnywhere installer properties (denoted by variable names beginning with lax.) and the Java properties. You can find a short set of Java properties at:

<http://java.sun.com/docs/books/tutorial/essential/system/properties.html>

```
Java System Properties:  
lax.n1.env.openwinhome=/usr/openwin  
java.vendor=Sun Microsystems Inc.  
lax.n1.env.dtend-usersession=raman-bay-0  
lax.n1.env.StartDtscreenPyro=StartDtscreenPyro  
...
```

Visual Tree

This section lists the names of all the files in the tree hierarchically. Each entry contains the fully qualified package name of its location in the tree and its own name.

Visual Tree:

```
-- com.zerog.ia.installer.Installer -- Test
  |-- com.zerog.ia.installer.InstallSet -- Typical Install
  |-- com.zerog.ia.installer.GhostDirectory -- Test (Destination Install Folder)
    |   |-- com.zerog.ia.installer.actions.InstallDirectory -- UninstallerData
    |   |   |-- com.zerog.ia.installer.actions.InstallUninstaller -- Uninstall Test
    |   |   |-- com.zerog.ia.installer.actions.InstallDirectory -- resource
    |   |   |   |-- com.zerog.ia.installer.actions.InstallDirectory -- i18nresources
    |   |   |   |-- com.zerog.ia.installer.actions.InstallFile -- custom_en
    |   |   |   |-- com.zerog.ia.installer.actions.InstallFile -- remove.sh
    |   |-- com.zerog.ia.installer.actions.DumpDebugInfo -- Output Debug Information
...

```

Component Tree

This section shows the files and hierarchy of the installable components.

Component Tree:

```
-- com.zerog.ia.installer.Installer -- Test
  |-- com.zerog.ia.installer.InstallSet -- Typical Install
    |   |-- com.zerog.ia.installer.InstallBundle -- Application
      |   |   |-- com.zerog.ia.installer.Billboard -- (default)
      |   |   |-- com.zerog.ia.installer.actions.InstallUninstaller -- Uninstall Test
      |   |   |-- com.zerog.ia.installer.actions.DumpDebugInfo -- Output Debug Information
      |   |   |-- com.zerog.ia.installer.actions.MakeExecutable -- Uninstall_Test
...

```

Pre-Install Actions

This section prints the fully qualified package name and a text description of the action.

Pre-install Actions:

```
com.zerog.InstallAnywhere.installer.actions.CheckDiskSpace -- Check Free Disk Space
...

```

Post-Install Actions

This section prints the fully qualified package name and a text description of the action.

Post-install Actions:

```
com.zerog.InstallAnywhere.installer.actions.CheckDiskSpace -- Check Free Disk Space
...

```



Note • See the [General Actions](#) section for reference information on the Output Debug Information action.

Exit Codes

This section includes exit codes for InstallAnywhere installers and for the build.

Section	Description
Installer Exit Codes	Lists the exit codes for InstallAnywhere installers with descriptions for each code.
Build Exit Codes	Lists the exit codes for the build, grouped into categories and including basic descriptions.

Installer Exit Codes

By default, an installation process returns zero (0) to the environment if it was successful and a nonzero value if it was not. The possible exit codes returned during an installation and their definitions are listed below.

Table 6-20 • Installer Exit Codes

Code	Description
0	Success: The installation completed successfully without any warnings or errors.
1	The installation completed successfully, but one or more of the actions from the installation sequence caused a warning or a non-fatal error.
-1	One or more of the actions from the installation sequence caused a fatal error.
1000	The installation was cancelled by the user.
1001	The installation includes an invalid command line option.
2000	Unhandled error.
2001	The installation failed the authorization check, may indicate an expired version.
2002	The installation failed a rules check. A rule placed on the installer itself failed.
2003	An unresolved dependency in silent mode caused the installer to exit.
2004	The installation failed because not enough disk space was detected during the execution of the Install action.
2005	The installation failed while trying to install on a Windows 64-bit system, but installation did not include support for Windows 64-bit systems.
2006	The installation failed because it was launched in a UI mode that is not supported by this installer.

Table 6-20 • Installer Exit Codes (cont.)

Code	Description
3000	Unhandled error specific to a launcher.
3001	The installation failed due to an error specific to the lax.main.class property.
3002	The installation failed due to an error specific to the lax.main.method property.
3003	The installation was unable to access the method specified in the lax.main.method property.
3004	The installation failed due to an exception error caused by the lax.main.method property.
3005	The installation failed because no value was assigned to the lax.application.name property.
3006	The installation was unable to access the value assigned to the lax.nl.java.launcher.main.class property.
3007	The installation failed due to an error specific to the lax.nl.java.launcher.main.class property.
3008	The installation failed due to an error specific to the lax.nl.java.launcher.main.method property.
3009	The installation was unable to access the method specified in the lax.nl.launcher.java.main.method property.
4000	A Java executable could not be found at the directory specified by the java.home system property.
4001	An incorrect path to the installer jar caused the relauncher to launch incorrectly.
5000	Modification of existing instance failed because the instance has not been uninstalled properly or because the Registry has been corrupted.



Note • For information on exit codes from the command-line builder, see [Build Exit Codes](#).

Build Exit Codes

The command-line build tool returns an exit code based upon the results of the build as follows.

Table 6-21 • Build Exit Codes

Code	Status
Resource Related	
cancelled1	Missing resources, build abort cancelled by preferences
Project File Related	
101	Project load error
102	Project copy load error
103	Project file not found
104	Project file is read-only
199	Project file unknown error
Command Line Options Related	
200	Illegal build flag
201	Insufficient build flag
VM Pack Related	
300	VM Pack replaced
301	VM Pack not found
302	VM Pack illegal format
399	VM Pack unknown error
File Write Errors	
400	File write not found
401	File write busy
402	File write protected
403	File write error

Table 6-21 • Build Exit Codes (cont.)

Code	Status
499	File write unknown error
File Read Errors	
500	File read not found
501	File read busy
502	File read protected
503	File read error
599	File read unknown error
Other	
-1	Other error/unknown error
0	No errors. Build completed successfully without errors or warnings
666	Insufficient rights in directory
799	Unknown internal error

Application Server Support

InstallAnywhere Enterprise Edition includes support for many different application servers.

Table 6-22 • Supported Application Servers

Application Server	Versions	Required JVM
Geronimo	1.1.1 or newer	1.4 or newer
JBOSS	4.0.5 or newer	1.4 or newer
Resin	3.1.n	1.4 or newer
Sun Application Server	9	1.4 or newer
Tomcat	5.0, 5.5, and 6.0	1.4 or newer
WebSphere	6.1	1.5 only IBM JVM required to use security

Table 6-22 • Supported Application Servers (cont.)

Application Server	Versions	Required JVM
WebLogic	10.0	1.5 only



Note • *InstallAnywhere does not support remote deployment (**Bundle Connection Libraries from Local Server Install**) to WebSphere, Tomcat, JBoss, or Resin. WAR and EAR deployments to those servers requires the presence of connection libraries on the target system.*

Database Server Support

InstallAnywhere Enterprise Edition includes support for many common database servers as well as for generic JDBC connections (for users who need to run SQL scripts on a database server that is not officially supported).



Note • *Database server support is based, in part, on the JDBC drivers you use. For details on locating database drivers for use with Database Server hosts and Run SQL Script actions, see the following article in the InstallAnywhere Knowledge Base: <http://support.installshield.com/kb/view.asp?articleid=Q113500>.*

Table 6-23 • Supported Database Servers

Database Server	Versions
DB2	9.n
Firebird	1.0.n, 1.5, and 2.0
Interbase	2007
MySQL	4.1, 5.0, 5.1, and 6.0 (alpha)
MS SQL Server	6.5, 7.0, 2000, and 2005
Oracle	10.2.0.1, 10.2.0.2, 10.2.0.3, and 10.2.0.4
PostgreSQL	7.0-8.3
Sybase ASE	10, 11, 12, and 15
Generic JDBC Connection	n/a

Reference

Reference information for InstallAnywhere is organized into the following sections:

Table 7-1 • InstallAnywhere Reference

Section	Description
Advanced Designer Interface	Describes the tasks, subtasks, and controls of the Advanced Designer.
Project Wizard Interface	Describes the frames and controls that comprise the Project Wizard.
Dialog Boxes	Includes details about common InstallAnywhere dialog boxes: New Project, Open Project, InstallAnywhere Preferences, About, and others.
Menus	Provides descriptions of the InstallAnywhere menus and menu commands available in the Advanced Designer interface.
Actions	Lists the general actions, panel actions, install actions, and System i (i5/OS) actions available to users in the Advanced Designer interface and describes their purpose and implementation.
Rules Reference	Identifies the rules available to users in Advanced Designer mode in the Project > Rules subtask and in action customizers.
Variables	Lists the standard InstallAnywhere variables, LAX properties, and Magic Folders.
Localization Reference	Includes a list of language codes and a table of common localizable elements.
Files and File Formats	Provides information on files such as the product registry, install logs, manifest files, debug output files, build properties files, and response files.

Table 7-1 • InstallAnywhere Reference (cont.)

Section	Description
Command-Line Reference	Includes descriptions and examples of command-line arguments for InstallAnywhere installers and uninstallers, the build, and InstallAnywhere launchers.
InstallAnywhere Ant Task Reference	Provides details on the InstallAnywhere Ant task definition, task settings, build parameters, platform options, build options, installer options, and configurations.
Custom Code APIs	Provides a brief overview of the InstallAnywhere APIs and links to InstallAnywhere javadocs.

Advanced Designer Interface

The Advanced Designer contains several tasks and subtasks which are displayed along the left side of the Advanced Designer development environment. The tasks are broken down into discrete subsets of the installer creation process.

The Advanced Designer divides the installer creation process into nine tasks that provide detailed control for intermediate and advanced users.

Table 7-2 • Advanced Designer Interface Tasks

Task	Description
Project	Displays project information, configures project settings, provides options to bundle or define acceptable VMs, and defines product, file installation, and localization options.
Installer UI	Sets user interface options such as the default UI mode, splash screen images, billboard graphics, panel backgrounds, installer help, and more.
Organization	Provides tools to arrange Install Sets, Product Features, Components, Merge Modules, DIM references, and Database Server and Application Server hosts.
Pre-Install	Represents the actions the installer takes prior to the Install task. Generally, this task collects the actions that execute before the installer deploys the payload of files to be installed.  Note • For a description of the similarities between all Install and Uninstall tasks, see Similarities Between Pre and Post Install and Uninstall Tasks .
Install	Contains the install actions that take place during the step when InstallAnywhere deploys files to the target system.

Table 7-2 • Advanced Designer Interface Tasks (cont.)

Task	Description
Post-Install	<p>Represents the actions the installer takes after the Install task. Generally, this task collects the actions that execute after the installer deploys files to the target system.</p>  <p>Note • For a description of the similarities between all Install and Uninstall tasks, see Similarities Between Pre and Post Install and Uninstall Tasks.</p>
Pre-Uninstall	<p>Represents the actions the uninstaller takes prior to the uninstall task. Generally, this task contains the actions that execute before the uninstaller removes installed files.</p>  <p>Note • For a description of the similarities between all Install and Uninstall tasks, see Similarities Between Pre and Post Install and Uninstall Tasks.</p>
Uninstall	<p>Represents the actions the uninstaller takes during un-installation. This task enables you to add, remove, or change some of the uninstall actions, giving you additional flexibility and more control over how the un-installation is performed.</p>
Post-Uninstall	<p>Represents the actions the uninstaller takes after the uninstall task. Generally, this task collects the actions that execute after the uninstaller removes installed files.</p>  <p>Note • For a description of the similarities between all Install and Uninstall tasks, see Similarities Between Pre and Post Install and Uninstall Tasks.</p>
Build	<p>Provides the options for building an installer including creating and selecting Build Configurations, setting build targets, bundling VM packs, and defining distribution options.</p>

Reorganization of Properties/Tasks in InstallAnywhere 2011

Significant reorganization of properties was performed in the InstallAnywhere 2011 release. Several new subtasks were created to consolidate existing properties and contain new properties including **Project > JVM Settings**, **Project > Variables**, and **Project > Log Settings**.

Also, settings previously found on the **Project > Config** and **Project > Java** subtasks have been moved to other locations in the Advanced Designer interface.

The following table lists where to find properties that existed in the InstallAnywhere 2010 release but which have moved to a new location starting with the InstallAnywhere 2011 release.

Table 7-3 • Property Reorganization Between InstallAnywhere 2010 and InstallAnywhere 2011

Property	Location in InstallAnywhere 2010	Location in InstallAnywhere 2011
Generate install log during installation <ul style="list-style-type: none">• Plain text format• XML format	Project > Info	Project > Log Settings
Uninstall install log	Project > Info	Project > Log Settings
Advertise Variables	Project > Info	Project > Variables
Configure Variables	Project > Info	Project > Variables
Windows JVM Search Paths <ul style="list-style-type: none">• Search Paths• Java Executable Patterns	Project > Platforms > Windows	Project > JVM Settings Search Panel Settings Tab / Windows Subtab
Unix JVM Search Paths <ul style="list-style-type: none">• Search Paths• Java Executable Patterns	Project > Platforms > UNIX	Project > JVM Settings Search Panel Settings Tab / UNIX Subtab
Windows Install Launcher Type <ul style="list-style-type: none">• Graphical Launcher• Console Launcher	Project > Platforms > Windows	Build Build Configuration Tab / Build Targets Subtab / Windows Build Target
Send stderr to Send stdout to	Project > Config	Project > Log Settings
Valid VM list	Project > Config	Project > JVM Settings Installer Settings Tab
Minimum Heap Size Maximum Heap Size	Project > Config	Project > JVM Settings Installer Settings Tab
Additional Arguments	Project > Config	Project > JVM Settings Installer Settings Tab

Table 7-3 • Property Reorganization Between InstallAnywhere 2010 and InstallAnywhere 2011

Property	Location in InstallAnywhere 2010	Location in InstallAnywhere 2011
Classpath List	Project > Java	Project > JVM Settings Installer Settings Tab
Add service support for custom code	Project > Java	Project > JVM Settings General Settings Tab
Install Bundled Virtual Machine <ul style="list-style-type: none"> • Only when installing a LaunchAnywhere • Only when a compatible VM is not found in the system 	Project > Java	Project > JVM Settings Installer Settings Tab
Do not remove bundled VM when uninstalling	Project > Java	Project > JVM Settings Installer Settings Tab
VM Install Folder	Project > Java	Project > JVM Settings Installer Settings Tab
VM Search Settings <ul style="list-style-type: none"> • Use installer's valid VM list • Use the valid VM list of all LaunchAnywhere actions • Use a specific valid VM list 	Project > Java	Project > JVM Settings Search Panel Settings Tab General Subtab
Use JCE Encryption	Project > Java	Project > Variables

Project

The **Project** task displays InstallAnywhere project information, configures project settings, provides options to bundle or define acceptable VMs, and defines product, file installation, and localization options. The **Project** task includes the following subtasks:

Table 7-4 • Project Task Subtasks

Subtask	Description
Info	Defines basic information about the installer, including the installer title and name, build location, installation log, and response file generation settings
Description	Sets the vendor and product information that uniquely identifies your product in the product registry.
File Settings	Defines timestamps of installed files and determine the overwrite behavior when attempting to install files that already exist in the target's install locations.
Platforms	Sets platform-specific options for Mac OS X, Windows, Unix, and System i.
Locales	Provides tools to define locale settings for your project.
Rules	Defines rules that apply to the installers generated by the project.
JVM Settings	Defines JVM search settings. <ul style="list-style-type: none">• General Settings Tab—Use to modify the classpath settings for your project's LaunchAnywhere applications.• Installer Settings Tab—Define a list of valid VMs for installers built from this project, set the heap size for the VMs, set optional installer arguments, decide whether to install the bundled/downloaded Java VM, and define classpath settings.• Installer Settings Tab—Specify criteria to use to find valid VMs for the application you are installing.
Variables	Advertise variables for merge modules, list variables to encrypt in or exclude from the response files, set security options, and view a list of all defined variables.
Log Settings	Enable logging during installation and also during the Maintenance Mode options of Uninstall, Add Features, Repair Installation, and Remove Features. Also configure logging options.
Advanced	Specify settings for Maintenance Mode, Instance Management, Rollback, and Build Configuration Tags.

Info

Use the **Info** subtask to define basic information about the installer, including the installer title and name, build location, and response file generation settings.

- [Project Information](#)
- [Project Locations](#)
- [Response File](#)
- [Product Registry](#)
- [Multiple Launch Settings](#)
- [Cancel Button Settings](#)

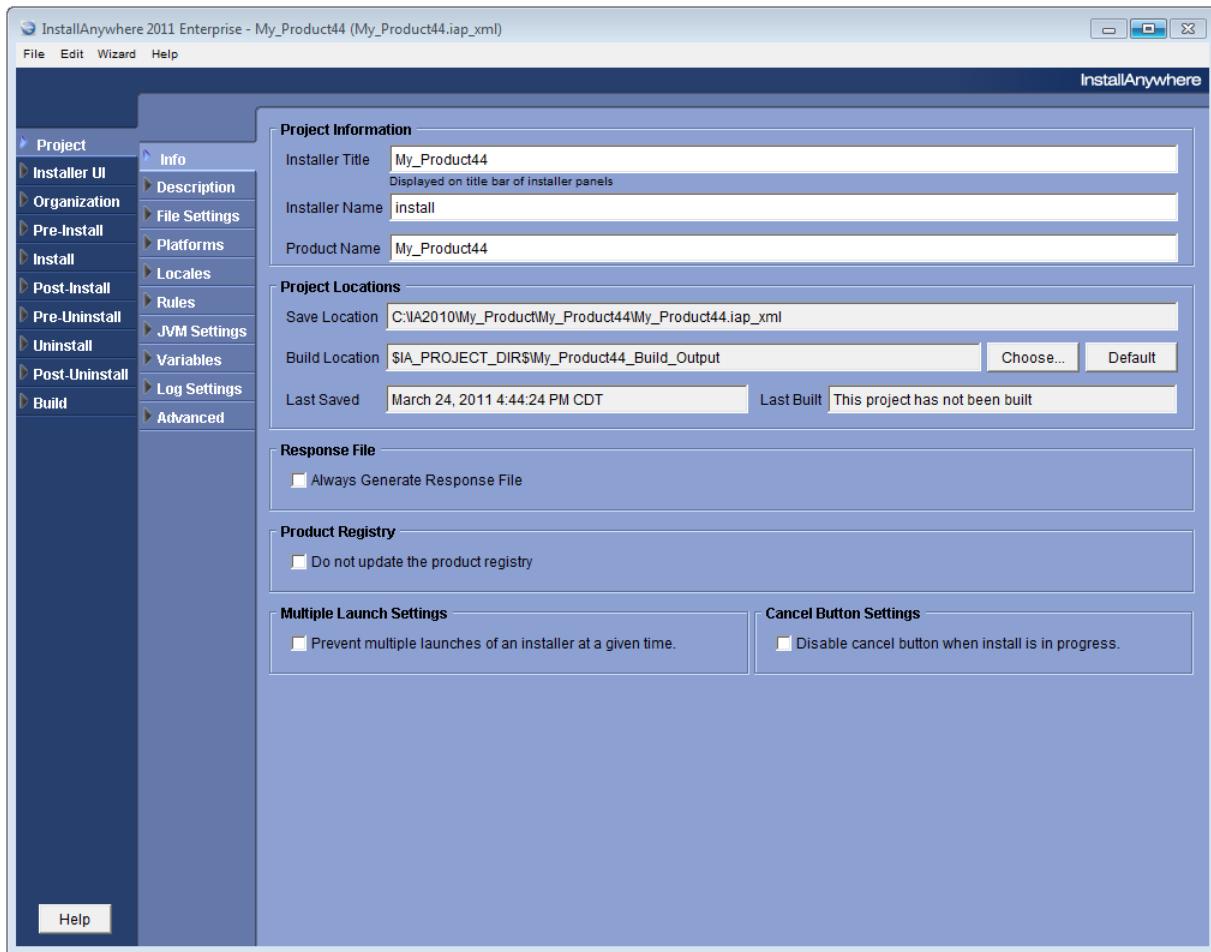


Figure 7-1: Project > Info Subtask

Project Information

The **Project Information** area includes the following controls:

Table 7-5 • Project Information Area of Project > Info Subtask

Control	Description
Installer Title	Shows the title that appears in the title bar of installers built from the project.
Installer Name	Shows the name of the installer files built from the project. For example, if you enter <code>myinstall</code> , the Windows installer file name will be <code>myinstall.exe</code> , the pure Java installer file name will be <code>myinstall.jar</code> , and the Unix installer file name will be <code>myinstall.sh</code> . 
Product Name	Shows the name of the product to be installed.

Project Locations

The **Project Locations** area includes the following controls:

Table 7-6 • Project Locations Area of Project > Info Subtask

Control	Description
Save Location	Shows the location where the project file is saved.
Build Location	Shows the location where your project's build output is stored. <ul style="list-style-type: none">• Click Choose to set a custom build location.• Click Default to reset the build location to the default.
Last Saved	Shows the date and time your project was last saved.
Last Built	Shows the date and time installers from your project were last built.

Response File

Select the **Always Generate Response File** option to activate the generation of response files (`installer.properties`). Response files record the default and user-specified settings from an installer's execution. These recorded settings can later be employed to provide settings for a silent install.

When **Always Generate Response File** option is selected, InstallAnywhere creates `installer.properties` files in the same location as the project's installers at build time.



Note • For more information, see [About Response Files and Silent Installers](#).

Product Registry

Select the **Do not update the product registry** option to prevent this project's installers from updating the InstallAnywhere product registry. By default, InstallAnywhere installers update the product registry when they are run. Select the **Do not update the product registry** option to leave the product registry unchanged.

Multiple Launch Settings

Select the **Prevent multiple launches of an installer at a given time** option to prevent end users from being permitted to launch multiple simultaneous instances of the same installer or uninstaller (as identified by having the same Product ID). Invoking multiple simultaneous instances of the installer or uninstaller could corrupt the InstallAnywhere registry.



Note • If there are multiple users connected to a machine, such as in a UNIX environment, and each user launches the same installer simultaneously, then the selection of the **Prevent multiple launches of an installer at a given time** option will not prevent multiple simultaneous instances of this installer from launching.

Cancel Button Settings

Select the **Disable cancel button when install is in progress** option to prevent end users from being able to cancel an installation using the **Cancel** button (or the close [X] button). Disabling the **Cancel** button during installation helps to avoid unfinished installations.



Note • This option is not applicable for uninstallers; by default, the **Cancel** button is disabled in uninstallers.

Interaction of Disabled Cancel Button with “Enable Rollback” Option Selection

The end result of disabling the **Cancel** button depends upon whether the **Enable Rollback** option on the **Project > Advanced** subtask is also selected. The following table presents the two scenarios:

Table 7-7 • Interaction of Disabled Cancel Button with “Enable Rollback” Option Selection

If “Enable Rollback” is ...	This will occur...
Not Selected	<ul style="list-style-type: none">The Cancel button and the window close [X] button are disabled.If a fatal error is encountered or a Trigger Rollback action occurs, installation will complete with errors. [Rollback is suppressed.]
Selected	<ul style="list-style-type: none">The Cancel button and the window close [X] button are disabled.If a fatal error is encountered or a Trigger Rollback action occurs, rollback will be initiated. [Rollback is enabled.]



Note • This same behavior occurs in Maintenance Mode-enabled projects when an attempt is made to Add Features, Remove Features, or Repair Installation.

Description

Use the **Description** subtask to enter vendor and product information that uniquely identifies your product in the product registry.

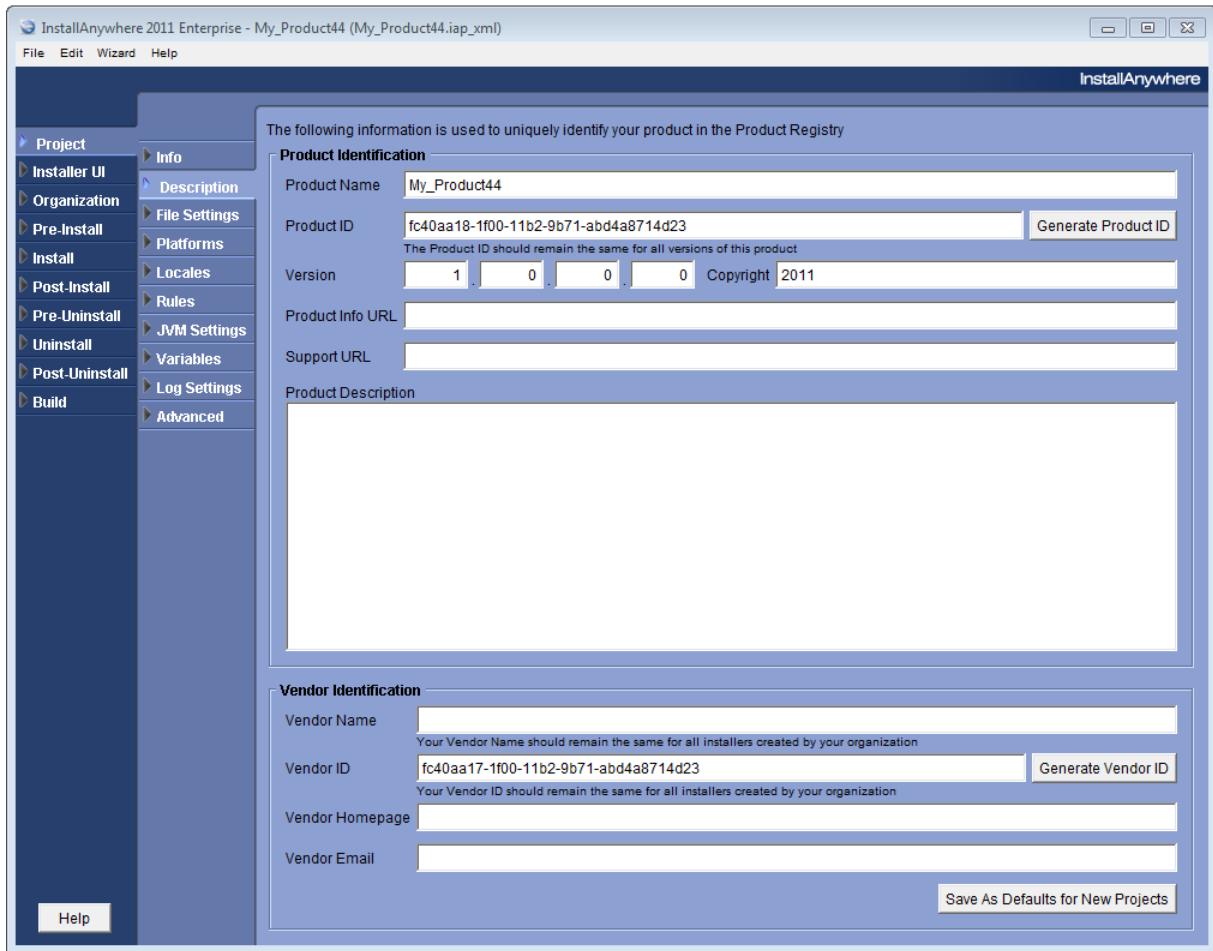


Figure 7-2: Project > Description Subtask

Product Identification

Product identification information is stored in the registry. The registry tracks specific instances of your product and its components.

Table 7-8 • Product Identification Controls on the Project > Description Task

Control	Description
Product Name	The name of your product.
Product ID	A GUID that identifies your product in the registry. You may have several products installed that have the same name—use this ID to track a specific instance of the product. The GUID must be in the format displayed.
Generate Product ID	Click Generate Product ID to add a unique product ID to your project.
Version	Type in the version of your product.
Copyright	Type copyright year here.
Product Info URL	A URL for information about your product.
Support URL	A URL for product support.
Product Description	A description of your product for registry purposes.

Vendor Identification

Vendor identification information helps track the products your installer deploys.

Table 7-9 • Vendor Identification Controls on the Project > Description Task

Control	Description
Vendor Name	The company name of the vendor.
Vendor ID	A unique GUID that identifies the vendor of the products this project's installers deploy.
Generate Vendor ID	Click Generate Vendor ID to add a unique vendor ID to this project.
Vendor Homepage	The vendor homepage URL.
Vendor Email	The vendor email address.
Save As Defaults for New Projects	Saves the current vendor information for use with any new projects you create.

File Settings

Use the **File Settings** subtask to define timestamps of installed files and determine the overwrite behavior when attempting to install files that already exist in the target's install locations.

Timestamps are considered when an installation file already exists on the target system.

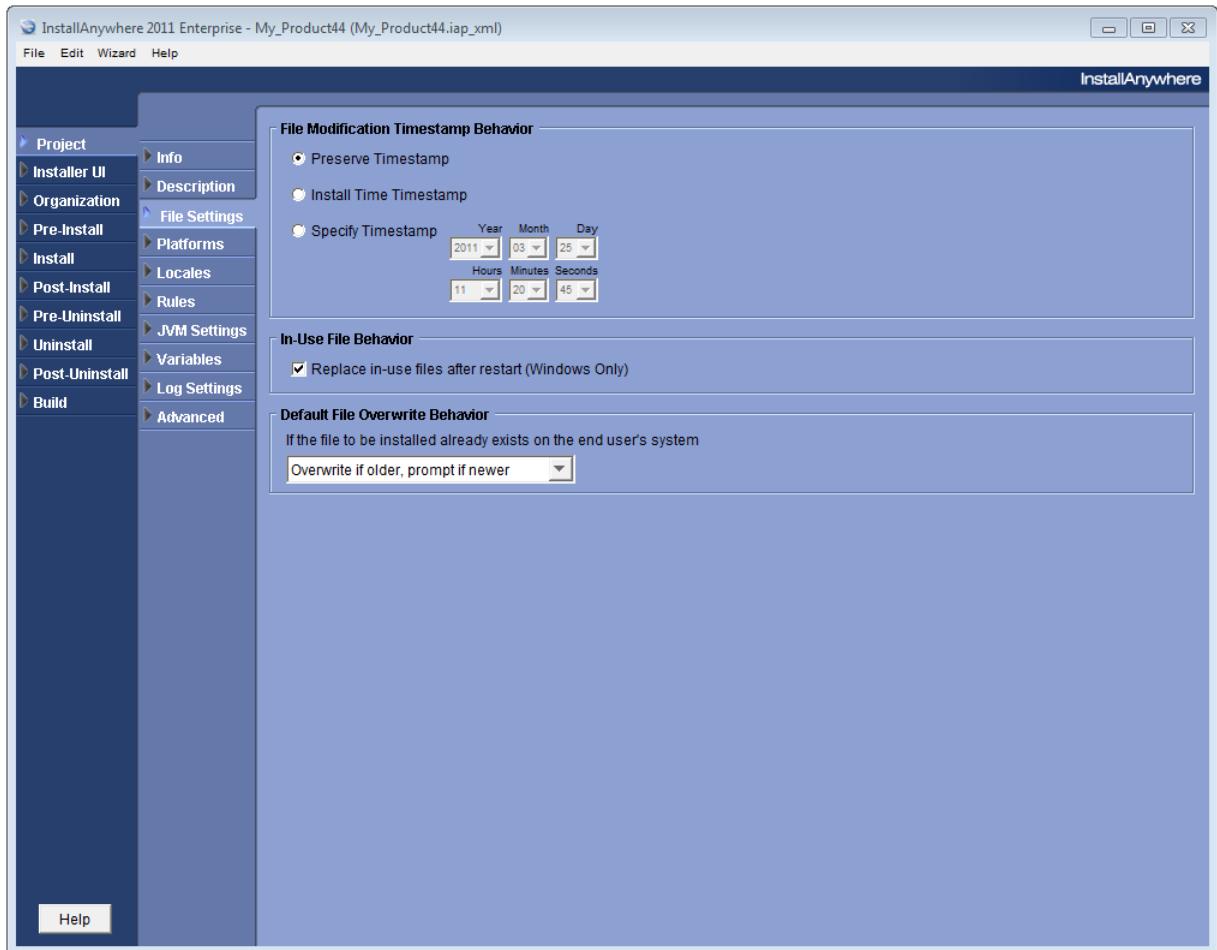


Figure 7-3: Project > File Settings Subtask

File Modification Timestamp Behavior

Use the timestamp behavior controls to specify how your project's installers handle timestamps.

Table 7-10 • Timestamp Controls

Control	Description
Preserve Timestamp	Maintains the existing timestamp on a file. This is the time that the file was last modified. For example, if you modify a file and then add it to be installed, the timestamp reflects the date you last modified the file, not the time it was installed.
Install Time Timestamp	Places the install-time as the timestamp on installed files. In this instance, the file creation property and the file modification property are identical, and reflect the time that the files were installed on the target system.
Specify Timestamp	Applies the date and time you set to the files your installers deploy.



Note • The InstallAnywhere Advanced Designer displays all timestamps in your system's local time zone. Behind the scenes, InstallAnywhere automatically maintains those timestamps in Greenwich Mean Time (GMT).

In-Use File Behavior

For Windows installers only, use the **Replace in-use files after restart** option to specify how your installer manages changes to files that are locked (in-use) at install time.

Default File Overwrite Behavior

Use this control to identify how your project's installers handles the installation of files that already exist on the target system. Choose from:

- **Always Overwrite**
- **Never Overwrite**
- **Overwrite if Older, Prompt if Newer**
- **Overwrite if Older, Do Not Install if Newer**
- **Prompt if Older, Do Not Install if Newer**
- **Always Prompt User**

Platforms

Use the **Platforms** task to define default settings unique to each target operating system. While InstallAnywhere runs on any Java-enabled platform, there are features that should be defined separately for each target operating system.

The **Platforms** task includes subtasks for the following systems:

- [Mac OS X](#)
- [Windows](#)
- [UNIX](#)
- [System i \(i5/OS\)](#)
- [Pure Java](#)

Mac OS X

The default settings for Mac OS X include default locations for install and alias folders, default Java VM used for LaunchAnywhere, whether authentication is required for installation, and permissions for files and folders created on the target system.

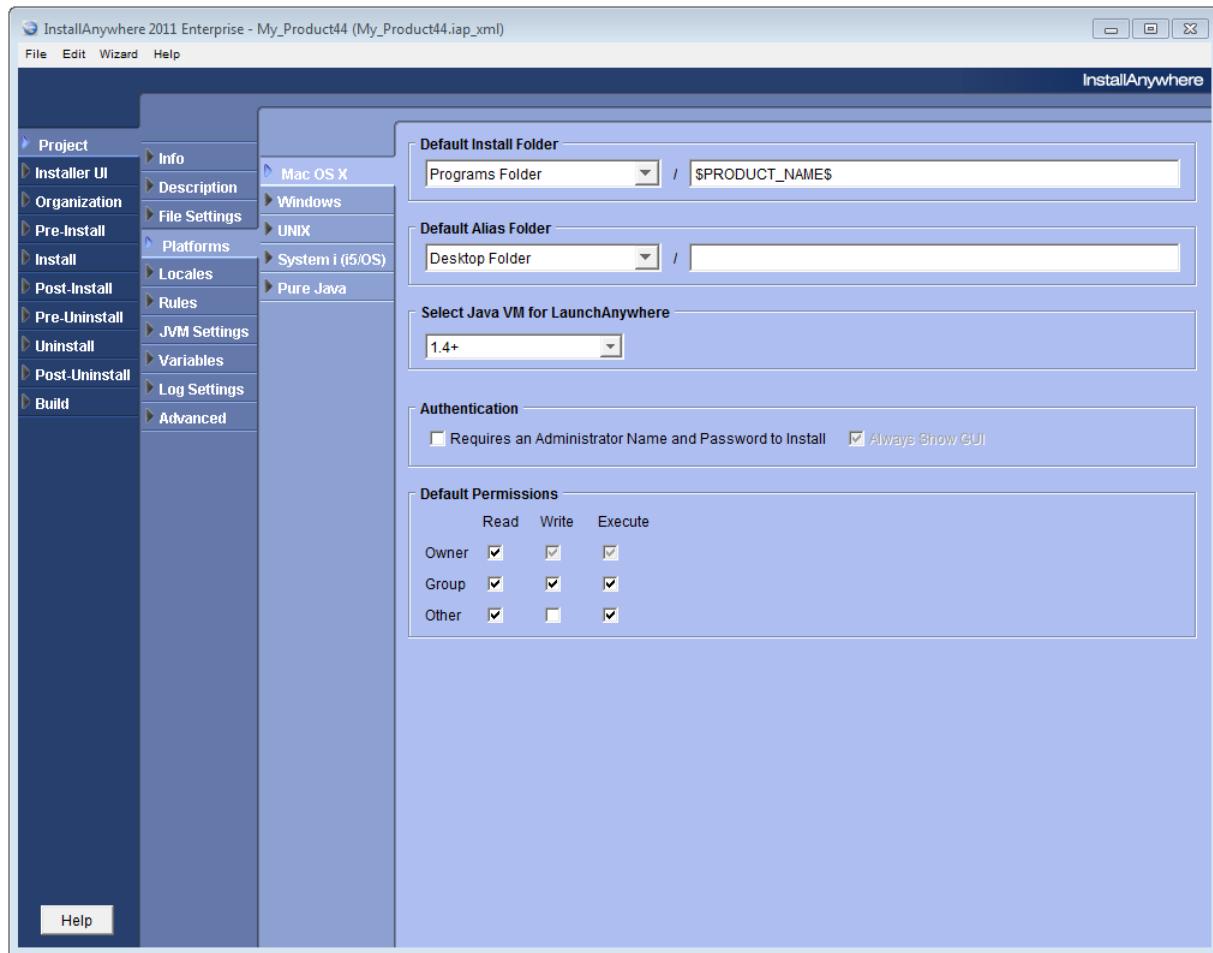


Figure 7-4: Mac OS X Platform Controls

The Mac OS X platform has the following controls:

Table 7-11 • Mac OS X Platform Controls

Control	Description
Default Install Folder	<p>Provide the default value for your install location. Select a magic folder from the list and specify a subdirectory as necessary.</p>  <p>Note • For more information on magic folders, see Magic Folders and Variables.</p>

Table 7-11 • Mac OS X Platform Controls (cont.)

Control	Description
Default Alias Folder	<p>Provide the default value for the alias location. Select a magic folder from the list and specify a subdirectory as necessary.</p> <p>This setting can be overridden by the user on the Choose Alias Folder panel.</p>  <p>Note • For more information on magic folders, see Magic Folders and Variables.</p>
Select Java VM for LaunchAnywhere	<p>Set the JVM required for LaunchAnywhere files. Select a VM version from the list. By default, Mac OS X installers use the most recent, valid VM available on the target system. Use the Select Java VM for LaunchAnywhere to cause LaunchAnywhere to use an older VM. For example</p> <ul style="list-style-type: none"> • Choose 1.4* to require LaunchAnywhere to use any 1.4 JVM. The VM version must be greater than or equal to 1.4.0_0 but less than 1.5.0_0. • Choose 1.4+ to require LaunchAnywhere to use any 1.4 or newer JVM. The VM version must be greater than or equal to 1.4.0_0. • Choose 1.5* to require LaunchAnywhere to use any 1.5 JVM. The VM version must be greater than or equal to 1.5.0_0 but less than 1.6.0_0. • Choose 1.5+ to require LaunchAnywhere to use any 1.5 or newer JVM. The VM version must be greater than or equal to 1.5.0_0. • Choose 1.6* to require LaunchAnywhere to use any 1.6 JVM. The VM version must be greater than or equal to 1.6.0_0 but less than 1.7.0_0. • Choose 1.6+ to require LaunchAnywhere to use any 1.6 or newer JVM. The VM version must be greater than or equal to 1.6.0_0.  <p>Note • Mac OS X installers do not show Choose Java VM panels nor do they respond to the settings in the Valid VM list field on the Installer Settings tab of the Project > JVM Settings task.</p>
Authentication	<p>Identify authorization requirements for your installer.</p> <ul style="list-style-type: none"> • Requires an Administrator Name and Password to Install—Select if your installer requires admin privileges to install. • Always Show GUI (default)—Select to ensure the Authenticate dialog box is shown. It is unnecessary to show the Authenticate dialog box when the user who runs the installer is logged in as the root user. When a user successfully authenticates, the installer can write to protected directories and files, such as /usr/bin and /usr/lib.
Default Permissions	<p>Set default Read, Write, and Execute permissions for your installer. These settings represent the default permissions for all files your installer deploys.</p>



Note • Merge modules cannot use authentication independently. To deploy a Merge Module that requires authentication, you must authenticate the parent installer.

Windows

The default settings for Windows include default locations for install and shortcut folders.

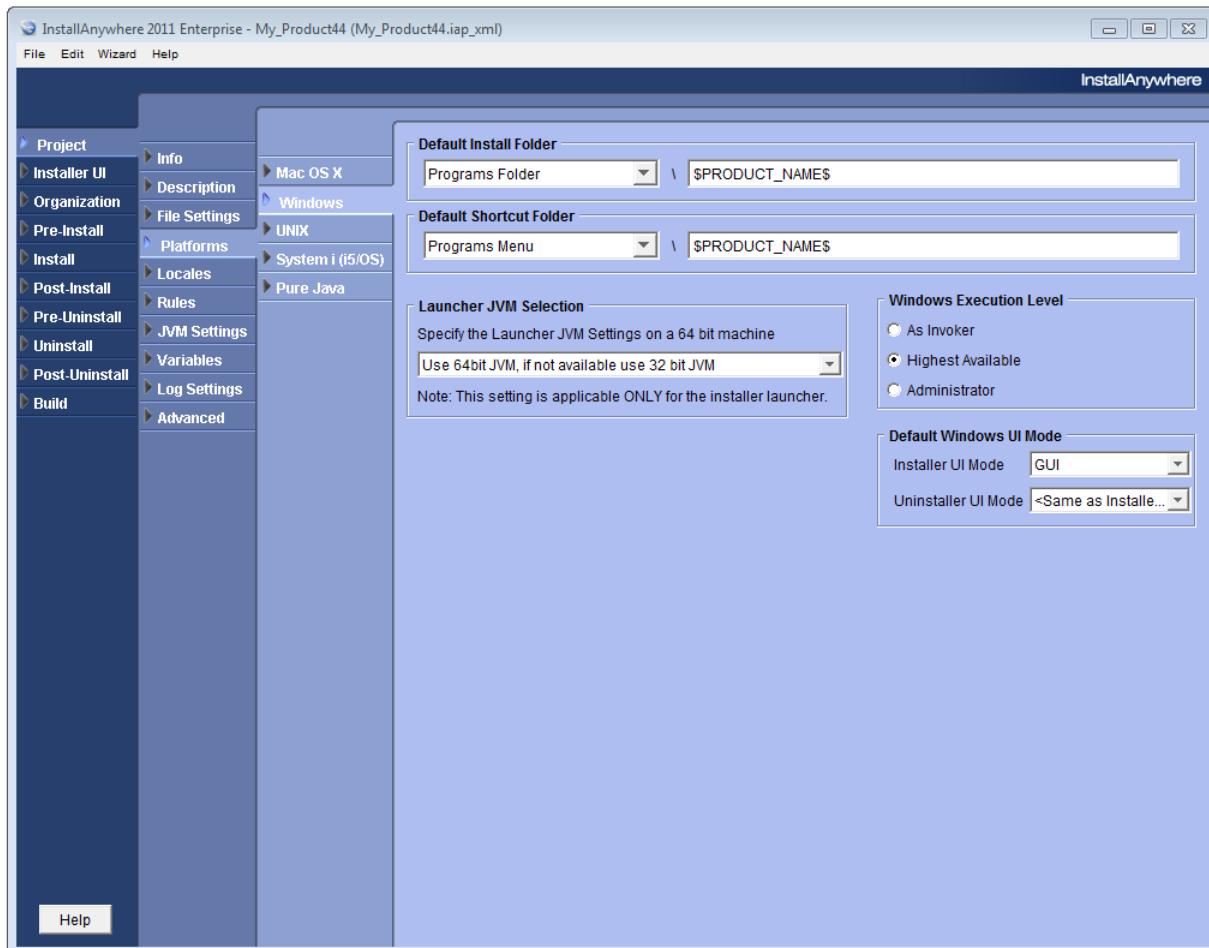


Figure 7-5: Windows Platform Controls

The **Windows** platform has the following controls:

Table 7-12 • Windows Platform Controls

Control	Description
Default Install Folder	<p>Provide the default value for your install location. Select a magic folder from the list and specify a subdirectory as necessary.</p>  <p>Note • For more information on magic folders, see Magic Folders and Variables.</p>
Default Shortcut Folder	<p>Provide the default value for the shortcut location. Select a magic folder from the list and specify a subdirectory as necessary.</p> <p>This setting can be overridden by the user on the Choose Shortcut Folder panel.</p>  <p>Note • For more information on magic folders, see Magic Folders and Variables.</p>
Launcher JVM Selection	<p>Specify which JVM the InstallAnywhere Launcher should use on a 64-bit machine—32-bit or 64-bit—if both versions exist on the machine. The following options are available.</p> <ul style="list-style-type: none"> • Use 64bit JVM, if not available use 32 bit JVM (default) • Use 32bit JVM, if not available use 64 bit JVM • Use 64bit JVM, if not available exit with error • Use 32bit JVM, if not available exit with error  <p>Note • This setting is applicable only for the installer launcher.</p>
Windows Execution Level	<p>Identify the execution level you want to request for your Windows launchers. Choose from</p> <ul style="list-style-type: none"> • As Invoker—The launcher acquires the same execution level as its parent process. • Highest Available—The launcher requests the highest execution level (Windows privileges and user rights) available to the current user. • Administrator—The launcher requires local admin privileges to run. Depending on the privileges of the current user account and the configuration of the target system, this setting may result in a launcher that will not start.

Table 7-12 • Windows Platform Controls (cont.)

Control	Description
Default Windows UI Mode	<p>Specify the default UI mode for both the installer and the uninstaller:</p> <ul style="list-style-type: none"> • Installer UI Mode—Select the UI mode for the Windows installer. Options are: GUI, Console, or Silent. • Uninstaller UI Mode—(Enterprise edition only) Select the UI mode for the Windows uninstaller. Options are: Same as Installer, GUI, Console, or Silent.  <p>Note • The choices available in these two lists depend, in part, on the Allowable UI Modes selected on the General UI Settings tab of the Installer UI > Look & Feel subtask. If, for example, Silent is not selected as an allowable mode in the Installer UI task, Silent is not listed in the Installer UI Mode or Uninstaller UI Mode lists. Only InstallAnywhere Enterprise Edition can create installers that operate in console and silent modes.</p>  <p>Note • For more information about silent and console installers, see Silent Installers and Console Installers.</p>

UNIX

The default settings for Linux include default locations for install and link folders, default user interface mode, permissions for files and folders created on the target system, and RPM (RedHat Package Management) settings for Linux installations. The RPM feature enables the installer to interact with and make entries into the RPM database.

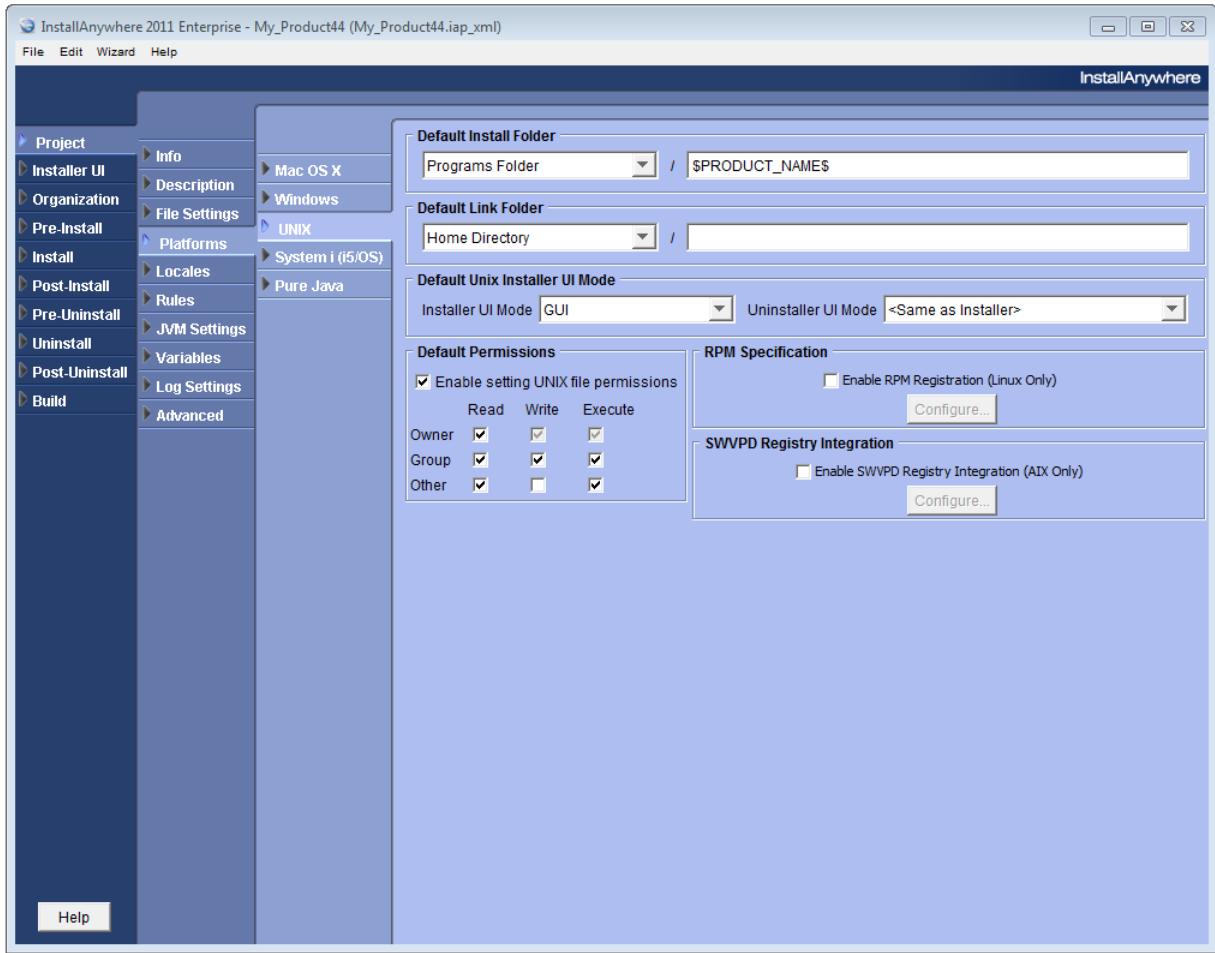


Figure 7-6: UNIX Platform Controls

The **UNIX** platform has the following controls:

Table 7-13 • UNIX Platform Controls

Control	Description
Default Install Folder	<p>Provide the default value for your install location. Select a magic folder from the list and specify a subdirectory as necessary.</p>  <p>Note • For more information on magic folders, see Magic Folders and Variables.</p>
Default Link Folder	<p>Provide the default value for the link location. (The default location is the user's home directory.) Select a magic folder from the list and specify a subdirectory as necessary.</p> <p>This setting can be overridden by the user on the Choose Link Folder panel.</p>  <p>Note • For more information on magic folders, see Magic Folders and Variables.</p>
Default Unix Installer UI Mode	<p>This section provides option to set the UI mode for the Unix installer and Unix uninstaller.</p> <ul style="list-style-type: none"> • Installer UI Mode—Set the UI mode for the Unix installer. Choose from GUI, Console, and Silent. • Uninstaller UI Mode—(Enterprise Edition Only) Use to set the UI mode for the Unix uninstaller. Choose from Same as Installer, GUI, Console, and Silent. <p>The choices available depend, in part, on the Allowable UI Modes selected on the General UI Settings tab of the Installer UI > Look & Feel subtask. If, for example, Silent is not checked as an allowable mode in the Installer UI task, Silent is not available as a default installer UI mode.</p>  <p>Note • Only InstallAnywhere Enterprise Edition can create installers that operate in console and silent modes.</p>  <p>Note • InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers will not run under those locales</p>  <p>Note • For more information about silent and console installers, see Silent Installers and Console Installers.</p>

Table 7-13 • UNIX Platform Controls (cont.)

Control	Description
Enable setting UNIX file permissions	<p>Select this option to activate or deactivate the setting of default permissions for the files the installer deploys.</p> <ul style="list-style-type: none"> • Selected—If you select this option, you can then set the default Read, Write, and Execute permissions to determine what the file's Owner can do with the file, what the file's owner Group can do with the file, and what everyone else can do with the file (Other). • Not selected—If you do not select this option, the default permissions for files the installer writes to the target system are determined by the target system.
Enable RPM Registration (Linux Only)	<p>Select this option to enable RPM registration. On supported Linux systems, RPM registration creates a virtual package and uses it to make entries to the RPM database.</p> <p>When you select the Enable RPM Registration option, the Configure button is then enabled. Click Configure to open the RPM Specification Settings dialog box. See RPM Specification Settings Dialog Box for detailed information.</p>
Enable SWVPD Registry Integration (AIX Only)	<p>Select this option to include AIX registry (Software Vital Product Data) support in your project's installers. On AIX systems, this option ensures that products are properly added to and removed from the SWVPD registry.</p> <p>When you select the Enable SWVPD Registry Integration (AIX Only) option, the Configure button is then enabled. Click Configure to open the SWVPD Registry Settings dialog box. See SWVPD Registry Settings Dialog Box for detailed information.</p> <p></p> <p>Note • As the SWVPD Registry Settings dialog box explains, when the Enable SWVPD Registry Integration option is checked, omitting values from any field causes installers to use corresponding values from the Project > Description subtask.</p>

System i (i5/OS)

The **System i (i5/OS)** subtask includes the controls to govern general settings for installers targeted at the i5/OS operating system running on System i. To build installers for i5/OS, you must have access to an i5/OS system.

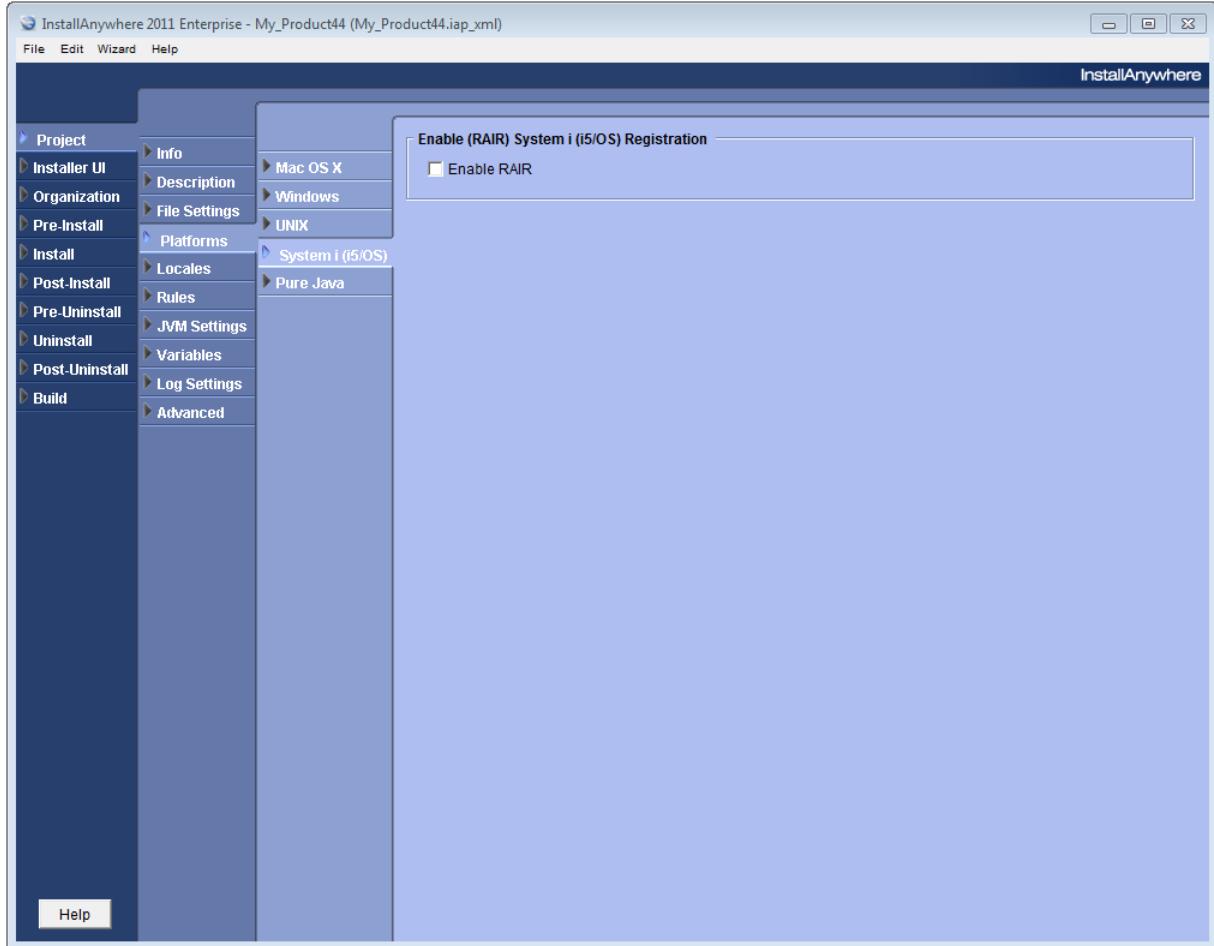


Figure 7-7: System i (i5/OS) Platform Settings

The System i (i5/OS) platform has the following controls:

Table 7-14 • System i (i5/OS) Platform Controls

Control	Description
Enable RAIR	<p>Select to cause the installer to enter product information in the i5/OS Registered Application Information Repository (RAIR). If this option is selected, the installer adds information about the product's features to the RAIR when the product is installed.</p>  <p>Note • Some systems management products, such as Management Central, use RAIR to determine what software is installed on an i5/OS system.</p>

Pure Java

The **Pure Java** subtask provide controls to determine the UI mode Pure Java installers and uninstallers use.

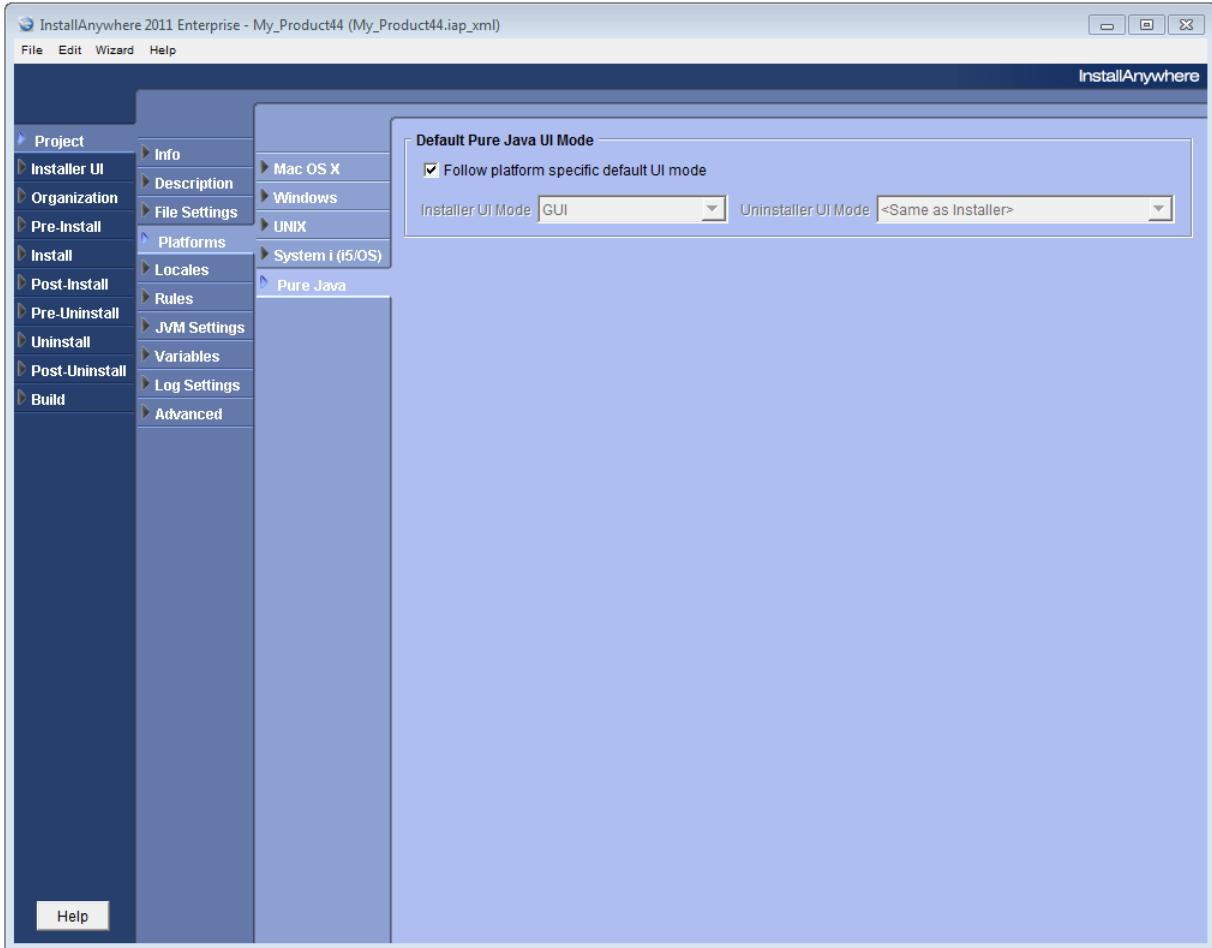


Figure 7-8: Pure Java Platform Settings

The **Pure Java** platform has the following controls:

Table 7-15 • Pure Java Platform Controls

Control	Description
Follow platform specific default UI mode	Determines whether or not Pure Java installers use the project's default UI mode. (Default: Checked.) Click (uncheck) Follow platform specific default UI mode to enable the Installer UI Mode and Uninstaller UI Mode controls. These controls allow you to specify a default UI mode for Pure Java installers and uninstallers.

Table 7-15 • Pure Java Platform Controls (cont.)

Control	Description
Installer UI Mode	<p>Specifies the UI mode to use for Pure Java installers.</p> <p>Choose from GUI, Console, and Silent.</p> <p>Available UI modes depend upon the current Allowable UI Modes settings on the General UI Settings tab of the Installer UI > Look & Feel task.</p> <p>Only InstallAnywhere Enterprise Edition can create installers that operate in console and silent modes.</p> <p>For more information about silent and console installers, see Silent Installers and Console Installers.</p>  <p>Note • <i>InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers will not run under those locales.</i></p>
Uninstaller UI Mode	<p>Specifies the UI mode to use for Pure Java uninstallers.</p> <p>Choose from Same as Installer, GUI, Console, and Silent.</p> <p>Available UI modes depend upon the current Allowable UI Modes settings on the General UI Settings tab of the Installer UI > Look & Feel task.</p> <p>Only InstallAnywhere Enterprise Edition can create installers that operate in console and silent modes.</p> <p>For more information about silent and console installers, see Silent Installers and Console Installers.</p>  <p>Note • <i>InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers will not run under those locales.</i></p>

Locales

Use the **Locales** subtask to set project-level language options.



Important • *Locales for a project are now selected on the **Locales** subtab of the **Build > Build Configurations** tab. For more information, see [Locales Subtab](#).*

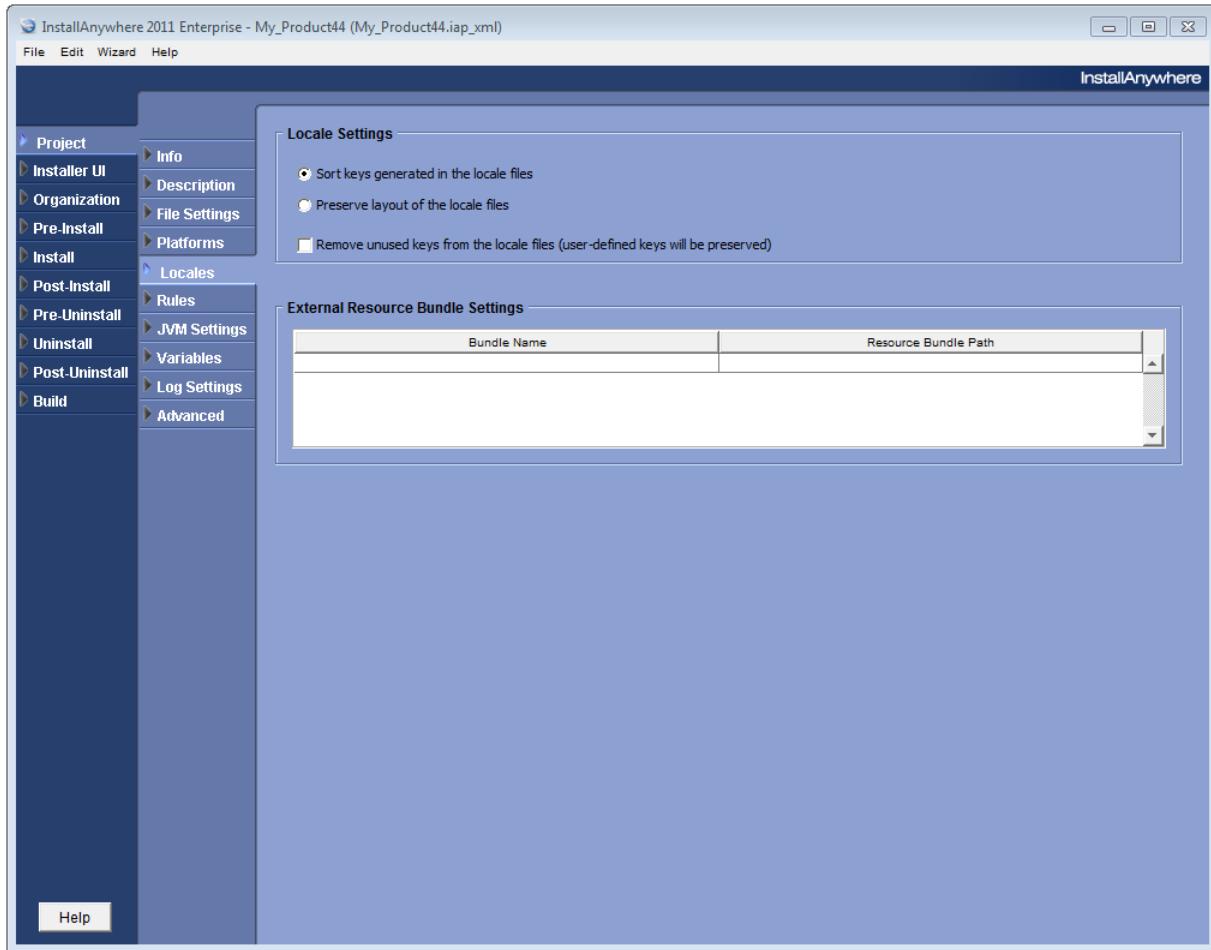


Figure 7-9: Project > Locales Subtask

The **Project > Locales** subtask includes the following controls:

Table 7-16 • Project > Locales Subtask Controls

Control	Description
Sort keys generated in the locale files	Automatically sorts keys in alphanumeric sequence within each locale file when you save the InstallAnywhere project with which the locale files are associated. As a product of the sorting routine, InstallAnywhere removes extra spaces and line breaks and deletes any non-auto-generated comments.
Preserve layout of the locale files	<p>Retains the existing sequence, formatting, and comments in each locale file when you save the InstallAnywhere project. InstallAnywhere does not sort the keys. All space, carriage return, and line feed characters are preserved (see note). With Preserve layout of the locale files enabled, InstallAnywhere retains all existing comments in the project's locale files.</p>  <p>Note • Even with this option set, the comments immediately preceding each key (containing the English version of the translated text) will still be overwritten.</p>
Remove unused keys from the locale files	<p>Deletes the keys associated with panel or console actions that have been removed. This option also removes any key you replace with a reference to an external resource bundle key.</p> <p>Setting the Remove Unused Keys option removes the unused keys that InstallAnywhere automatically generates for a panel or console each time you save the project. This option also removes the comments immediately preceding each key as well as the line break following each key-value pair. Keys associated with custom code are always preserved.</p>
External Resource Bundle Settings	<p>Shows any external resource bundles included in the project:</p> <ul style="list-style-type: none"> • Bundle Name—The name of the external resource bundle as it is referenced in your InstallAnywhere project. This name is used each time you reference a key in the bundle's locale properties files. (For more information, see Referencing an External Resource Key.) • Resource Bundle Path—The path to the locale properties files that make up the external resource bundle. <p>External resource bundles are available only in InstallAnywhere Enterprise edition.</p>  <p>Note • For more information about adding external resource bundles to the Locales task, see Adding an External Resource Bundle.</p>

Rules

Use the **Rules** subtask to add logic that executes prior to **Pre-Install** tasks. Use this option to check if the target system is the right platform for this installation or if the user is logged into the root or if the user has the necessary permissions to perform the installation.

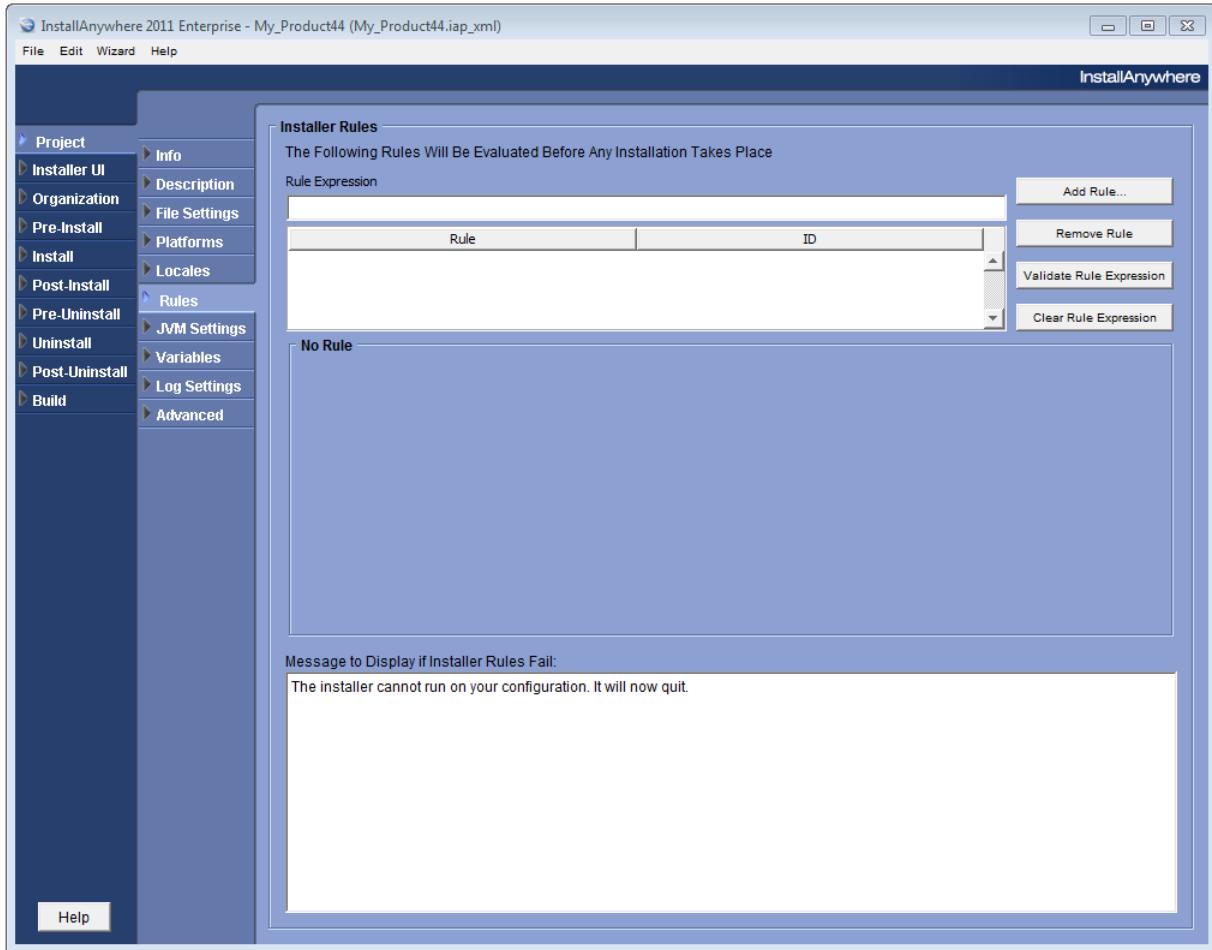


Figure 7-10: Project > Rules Subtask

The **Project > Rules** subtask includes the following controls:

Table 7-17 • Project > Rules Subtask Controls

Control	Description
Rule Expression	<p>When you click Add Rule to add a rule to your installation project, the unique rule ID of each rule is listed in the Rule Expression field.</p> <p>If you want to write complex rule expressions, you can edit the expression in this field to use multiple operators to express the relationship between two or more rules, such as:</p> <ul style="list-style-type: none">• && (and)—Means that both rules must evaluate true for the installer to continue. If any rule fails to pass, the installer stops and issues the failure message.• (or)—Implies that at least one of the rules must evaluate to true. If none of the rules pass, the installer stops and issues the failure message.• ! (not)—Implies that at least one of the rules must evaluate to false. If all of the rules pass, the installer stops and issues the failure message. <p>You can also use precedence operators—such as parentheses ()—to compose rule expressions.</p> <p></p> <p>Note • For more information, see Building Complex Rule Expressions.</p> <p></p> <p>Note • You can also write complex rule expressions in the Rules customizers on individual actions, panels or consoles.</p>
Rules List	<p>Shows all rules currently defined for your project.</p> <p></p> <p>Note • New projects have no rules set; thus, the Rules List is initially empty.</p>

Table 7-17 • Project > Rules Subtask Controls (cont.) (cont.)

Control	Description
Add Rule	<p>Opens the Choose a rule dialog box. Choose from</p> <ul style="list-style-type: none"> • Check File/Folder Attributes • Check Platform • Check System Architecture • Check User-Chosen Language • Compare InstallAnywhere Variables • Compare InstallAnywhere Variables Numerically • Evaluate Custom Rule • Match Regular Expression • System i (i5/OS) Licensed Program Exists Condition • System i (i5/OS) Primary Language Install Condition • System i (i5/OS) Program Temporary Fix (PTF) Condition <p>When you choose a rule, InstallAnywhere shows the controls for providing settings specific to that rule type in the area below the Rules table.</p>
Remove Rule	Deletes the currently selected rule from the Rules List.
Validate Rule Expression	Click to validate the rule expression listed in the Rule Expression field. If expression is invalid, an error message will be displayed.
Clear Rule Expression	Click to clear the expression in the Rule Expression field.
No Rule	This area shows the controls for providing settings specific to the currently selected rule. When no rules are selected, this area is labeled No Rule . When a Check Platform rule is selected, for example, this area is labeled Check Platform and shows platform selection controls.
Message to Display if Installer Rules Fail	The message your installers display if the expression in the Rule Expression field does not collectively evaluate to TRUE.

JVM Settings

Use the **JVM Settings** subtask to define JVM search settings. The **JVM Settings** subtask includes three tabs:

- **General Settings**—Use to modify the settings for the entire project, such as classpath settings for your project's LaunchAnywhere applications. See [General Settings Tab](#).

- **Installer Settings**—Use to modify the settings for the installer launcher, such as defining a list of valid VMs for installers built from this project, setting the heap size for the VMs, setting optional installer arguments, deciding whether to install the bundled/downloaded Java VM, and defining classpath settings. See [Installer Settings Tab](#).
- **Search Panel Settings**—Specify criteria to use to find valid VMs for the application you are installing. See [Search Panel Settings Tab](#).

General Settings Tab

On the **General Settings** tab, you can modify the classpath settings for your project's LaunchAnywhere applications.

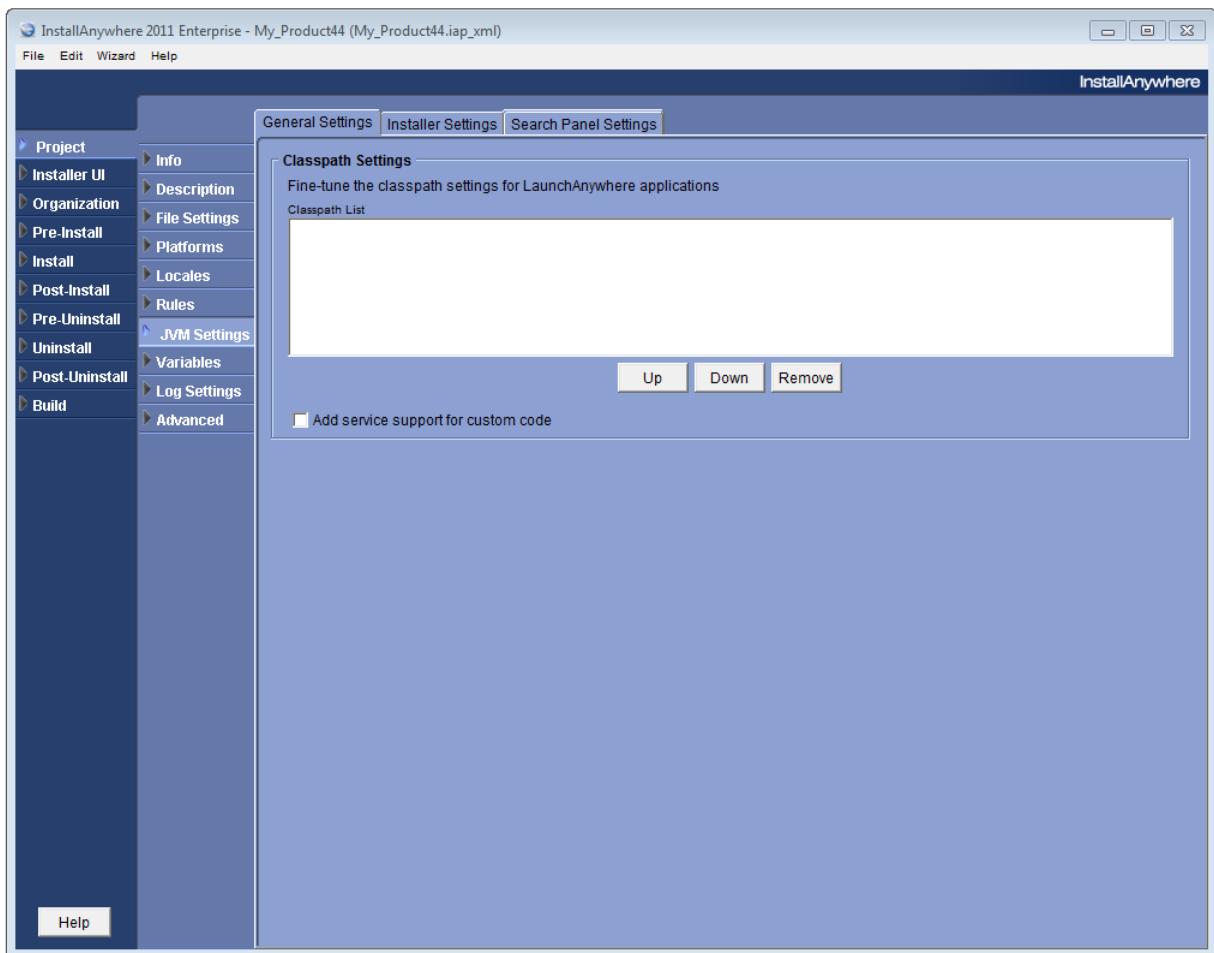


Figure 7-11: General Settings Tab of Project > JVM Settings Subtask

Use the following controls to modify the classpath settings for your project's LaunchAnywhere applications:

Table 7-18 • Classpath Settings Controls

Control	Description
Classpath List	Lists classpath settings for your project's LaunchAnywhere applications.
Up	Moves the currently selected entry up one position in the Classpath list.
Down	Moves the currently selected entry down one position in the Classpath list.
Remove	Deletes the currently selected entry from the Classpath list.
Add service support for custom code	Includes support for the custom code services layer.

Installer Settings Tab

Use the **Installer Settings** tab of the **JVM Settings** subtask to define a valid list of Java VMs the installer can use, set the heap size for the VMs, set optional installer arguments to send to the Java VM (in addition to the arguments the installer already sets), decide whether to install the bundled/downloaded Java VM, and define classpath settings.

- [Installer Valid VM List](#)
- [Installer VM Heap Size](#)
- [Optional Installer Arguments](#)
- [Bundled/Downloaded Virtual Machine](#)
- [Additional Classpath Settings](#)

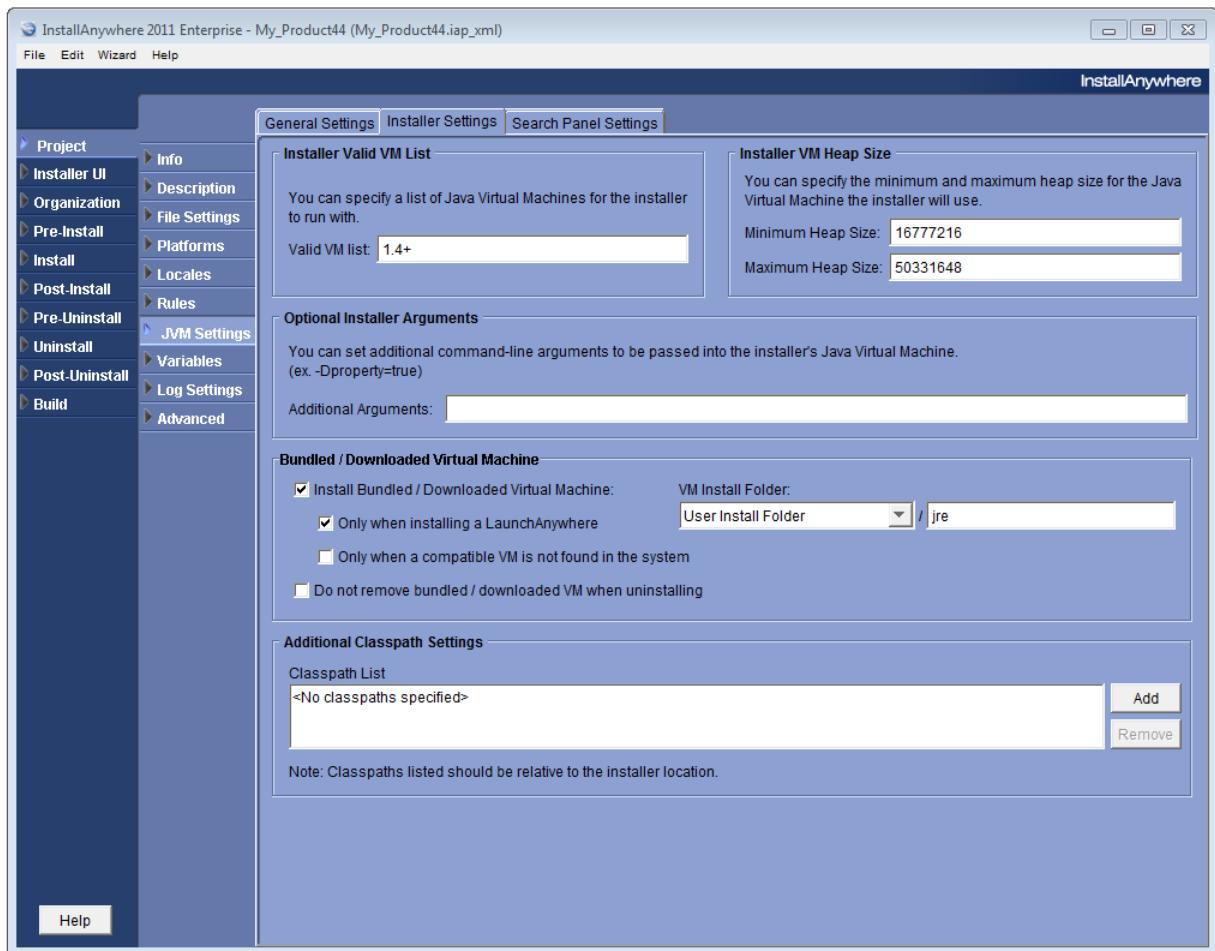


Figure 7-12: Installer Settings Tab of Project > JVM Settings Subtask

Installer Valid VM List

Type an operator that defines what VMs are valid for your project's installers. You can specify criteria for the virtual machine family, vendor, and version.

For example, **IBM_JRE_1.6+** selects any IBM JRE of version 1.6.0_0 or greater.



Note • See [About Java VM Selection Criteria](#) for details about the operators InstallAnywhere accepts for VM selection criteria.



Tip • These settings have no effect on Mac OS X installers. By default, Mac OS X installers use the most recent, valid VM available on the target system. Use the **Select Java VM for LaunchAnywhere** on the **Project > Platforms > Mac OS X** subtask to cause LaunchAnywhere to use an older VM for Mac OS X installers.

Installer VM Heap Size

Set the minimum and maximum heap size for the JVM used by the installer.



Note • Consider changing the heap size settings in response to out-of-memory conditions. For installers that install many files, you may need to increase the heap size.

Table 7-19 • Installer VM Heap Size controls

Control	Description
Minimum Heap Size	Enter the minimum heap size in bytes. This value corresponds to the LAX property lax.nl.java.option.java.heap.size.initial .
Maximum Heap Size	Enter the maximum heap size in bytes. This value corresponds to the LAX property lax.nl.java.option.java.heap.size.max .

Optional Installer Arguments

Enter command-line arguments to be passed to the installer's virtual machine in the **Additional Arguments** text box. For example, to run the installer in all supported locales, enter the following argument:

-Dlax.locales=all



Note • For more information on command-line arguments, see [Using Command-Line Arguments with Installers and Uninstallers](#) and [LAX Properties](#).

Bundled/Downloaded Virtual Machine

Use the following controls to determine how your InstallAnywhere installer handles bundled/downloaded Java Virtual Machines. The settings here determine how, when, and where the installer deploys a bundled/downloaded VM to support the application you are installing.

Table 7-20 • Bundled/Downloaded Virtual Machine Controls

Control	Description
Install Bundled/Downloaded Virtual Machine	<p>Determines whether the installer deploys a bundled VM for use by the application you are installing.</p> <p>Enable Install Bundled/Downloaded Virtual Machine to always deploy the bundled/downloaded VM when a bundled VM is present. Disable Install Bundled/Downloaded Virtual Machine to never deploy a bundled VM.</p> <p>Set the conditions under which your installer deploys the bundled/downloaded VM. Choose from the following options:</p> <ul style="list-style-type: none">• Only when installing a LaunchAnywhere—Deploys the bundled/downloaded VM (when a bundled VM is present) only if one or more LaunchAnywhere launchers must be installed. Otherwise, the bundled/downloaded VM is not installed.• Only when a compatible VM is not found in the system—Deploys the bundled/downloaded VM unless the installer locates a valid VM (based on the criteria specified in VM Search Settings).• Do not remove bundled/downloaded VM when uninstalling—Prevents the uninstaller from removing the bundled/downloaded VM.
VM Install Folder	<p>Choose the install location (or parent directory) for the bundled VM from a list of Magic Folders.</p> <p>By default, the installer stores the bundled VM in a JRE subdirectory of the Magic Folder you chose.</p>

Additional Classpath Settings

You can add additional classpaths to your installation project by embedding them (adding the resource as part of the payload, using custom codes along with dependencies, or using the **Install from Manifest** action).

If you want to add additional resources/classpaths *without* embedding them, you can use the fields in the **Additional Classpath Settings** area.

- **Resolved relative to installer's location**—The paths are resolved relative to the installer's location during runtime.
- **Uniform to all modes**—The resolved paths are uniform to all modes: installation, uninstallation, maintenance mode, and instance management.

Use the following controls to specify addition classpath settings:

Table 7-21 • Additional Classpath Settings

Control	Description
Classpath List	List of classpaths added to the installation project.
Add	Click to add a classpath to the project. The Add Classpaths dialog box opens, prompting you to enter the name of the additional classpath to set.
Remove	Click to remove the selected classpath from the list.

Search Panel Settings Tab

Use the **Search Panel Settings** tab of the **Project > JVM Settings** subtask to specify criteria to use to find valid VMs for the application you are installing. The **Search Panel Settings** tab includes three subtabs:

- [General Subtab](#)
- [Windows Subtab](#)
- [UNIX Subtab](#)

General Subtab

The **VM Search Settings** area of the **General** subtab determine the criteria the installer uses to find valid VMs for the software your project installs.

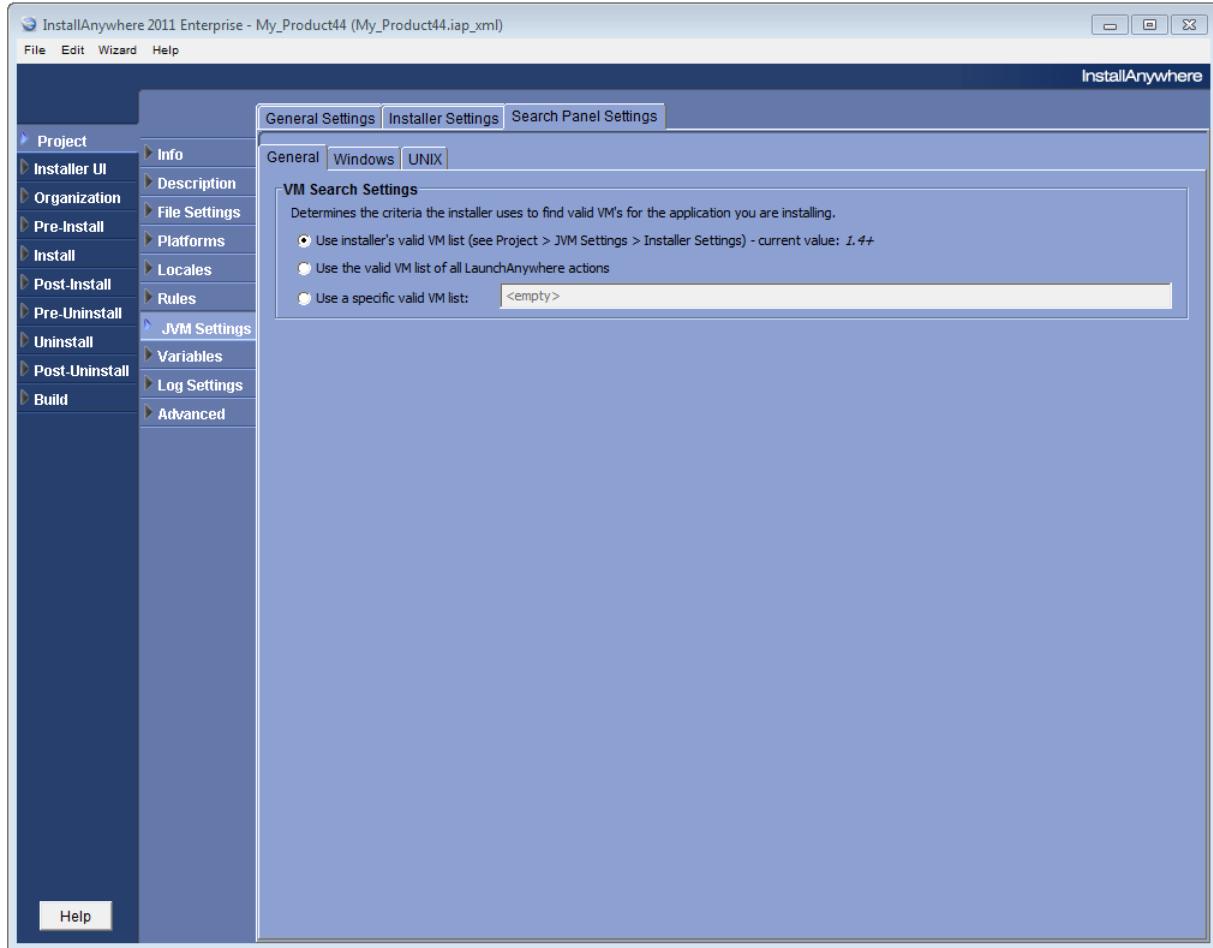


Figure 7-13: General Subtab

The **General** subtab includes the following controls in the **VM Search Settings** area:

Table 7-22 • General Subtab Controls

Control	Description
Use installer's valid VM list	Uses the criteria specified in the Installer Valid VM List on the Project > JVM Settings > Installer Settings subtask.

Table 7-22 • General Subtab Controls

Control	Description
Use the valid VM list of all LaunchAnywhere actions	<p>Uses the lax.nl.valid.vm.list value for your application's launcher as the criteria for the VM search.</p>  <p>Note • If more than one launcher exists in the installer, the installer combines the values for all launchers, searching for a VM that meets all VM criteria.</p>
Use a specific valid VM list	<p>Specify the VM criteria you want the installer to use.</p>  <p>Note • See About Java VM Selection Criteria for valid VM criteria.</p>

Windows Subtab

The **Windows JVM Search Paths** area includes the following subtabs:

- [Search Paths Tab](#)
- [Java Executable Patterns Tab](#)

Search Paths Tab

The **Windows JVM Search Paths** area of the **Windows** subtab provides a list of search paths and search patterns for your project's Windows installers to use to locate VMs.



Tip • These settings provide parameters for the VM search, while the **VM Search Settings** on the [General Subtab](#) specify which VMs are valid for the software deployed by your project.



Note • These settings apply only when the installer searches for a VM on the target system and when the installer attempts to provide a list of VMs on the **Choose Java VM** panel. They do not apply to any searches independently performed by LaunchAnywhere launchers in your project.

On the **Search Paths** tab, you can add, remove, or move entries in the **Search Paths** list.

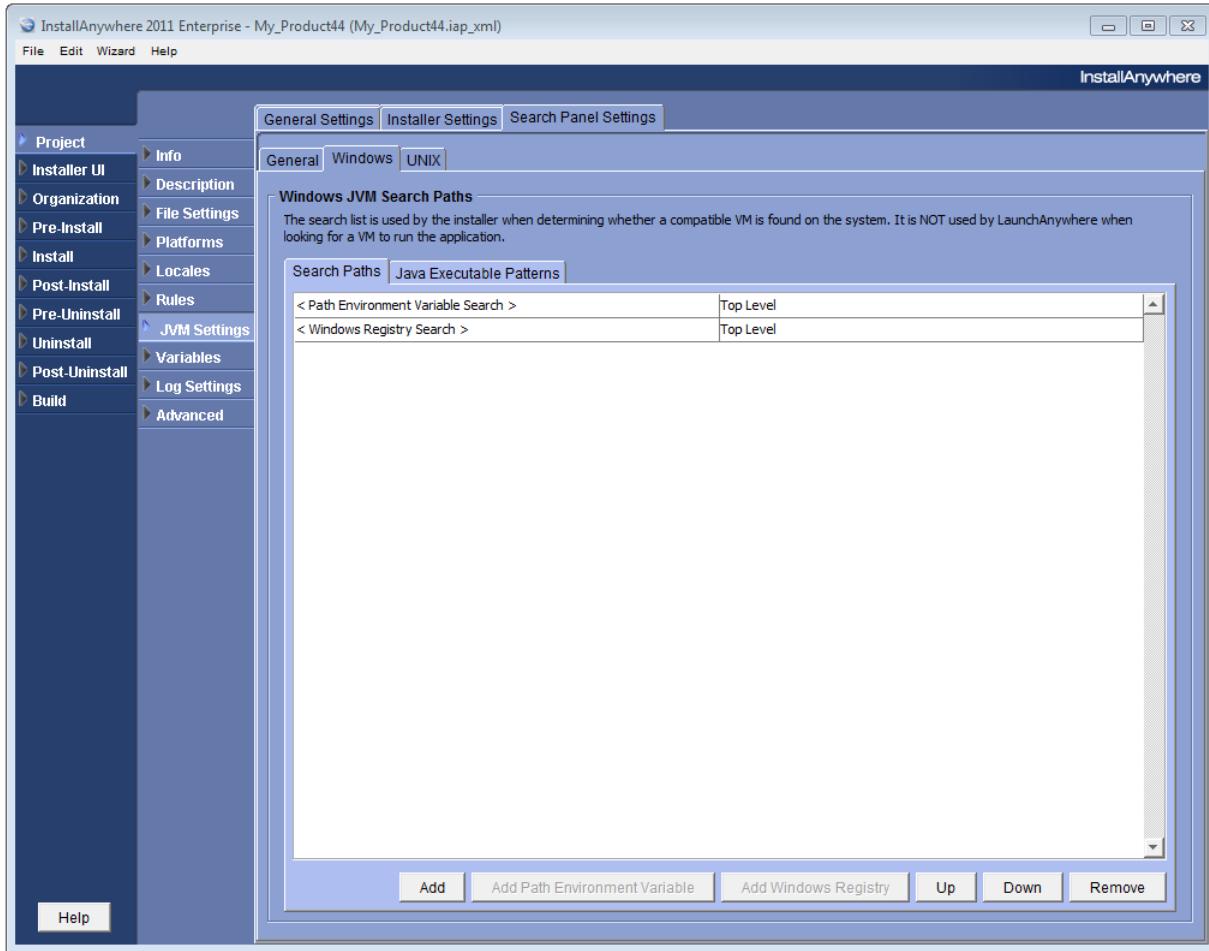


Figure 7-14: Search Paths Subtab of Windows Tab

The **Search Paths** tab includes the following controls:

Table 7-23 • Search Paths Tab Controls

Control	Description
Add	<p>Click to insert a new entry in the Search Paths list.</p> <p>In the first column of the new entry, enter a path to search for Java virtual machines.</p> <p>In the second column, select one of the following to specify how deep the installer searches subdirectories of the given path:</p> <ul style="list-style-type: none"> • Top Level—Search only the specified directory: no subdirectories. • First Level—Search only the specified directory plus its first-level subdirectories. • All Levels—Search the specified directory and all subdirectories.

Table 7-23 • Search Paths Tab Controls

Control	Description
Add Path Environment Variable	Click to insert <Path Environment Variable Search> in the Search Paths list. This entry causes the installer's launcher to search the paths present in the target system's Path environment variable for valid VMs.
Add Windows Registry	Click to insert <Windows Registry Search> in the Search Paths list. This entry causes the installer's launcher to search the paths (JavaHome) in the target system's Windows registry for valid VMs.
Up	Click to move the currently selected entry in the Search Paths list up one position.
Down	Click to move the currently selected entry in the Search Paths list down one position.
Remove	Click to delete the currently selected entry from the Search Paths list.



Note • *InstallAnywhere does not support the use of InstallAnywhere variables or Magic Folders in path entries.*

Java Executable Patterns Tab

On the **Java Executable Patterns** tab, you can add, remove, or change the patterns the installer uses to identify VMs.

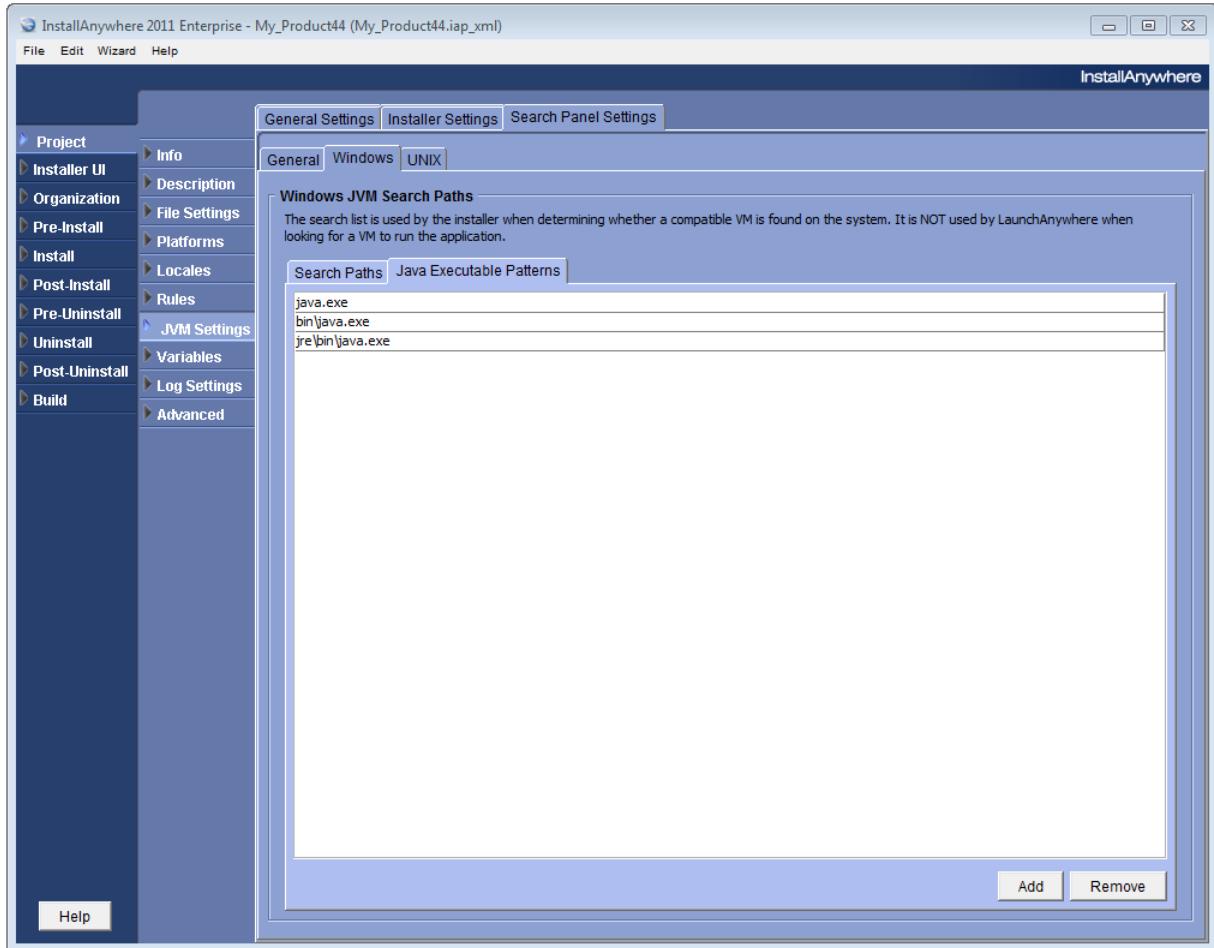


Figure 7-15: Java Executable Patterns Subtab of Windows Tab

The **Java Executable Patterns** tab includes the following controls:

Table 7-24 • Java Executable Patterns Tab Controls

Control	Description
Path List	List of defined Java Executable patterns.
Add	Click to insert a new pattern in the Java Executable Patterns list. Type the string you want your project's installers to use to identify VMs on the target machine. To change an existing entry, double-click the entry and type your edits.
Remove	Click to remove the currently selected entry from the Java Executable Patterns list.

UNIX Subtab

The **Unix JVM Search Paths** area of the **UNIX** subtab is a list used by the installer when determining whether a compatible VM is found on the system.

The **Unix JVM Search Paths** area includes the following subtabs:

- [Search Paths Tab](#)
- [Java Executable Patterns Tab](#)

Search Paths Tab

On the **Search Paths** tab, provide a list of search paths and search patterns your project's Unix installers use to locate VMs.



Tip • These settings provide parameters for the VM search, while the **VM Search Settings** on the [General Subtab](#) specify which VMs are valid for the software deployed by your project.



Note • These settings apply only when the installer searches for a VM on the target system and when the installer attempts to provide a list of VMs on the **Choose Java VM** panel. They do not apply to any searches independently performed by LaunchAnywhere launchers in your project.

On the **Search Paths** tab, you can add, remove, or move entries in the **Search Paths** list.

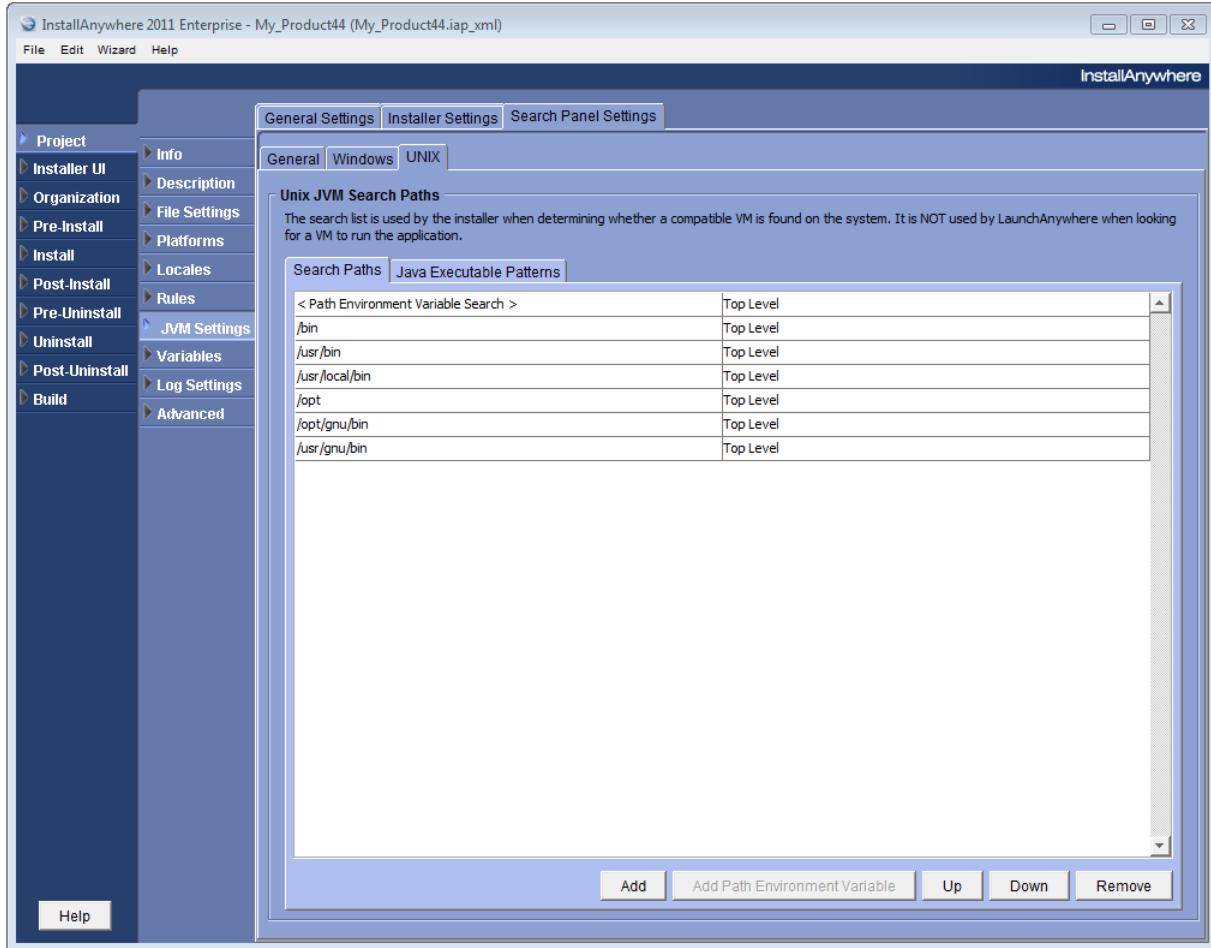


Figure 7-16: Search Paths Subtab of UNIX Tab

The **Search Paths** tab includes the following controls:

Table 7-25 • Search Path Tab Controls

Control	Description
Add	<p>Inserts a new entry in the Search Paths list. Type a path to search for Java virtual machines. Choose from Top Level, First Level, or All Levels to specify how deep the installer searches subdirectories of the given path:</p> <ul style="list-style-type: none"> • Top Level—Search only the specified directory: no subdirectories. • First Level—Search only the specified directory plus its first-level subdirectories. • All Levels—Search the specified directory and all subdirectories.

Table 7-25 • Search Path Tab Controls

Control	Description
Add Path Environment Variable	Inserts <Path Environment Variable Search> in the Search Paths list. This entry causes the installer's launcher to search the paths present in the target system's Path environment variable for valid VMs.
Up	Moves the currently selected entry in the Search Paths list up one position.
Down	Moved the currently selected entry in the Search Paths list down one position.
Remove	Deletes the currently selected entry from the Search Paths list.



Note • *InstallAnywhere does not support the use of InstallAnywhere variables or Magic Folders in path entries.*

Java Executable Patterns Tab

On the **Java Executable Patterns** tab, you can add, remove, or change the patterns the installer uses to identify VMs.

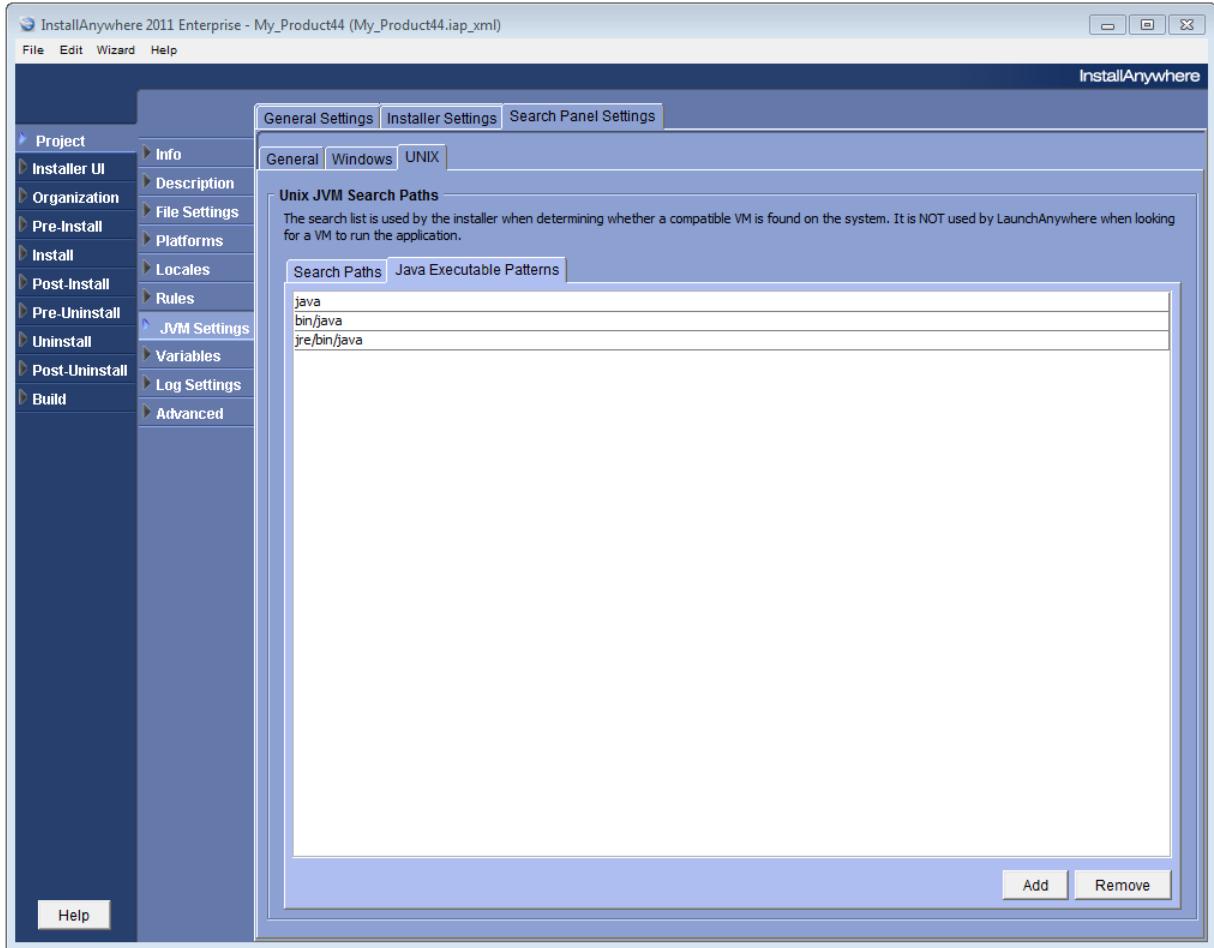


Figure 7-17: Java Executable Patterns Subtab of UNIX Tab

The **Java Executable Patterns** tab includes the following controls:

Table 7-26 • Java Executable Patterns Tab Controls

Control	Description
Path List	List of defined Java Executable patterns.
Add	Click to insert a new pattern in the Java Executable Patterns list. Type the string you want your project's installers to use to identify VMs on the target machine. To change an existing entry, double-click the entry and type your edits.
Remove	Click to remove the currently selected entry from the Java Executable Patterns list.

Variables

Use the **Variables** subtask to advertise variables for merge modules, list variables to encrypt in or exclude from the response files, set security options, and view a list of all defined variables.

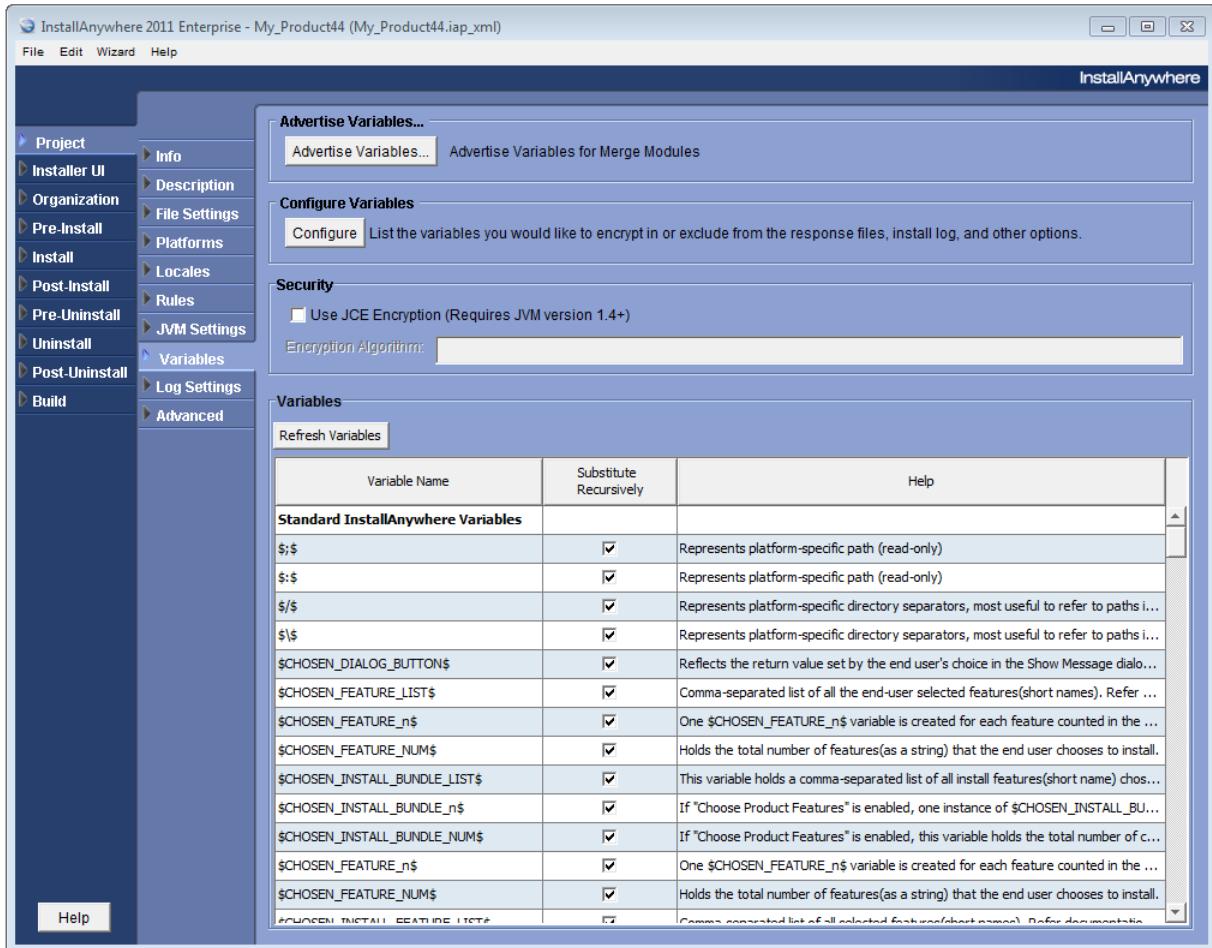


Figure 7-18: Project > Variables Subtask

The **Project > Variables** subtask includes the following areas:

- [Advertise Variables](#)
- [Configure Variables](#)
- [Security](#)
- [Variables](#)

Advertise Variables

In the **Advertise Variables** area, you advertise variables for merge modules.

Table 7-27 • Advertise Variables Controls

Control	Description
Advertise Variables	<p>Click to open the Advertise Variables dialog box where you can add variables, set default values, and add comments. Advertised variables are used by Dynamic Merge Modules and the Install Merge Module action.</p> <p>On this dialog box, you can specify which variables in the Merge Module can be set by the parent installer and which variables in the parent installer can be set by the Merge Module.</p>  <p>Note • For more information, see Advertise Variables Dialog Box.</p>

Configure Variables

Use the following controls to list the variables you would like to encrypt in or exclude from the response files, install log, and other options.

Table 7-28 • Configure Variables Controls

Control	Description
Configure	<p>Click to open the Configure Variables dialog box, where you can add or remove configuration settings for InstallAnywhere variables. These settings allow you to suppress the appearance of variables, in the response file and the install log, or encrypt variables in all output. Possible configuration settings include</p> <ul style="list-style-type: none">• Exclude Variable Entirely• Exclude Value Only• Encrypt Variable Value  <p>Note • For more information on excluding variables, see Generating Response Files.</p>  <p>Note • For more information on encrypting variable values, see Encrypting Variable Values.</p>

Security

Use the following controls to customize security settings for encryption.

Table 7-29 • Security Controls

Control	Description
Use JCE Encryption	Activates or deactivates Java Cryptography Extension (JCE) encryption. InstallAnywhere includes basic encryption to obfuscate encrypted elements. Checking Use JCE Encryption substitutes the JVM's JCE encryption for InstallAnywhere's basic encryption. If JCE encryption is not available (as in pre-1.4 JVMs), InstallAnywhere defaults to its own encryption method.
Encryption Algorithm	Type the algorithm name you want to use with JCE encryption.  Note • Using JCE encryption, along with a FIPS-compliant JCE library and supported algorithm, you can ensure your installers are FIPS 140-2 compliant. See Making a VM Pack FIPS-Compliant for more information.



Caution • Encryption is not fully supported for Merge Modules. Sensitive information that requires encryption should not be included in a Merge Module.

Variables

The **Variables** area of the **Project > Variables** subtask provides a scrollable list of all variables defined in the installation project.

Variable Name	Substitute Recursively	Help
Standard InstallAnywhere Variables		
\$;\$	<input checked="" type="checkbox"/>	Represents platform-specific path (read-only)
\$:\$	<input checked="" type="checkbox"/>	Represents platform-specific path (read-only)
\$/	<input checked="" type="checkbox"/>	Represents platform-specific directory separators, most useful to refer to paths i...
\$_	<input checked="" type="checkbox"/>	Represents platform-specific directory separators, most useful to refer to paths i...
\$CHOSEN_DIALOG_BUTTON\$	<input checked="" type="checkbox"/>	Reflects the return value set by the end user's choice in the Show Message dialog...
\$CHOSEN_FEATURE_LIST\$	<input checked="" type="checkbox"/>	Comma-separated list of all the end-user selected features(short names). Refer ...
\$CHOSEN_FEATURE_n\$	<input checked="" type="checkbox"/>	One \$CHOSEN_FEATURE_n\$ variable is created for each feature counted in the ...
\$CHOSEN_FEATURE_NUM\$	<input checked="" type="checkbox"/>	Holds the total number of features(as a string) that the end user chooses to install.
\$CHOSEN_INSTALL_BUNDLE_LIST\$	<input checked="" type="checkbox"/>	This variable holds a comma-separated list of all install features(short name) chos...
\$CHOSEN_INSTALL_BUNDLE_n\$	<input checked="" type="checkbox"/>	If "Choose Product Features" is enabled, one instance of \$CHOSEN_INSTALL_BU...
\$CHOSEN_INSTALL_BUNDLE_NUM\$	<input checked="" type="checkbox"/>	If "Choose Product Features" is enabled, this variable holds the total number of c...
\$CHOSEN_FEATURE_n\$	<input checked="" type="checkbox"/>	One \$CHOSEN_FEATURE_n\$ variable is created for each feature counted in the ...
\$CHOSEN_FEATURE_NUM\$	<input checked="" type="checkbox"/>	Holds the total number of features(as a string) that the end user chooses to install.
\$CHOSEN_INSTALL_FEATURE_LIST\$	<input checked="" type="checkbox"/>	Comma-separated list of all selected features(short names). Refer documentation.

Figure 7-19: Variables Area of Project > Variables Subtask

Substitute Recursively Option

Each listed variable includes a **Substitute Recursively** option, enabling you to specify that you do not want the variable to be substituted at runtime. In cases where you expect that the variable could contain multiple \$ characters—such as in a user-defined password—you would clear the selection of the **Substitute Recursively** option for a variable.

For example, suppose the installer includes a variable named \$PASSWORD\$ and the end user enters a value of **apple\$newton\$found** for that variable at runtime:

- **Substitute Recursively not selected**—The value of \$PASSWORD\$ is set to **apple\$newton\$found**.
- **Substitute Recursively selected**—The \$newton\$ portion of the **apple\$newton\$found** value is read as an unknown variable, which is replaced with a blank value, resulting in setting the value of \$PASSWORD\$ to **applefound**, which is incorrect.

Log Settings

Logging is one of the ways to record the results during runtime, reporting the success and/or failure execution details that occur. On the **Log Settings** subtask, you can enable logging during installation and also during the Maintenance Mode options of **Uninstall**, **Add Features**, **Repair Installation**, and **Remove Features**.



Edition • Logging during the Maintenance Mode phases of Add Features, Repair Installation, Remove Features, and Uninstall is only available in InstallAnywhere Enterprise Edition.

- [Log Options](#)
- [Installer Debug Output](#)

Log Options

The **Log Options** area of the **Project > Log Settings** subtask includes the following controls:

Table 7-30 • Log Options Controls

Control	Description
Enable Logging	Select this option to activate the creation of logs, which record the actions that the installer executes.

Table 7-30 • Log Options Controls

Control	Description
Mode	<p>After you have selected the Enable Logging option, select the phases for which you want to enable logging: Install, Add, Repair, Remove, and/or Uninstall.</p> <p>The default file name for the install logs is:</p> <p>\$PRODUCT_NAME\$<Mode>_<MM_DD_YYYY_HH_MM_SS>.log</p> <p>For example:</p> <p>My_Product_Install_08_13_2010_11_48_46.log My_Product_Add_08_13_2010_11_48_46.log</p>  <p>Edition • Logging during the Add, Repair, Remove, and Uninstall phases is only available in InstallAnywhere Enterprise Edition.</p>
Path	<p>For each selected phase, the default path for the log file is:</p> <p>\$USER_INSTALL_DIR\$\$\\$_\$PRODUCT_NAME\$_installation\$\Logs\$/</p> <p>You can customize this path by editing these fields. Click Restore Default Paths to reset all log paths to the default settings.</p>  <p>Note • You can also enter absolute paths as the log directory path. In the case of an invalid path, the log will be generated in the default location.</p>  <p>Note • The default log location will be resolved by the InstallAnywhere variable \$INSTALL_LOG_DESTINATION\$.</p>
Log Format	<p>Specify the format that you want the logs to use:</p> <ul style="list-style-type: none"> • Plain text format—Select to generate install logs in plain text. • XML format—Select to generate install logs in XML.
Uninstall & Debug Options	<p>Select the following options:</p> <ul style="list-style-type: none"> • Uninstall all logs—Select to remove the logs previously generated by the installer when a user chooses to install, add features, repair, remove features, or uninstall. The most recent uninstall log will not be removed. • Include debug output (stderr and stdout)—Select this option to append the debug output to the log files. If this option is selected, the stderr and stdout entries are appended to the end of the log files (for text format) or inserted as <! [CDATA[stderr/stdout entries]] > (for xml format). <p>Also, if this option is selected, the Installer Debug Output options are disabled.</p>

Installer Debug Output

Use the text boxes provided to send installer debug output (stderr and stdout) to a console (or Console.app on Mac OS X) or store it in a file. For information on interpreting the output sent to stderr and stdout, see [Reviewing Debug Information](#).



Note • If the **Include debug output (stderr and stdout)** option under **Uninstall & Debug Options** is selected, these options are disabled.



Tip • When your project includes one or more Merge Modules, the stderr and stdout settings of the parent project override the stderr and stdout settings in the Merge Modules.

Table 7-31 • Installer Debug Output Controls

Control	Description
Send stderr to	Enter one of the following to direct the stderr output: <ul style="list-style-type: none">• Console—Enter <code>console</code> to send stderr output to a console window.• File—Enter a file name to send stderr output to a file. For example, <code>installer_stderr.txt</code> causes the installer to create a file with that name, in the same location as the installer, and record the stderr output there.• Platform-neutral paths—Use <code>\$/</code> or <code>\\$</code>, Java properties (<code>\$prop.*\$</code>), or LAX environment variables (<code>\$lax.n1.env.*\$</code> or <code>\$lax.n1.env.exact_case.*\$</code>) to specify platform-neutral paths for the debug output. (LAX environment variables are not supported for Pure Java installers or Mac OS X installers.)• No output—Leave blank to discard stderr output.
Send stdout to	Enter one of the following to direct the stdout output: <ul style="list-style-type: none">• Console—Type <code>console</code> to send stdout output to a console window.• File—Type a file name to send stdout output to a file. For example, <code>installer_stdout.txt</code> causes the installer to create a file with that name, in the same location as the installer, and record the stdout output there.• Platform-neutral paths—Use <code>\$/</code> or <code>\\$</code>, Java properties (<code>\$prop.*\$</code>), or LAX environment variables (<code>\$lax.n1.env.*\$</code> or <code>\$lax.n1.env.exact_case.*\$</code>) to specify platform-neutral paths for the debug output. (LAX environment variables are not supported for Pure Java installers or Mac OS X installers.)• No output—Leave blank to discard stdout output.



Note • The output from **Project > Log Settings - Installer Debug Output** produces different results from the output of the **Output Debug Information** panel action.



Tip • The values of the **Send stderr to** and **Send stdout to** map to the LAX properties `lax.stderr.redirect` and `lax.stdout.redirect`, respectively.

Advanced

Use the **Advanced** subtask to enable Maintenance Mode support, specify the number of instances of the product per machine, manage Build Configuration Tags, and enable product rollback.

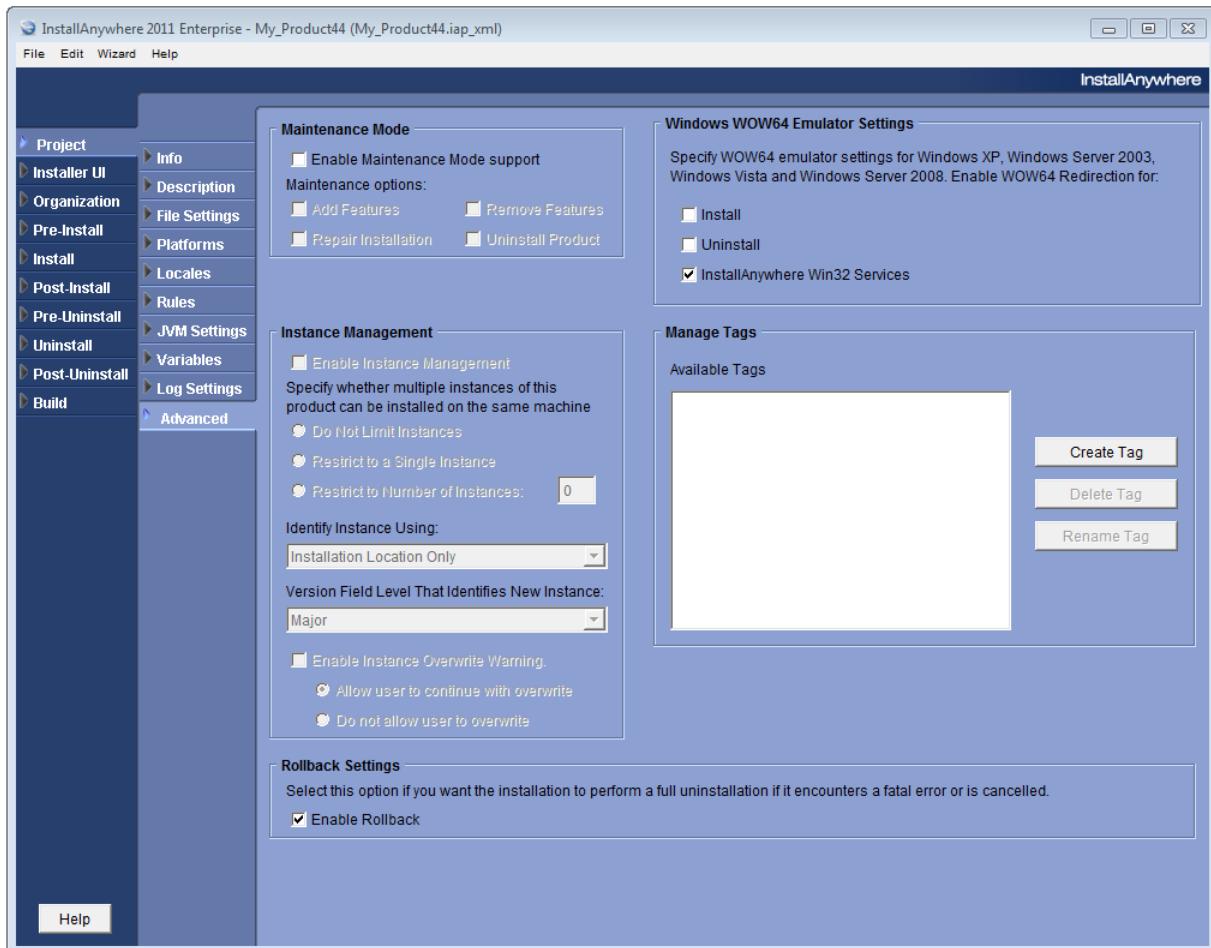


Figure 7-20: Project > Advanced Subtask

The **Project > Advanced** subtask includes controls in the following areas:

- [Maintenance Mode](#)
- [Instance Management](#)
- [Manage Tags](#)
- [Rollback Settings](#)
- [Windows WOW64 Emulator Settings](#)

Maintenance Mode



Edition • The Maintenance Mode option is available in InstallAnywhere Enterprise Edition.

You can choose to implement Maintenance Mode in an installer so that end users are able to add or remove features to previously installed products as well as repair broken installations. The Maintenance Mode options are on the **Project > Advanced** subtask.

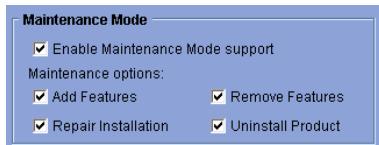


Figure 7-21: Maintenance Mode Options on the Project > Advanced Subtask

When Maintenance Mode is enabled, the end user will be able to launch Maintenance Mode by:

- Executing the **Change Installation** launcher.
- Selecting **Add or Remove Programs** from the Windows Control Panel (Microsoft Windows OS only).

When an end user launches Maintenance Mode, they will be prompted to select from the listed options, which may include **Add Features**, **Remove Features**, **Repair Product**, and **Uninstall Product**.

The **Maintenance Mode** area of the **Project > Advanced** subtask includes the following options:

Table 7-32 • Maintenance Mode Options

Option	Description
Enable Maintenance Mode Support	Select this option to enable Maintenance Mode support. When you select this option, the four Maintenance options are enabled. Select those options that you would like to include in this installation project. By default, this option is not selected.

Table 7-32 • Maintenance Mode Options (cont.)

Option	Description
Add Features	<p>Select this option to enable the end user to use Maintenance Mode to install previously uninstalled product features.</p> <p>When the end user launches Maintenance Mode and chooses Add Features, the installer will advance to the Pre-Install phase of the installation and then proceed as a regular installation.</p>  <p>Note • <i>The set of actions executed when the end user selects Add Features do not include the whole set of actions in Pre-Install, but a smaller subset for which the Check Running Mode Rule is set to Pre-Installation. See About Check Running Mode Rules.</i></p>  <p>Note • <i>For more information, see Adding a Feature User Experience.</i></p>
Remove Features	<p>Select this option to enable the end user to use Maintenance Mode to remove previously installed product features.</p>  <p>Tip • <i>The InstallAnywhere Uninstaller also provides you with the ability to selectively remove individual features.</i></p> <p>When the end user launches Maintenance Mode and chooses Remove Features, the installer will advance to the Pre-Uninstall phase of the uninstallation and then proceed with the uninstallation of the selected features.</p>  <p>Note • <i>The set of actions executed when the end user selects Remove Features do not include the whole set of actions in Pre-Uninstall, but a smaller subset for which the Check Running Mode Rule is set to Remove Features. See About Check Running Mode Rules.</i></p>  <p>Note • <i>For more information, see Removing a Feature User Experience.</i></p>
Repair Installation	<p>Select this option to enable the end user to use Maintenance Mode to repair a previously installed product.</p> <p>When the end user launches Maintenance Mode and chooses Repair Product, all of those actions in the Pre-Install phase with a Check Running Mode Rule set to Repair Installation will be executed (not just those actions that exist in the Repair Installation Action Group). See About Check Running Mode Rules.</p>  <p>Note • <i>For more information, see Repairing a Product User Experience.</i></p>

Table 7-32 • Maintenance Mode Options (cont.)

Option	Description
Uninstall Product	<p>Select this option to enable the end user to use Maintenance Mode to uninstall a previously installed product.</p> <p></p> <p>Tip • <i>The InstallAnywhere Uninstaller also provides you with the ability to uninstall a product.</i></p> <p>When the end user launches Maintenance Mode and chooses Uninstall Product, the installer will advance to the Pre-Uninstall phase of the uninstallation and proceed with the uninstallation of the product.</p> <p></p> <p>Note • <i>For more information, see Uninstalling a Product User Experience.</i></p>

Instance Management



Edition • This feature is included with InstallAnywhere Enterprise Edition.

InstallAnywhere's Maintenance Mode support includes an **Instance Management** feature that enables you to specify whether multiple instances of a product can be installed on the same machine.

If an installation includes both Maintenance Mode support and support for multiple instances of the product on the same machine, when an end user launches Maintenance Mode, they are able to specify which instance of the product they want to perform maintenance on.



Note • *The instance count is only as secure as the security of the InstallAnywhere product registry file. For more information, see [Product Registry](#).*

The **Instance Management** options are on the **Project > Advanced** subtask of the Advanced Designer.

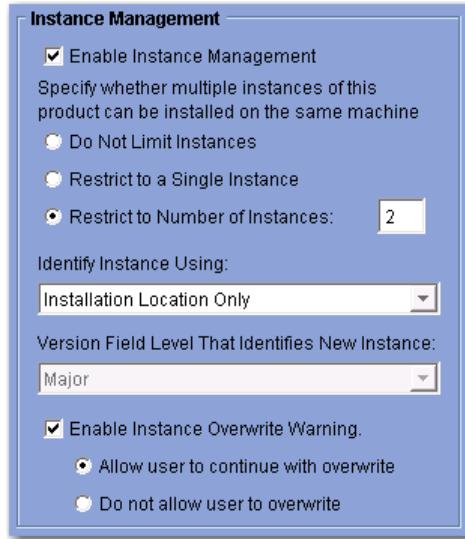


Figure 7-22: Instance Management Options on Project > Advanced Subtask



Important • The **Enable Instance Management** option is only enabled if the **Enable Maintenance Mode support** option is selected.

Instance Management has the following options:

Table 7-33 • Instance Management Options

Option	Description
Enable Instance Management	Select this option to enable the Instance Management options: Do Not Limit Instances , Restrict to a Single Instance , and Restrict to Number of Instances .

Table 7-33 • Instance Management Options (cont.)

Option	Description
Do not Limit Instances	<p>Select this option to enable end users to install an unlimited number of instances of this product on a machine.</p> <p>End users may want to install multiple instances on the same machine if they want to configure each instance differently (each with different features, configuration settings, environments) and want to launch each instance independently (such as launching multiple instances on WebSphere/Tomcat using different JVM versions).</p> <p>If an installer was created with the Do not limit instances option selected, the following occurs:</p> <ul style="list-style-type: none"> When an end user attempts to install a second instance of that product, the Manage Instances dialog box opens, prompting the user to specify that they want to Install a New Instance (rather than Modify an Existing Instance). When an end user has installed multiple instances of a product on a machine and then launches Maintenance Mode, the Manage Instances dialog box opens, prompting the user to select the instance that they want to modify.  <p>Note • For more information, see Maintenance Mode User Experience on a Machine With Multiple Installed Instances.</p>
Restrict to a Single Instance	<p>Select this option to restrict end users to be able to install only one instance of the product on a machine.</p> <p>By selecting the Restrict to a Single Instance option, you are preventing the end user from being able to do either of the following:</p> <ul style="list-style-type: none"> Installing additional instances of the product on the same machine. Overwriting an existing instance of the product. <p>If an installer was created with the Restrict to a Single Instance option selected, when an end user attempts to install a second instance of that product, the Manage Instances dialog box opens, prompting the user to modify the existing instance, rather than to install a new instance.</p>
Restrict to Number of Instances	Select this option and enter a number in the text box to restrict end users to be able to install only a specified number of instances of the product on a machine.

Table 7-33 • Instance Management Options (cont.)

Option	Description
Identify Instance Using	<p>Select one of the following options to specify how to identify an instance of an application:</p> <ul style="list-style-type: none"> • Installation Location Only—Identify an instance by examining the installation location. No filters are applied to the detected instances. • Installation Location and Version—Identify an instance by examining the installation location and the application version of the installed application. When this option is selected, the Version Field Level That Identifies New Instance option is enabled, which enables you to identify what level of version change constitutes a new version.
Version Field Level That Identifies New Instance	<p>When you select Installation Location and Version from the Identify Instance Using option, this option is enabled. You are prompted to specify what level of version change constitutes a new version by selecting one of the following options:</p> <ul style="list-style-type: none"> • Major • Minor • Revision • Subrevision <p></p> <p>Note • Versions are conventionally represented in the following format: [Major].[Minor].[Revision].[Subrevision], such as: 1.0.2.1047.</p> <p>For more information on how this option is used to determine which instances are listed on the Manage Instances dialog box, see Identifying a New Instance Based on Version Field Level.</p> <p></p> <p>Note • Reinstalling an application to the same location as an existing application would result in only one instance, irrespective of the version.</p> <p></p> <p>Note • If you choose the Installation Location and Version option from the Identify Instance Using list, the installation does not recognize those instances which are from a more recent version, because an installer of a previous version might not be able to successfully manage an instance of the installer of a more recent version. For example, a Version 2.0.0.0 installer will not consider Version 3.x instances, but will consider all Version 1.x instances and Version 2.0.0.0 instances.</p>

Table 7-33 • Instance Management Options (cont.)

Option	Description
Enable Instance Overwrite Warning	<p>Select this option to warn the end user when they are about to overwrite an existing instance of an application and specify whether they will be permitted to overwrite the instance. After you select this option, select one of the following options:</p> <ul style="list-style-type: none">• Allow user to continue with overwrite—Permit end user to overwrite an existing instance of an application.• Do not allow user to overwrite—Do not permit end user to overwrite an existing instance of an application. Instead, prompt end user to change the installation location in order to continue with the installation. <p></p> <p>Note • If this option is selected, you are required to include a Panel: Choose Install Folder action in the Pre-Install task.</p>



Note • When using the Instance Management feature, note the following:

- Instance Management uses the global/local zeroG registries. If these directories are altered, then the Instance Management feature will not work properly.
- The **InstallAnywhere Uninstall Component**, which is a standard component in an InstallAnywhere project, should have the **Key File** set to the Uninstaller executable file, which is the default setting. However, if the Key File is modified, then the Instance Management feature may not be in a position to manage instances because it would be unable to find the Uninstaller.

The **Key File** for the **InstallAnywhere Uninstall Component** is set on the **Properties > Component** subtab of the **Organization > Components** subtask.

Identifying a New Instance Based on Version Field Level

The following table demonstrates how the selection made in the **Version Field Level That Determines New Instance** list as well as the comparison of the installer vs. installed instance versions determine whether an existing installed instance is found on the target system. In this table, an instance is found when the listed conditions are met.

Table 7-34 • Conditions That Identify an Instance Based on Version Field Level

Selected Version Field Level	Major Version	Minor Version	Revision Version	Entire Version
[Any]				Identical
Major	Installer > Instance			
	Installer = Instance			Installer > Instance
Minor	Installer > Instance			
	Installer = Instance	Installer > Instance		
	Installer = Instance	Installer = Instance		Installer > Instance
Revision	Installer > Instance			
	Installer = Instance	Installer > Instance		
	Installer = Instance	Installer = Instance	Installer > Instance	
	Installer = Instance	Installer = Instance	Installer = Instance	Installer > Instance
Subrevision				Installer > Instance

Here are a few examples:

Table 7-35 • Examples of Identifying a New Instance Based on Version Field Level

Selected Version Field Level	Installer Version	Instance Version	Result
Major	2.9.0.1	2.8.7.3	Instance will be listed on the Manage Instances dialog box because all of the following conditions are true: 1. Major is selected 2. Installer Major version (2) is equal to the Instance Major version (2) 3. Installer entire version (2.9.0.1) is greater than entire Instance version (2.8.7.3)
Minor	2.7.3.0	2.5.1.0	Instance will be listed on the Manage Instances dialog box because all of the following conditions are true: 1. Minor is selected 2. Installer Major version (2) is equal to the Instance Major version (2) 3. Installer Minor version (7) is greater than Instance Minor version (5)
Revision	3.5.2.5	3.5.4.8	Instance will not be listed on the Manage Instances dialog box because condition 4 is false: 1. Revision is selected 2. Installer Major version (3) is equal to the Instance Major version (3) 3. Installer Minor version (5) is equal to the Instance Minor version (5) 4. Installer Revision version (2) is greater than Instance Revision version (4) [FALSE]

Manage Tags



Edition • This feature is included with InstallAnywhere Enterprise Edition.

You can create new project Tags in the **Manage Tags** area of the **Project > Advanced** subtask.

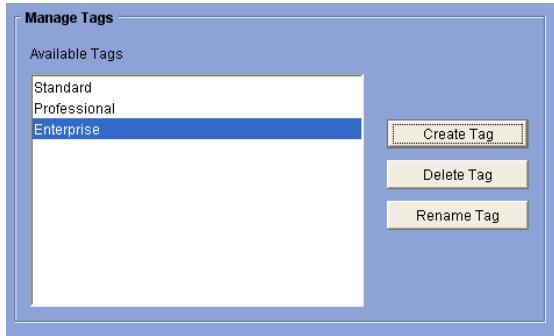


Figure 7-23: Manage Tags Settings on the Project > Advanced Subtask

The **Manage Tags** area has the following controls:

Table 7-36 • Manage Tags Area of the Project > Advanced Subtask

Control	Description
Available Tags	Lists all Tags defined in the project.
Create Tag	Click to open the Manage Tags dialog box, where you can enter the name for a new Tag. If you want this new Tag to be automatically associated with all existing project elements, select the Associate with all project elements option on the Manage Tags dialog box.  Note • If you select the Associate with all project elements option when creating a new Tag, that Tag is automatically associated with all existing project elements, but it is not automatically associated with additional project elements that are added to the project from that point forward.
Delete Tag	Click to delete the selected Tag. You will be prompted to confirm the deletion.
Rename Tag	Click to open the Rename Tag dialog box, where you can enter a new name for the selected Tag.

Rollback Settings

If an end user cancels an installation before it has completed, or if a fatal error occurs during the installation, the result can be an incomplete, corrupt application. To avoid this problem, you can enable the **Rollback** feature.

To enable installation rollback, select the **Enable Rollback** option on the **Project > Advanced** subtask.



Figure 7-24: Rollback Settings on the Project > Advanced Subtask

The **Rollback Settings** area of the **Project > Advanced** subtask includes the following controls:

Table 7-37 • Rollback Settings Area of the Project > Advanced Subtask

Control	Description
Enable Rollback	Select this option to enable the installer to perform a complete uninstall if the installation encounters a fatal error or is cancelled. This option is selected by default for all new projects.

If you select this option, and the end user cancels an installation or a fatal error occurs, the installer will automatically revert what has been altered or added to the system, including deleting all temporary files. All processes related to the installation will be shut down properly. The installation log will contain all status messages, including the action that triggered the rollback.



Note • If **Enable Rollback** is not selected and an end user cancels an installation or a fatal error occurs, none of the folders or files that had already been installed by the installer will be deleted.

Note the following additional information regarding the Rollback feature:

- [Additional Rollback Options](#)
- [About Rollback and Custom Code](#)
- [Rollback Handlers](#)

Additional Rollback Options

If you select the **Enable Rollback** option, you can also specify the following options to fine tune how installation rollbacks are performed:

- **Setting Rollback options on project elements in the Install task**—You can fine tune how the installer treats individual project elements during a rollback and can specify which project elements would trigger a rollback if the installation of that element fails. On the **Rollback** subtab of action customizers in the **Install** task, you can specify rollback options on a file-by-file basis. See [Setting Installation Rollback Options](#).
- **Adding Trigger Rollback actions**—You can also choose to add a **Trigger Rollback** action to specifically initiate a rollback if a certain rule or condition is met. See the [Trigger Rollback](#) action.

About Rollback and Custom Code

Regarding Rollback and custom code, note the following:

- If you want to trigger a rollback from custom code, you need to throw an exception of type `FatalInstallException`.
- If you want to execute some custom code during a rollback, you can include the custom code in the **Uninstall** phase and add a rule to execute it only when the `$IA_ROLLBACK$` variable is set to TRUE.

Rollback Handlers

In previous releases, InstallAnywhere had support for custom code that provided a Rollback Handler that would be called if an end user cancelled his installation. Starting with InstallAnywhere 2010, this new Rollback functionality has been added. Therefore, the Rollback Handlers custom code will be executed only when the **Enable Rollback** option on the **Project > Advanced** subtask is not selected.

Windows WOW64 Emulator Settings

WOW, which stands for Windows on Windows, is an emulation technology—available on Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008—which enables 32-bit Windows-based applications to run seamlessly on 64-bit Windows. When WOW64 emulation is on, the system isolates 32-bit applications from 64-bit applications, which prevents file and registry collisions.



Note • *WOW is not available on Windows 7 or Windows Server 2008 R2.*

Having WOW64 emulation *enabled* during Install or Uninstall could cause problems. For example, 64-bit Magic Folder operations would automatically be redirected to 32-bit folders, causing 64-bit applications to be installed incorrectly on 64-bit machines.

However, having WOW64 emulation *disabled* during Install and Uninstall can also cause issues, such as Custom Code which makes use of the InstallAnywhere Windows Service API not getting redirected properly.

Therefore, InstallAnywhere gives you control over when you want WOW64 emulation to be enabled. You specify Windows WOW64 Emulator settings for an InstallAnywhere project on the **Project > Advanced** subtask of the Advanced Designer.

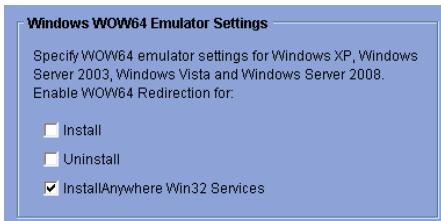


Figure 7-25: Windows WOW64 Emulator Settings

The **Windows WOW64 Emulator Settings** area has the following options:

Control	Description
Enable WOW64 Redirection for	Choose one or more of the following options to enable Windows WOW64 emulation: <ul style="list-style-type: none">• Install—Choose to enable WOW64 redirection for the Install phase.• Uninstall—Choose to enable WOW64 redirection for the Uninstall phase• InstallAnywhere Win32 Services—Choose to enable WOW64 redirection for the Custom Code that the user writes, regardless of whether this Custom Code is placed in the Install phase or the Uninstall phase. See WOW64 Custom Code API Support.

WOW64 Custom Code API Support

In addition to the WOW64 settings available on the **Project > Advanced** subtask, InstallAnywhere's API (`IAClasses.zip`) contains a new class called `WOW64` in `com.zerog.ia.platform` platform that you can use to enable or disable WOW64 emulation.

Table 7-38 • WOW64 Custom Code API Support

Option	Custom Code API
Enable WOW64 Redirection	<code>WOW64.getInstance().enableRedirection();</code>
Disable WOW64 Redirection	<code>WOW64.getInstance().disableRedirection();</code>

For example, you can disable WOW64 emulation for the `Install` phase but use Custom Code to enable it in specific situations.



Important • If you enable WOW64 redirection in custom code, then you need to remember to disable it after it is done; if you do not, the same WOW64 settings would carry forward for the rest of the session.

Installer UI

The **Installer UI** task includes the following subtasks.

Table 7-39 • Installer UI Subtasks

Subtask	Description
Look & Feel	Contains three tabs which enable developers to configure the appearance and behavior of the installer: General UI Settings, Installer Steps, and Install Progress Panel.
Billboards	Provides tools to add billboards to the installer panels.
Help	Defines general installer help—as plain text or HTML. It is also possible to define more specific help for each installer panel.

Look & Feel

The **Look & Feel** subtask of the Installer UI task contains three tabs which enable developers to configure the appearance and behavior of the installer.

These tabs allow developers to customize many graphic elements and progress panels within the installer. Within these three tabs are various Preview buttons which display the look and feel of the installer with the current settings.

Table 7-40 • Installer UI > Look & Feel Tabs

Tab	Description
General UI Settings	Sets the general look and feel of your installer.
Installer Steps	Determines the type of panel additions your project uses: a background image or a list of steps. This tab also allows you to set the type of border each panel will use around your panel addition.
Install Progress Panel	Configures settings for the images or labels on the progress pane.
Maintenance Mode Labels	Customize the text and images used on the Maintenance Mode panel.
Installer Icon	Specify a custom Windows icon file (.ico) to be used for the installer.

General UI Settings

Use the **General UI Settings** tab to set general look and feel settings for the installer.

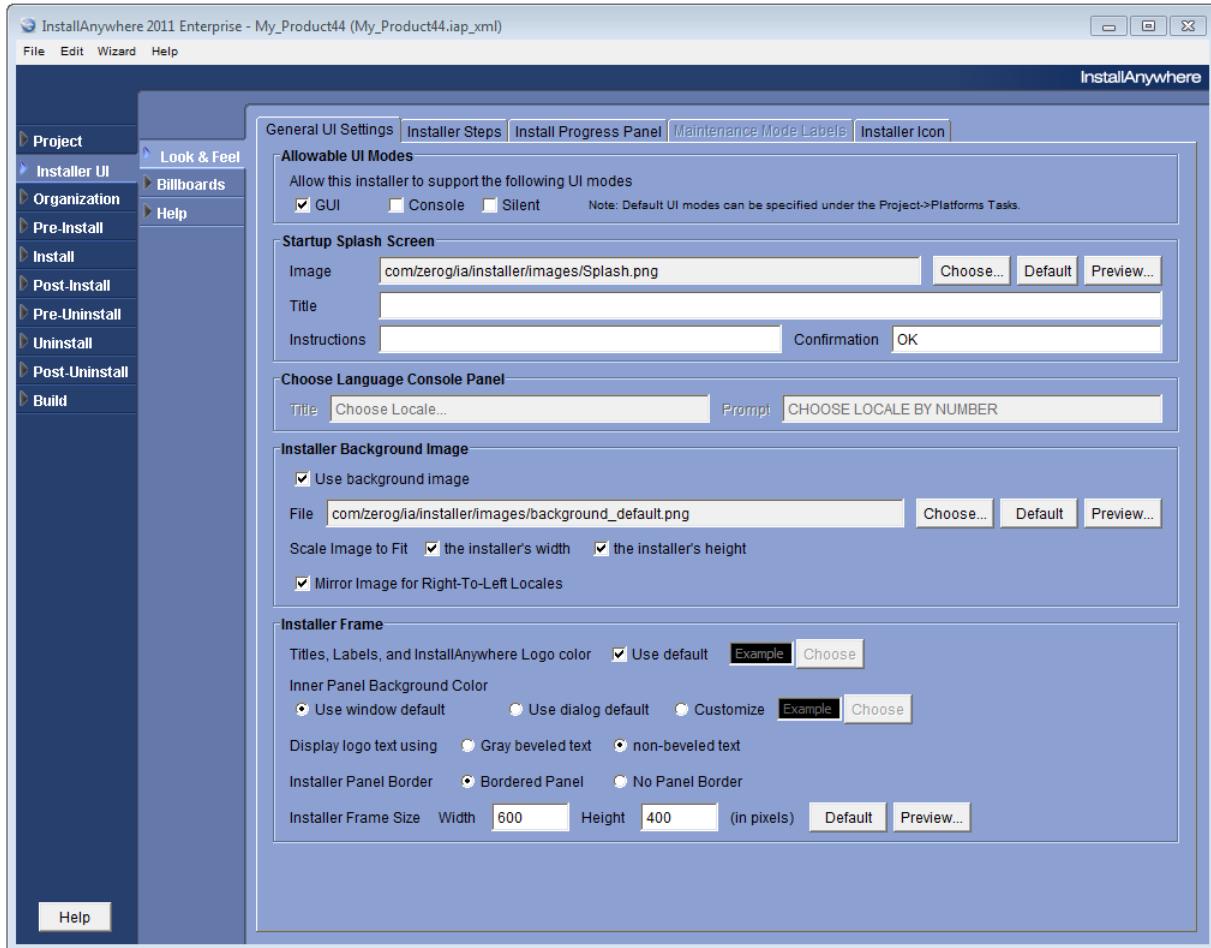


Figure 7-26: General UI Settings Tab of the Installer UI > Look & Feel Subtask

This tab allows Enterprise edition users to define which UI modes their installers will support (GUI, Console, and Silent). Graphical installers offer the ability to show a background image, such as a company logo, displayed in the installer. (The specific background image can also be defined using the **Installer Background Image** controls on this tab.) You can also choose a custom title and image for the **Startup Splash Screen**, enable or disable **Additions to GUI Installer Panels**, and define other **Installer Frame UI Settings**.

Allowable UI Modes

Select one or more supported UI modes: **GUI**, **Console**, and **Silent**.

Table 7-41 • UI Modes

Control	Description
GUI	Enables the project to create graphical (GUI) installers.
Console	Enables the project to create console installers.
Silent	Enables the installers the project creates to run in silent mode.



Note • Some important limitations for UI modes:

- Only InstallAnywhere Enterprise Edition can create installers that operate in console and silent modes. Silent and console modes are not available for installers built with InstallAnywhere Standard Edition.
- InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers will not run under those locales.

Startup Splash Screen

Graphical installers can show a customized title and image on the installer splash screen. (By default, the splash screen for internationalized installers renders an empty title bar and the default splash screen image along with a Locale selector. Installers that support only one locale do not show the title bar nor the Locale selector.)



Edition • The ability to localize the splash screen is available only in InstallAnywhere Enterprise edition.

Table 7-42 • Splash Screen Controls

Control	Description
Image	<p>The path and file name of the splash screen image.</p> <ul style="list-style-type: none"> • The splash screen image must be a PNG, GIF, or JPEG file of any size (470 pixels wide by 265 pixels tall, preferred.) • Click Choose to open the Select an Image File dialog box. • Click Default to revert a custom image to the splash screen default image: com/zerog/ia/installer/images/Splash.png • Click Preview to show a sample splash screen with the current Startup Splash Screen settings.

Table 7-42 • Splash Screen Controls

Control	Description
Title	The text that displays in the title bar of the installer splash screen. Default: no title. Title can be text, an InstallAnywhere variable, or a user-defined variable; however, only \$INSTALLER_TITLE\$ and \$PRODUCT_NAME\$ are automatically resolved in the splash screen. All other variables must have their value assigned when the installer is invoked in order to be correctly resolved in the splash screen. Variable values can be passed to the installer using the -D command-line option or an <code>installer.properties</code> file.
Instructions	The text that appears to describe how to choose a locale on the splash screen. (Default: blank.) These instructions only appear on the splash screen when the installer supports more than one locale.
Confirmation	Text for splash screen button that confirms the installer locale setting. (Default: OK) The confirmation button only appears on the splash screen when the installer supports more than one locale.



Note • See [Common Localizable Elements](#) for details on localization elements related to the startup splash screen: `splashScreenGUllimageName`, `splashScreenGUllImagePath`, `splashScreenGUllInstructions`, `splashScreenGUITitle`, `splashScreenGUIConfirm`, `splashScreenConsoleTitle`, and `splashScreenConsolePrompt`. The keys for these elements appear in the locale files as `installer.<referenceID>.<element>`. For example, `Installer.68a9edb1a601.splashScreenGUITitle`.

Choose Language Console Panel

Console mode installers that support multiple languages provide a Choose Locale console to allow users to set the language for the installer. The controls in this section allow you to customize the title and prompt on that console.

Table 7-43 • Choose Language Console Controls

Control	Description
Title	The text for the title of the Choose Locale console. (Default: Choose Locale...) The Choose Locale console only appears when the console installer supports more than one locale.

Table 7-43 • Choose Language Console Controls

Control	Description
Prompt	The text for the prompt on the Choose Locale console. The default text is: CHOOSE LOCALE BY NUMBER The Choose Locale console only appears when the console installer supports more than one locale.

Installer Background Image

The **Installer Background Image** section determines the properties of the background image. Click **Use background image** to enable the other controls in this section.

Click **Choose** to open the **Select an Image File** dialog box. When you browse to the background image you want to use and click **Open**, InstallAnywhere shows the path to that image file in the **File** text box. Click **Preview** to verify the appearance of the background image. Click **Default** to reset the image to the InstallAnywhere default background image. You can also set the image to scale to fit the installer window by width, height, or both.

Mirror Image for Right-to-Left Locales is specifically for right-to-left languages (Arabic and Hebrew). If your project builds installers for those locales, this check box causes the background image file you provide to be mirrored on its vertical axis (flipped left-right) for those locales.

Installer Frame UI Settings

The **Installer Frame UI Settings** section allows you to customize the colors, style, and size of the installer panels. Click **Default** to restore the default size for panels (600 pixels wide and 400 pixels tall). If you enter a custom size, click **Preview** to view a sample panel.

Installer Steps

Use the **Installer Steps** tab to choose the type of panel additions and provide settings for a border, a background image for installer steps, or a list of labels for those steps.

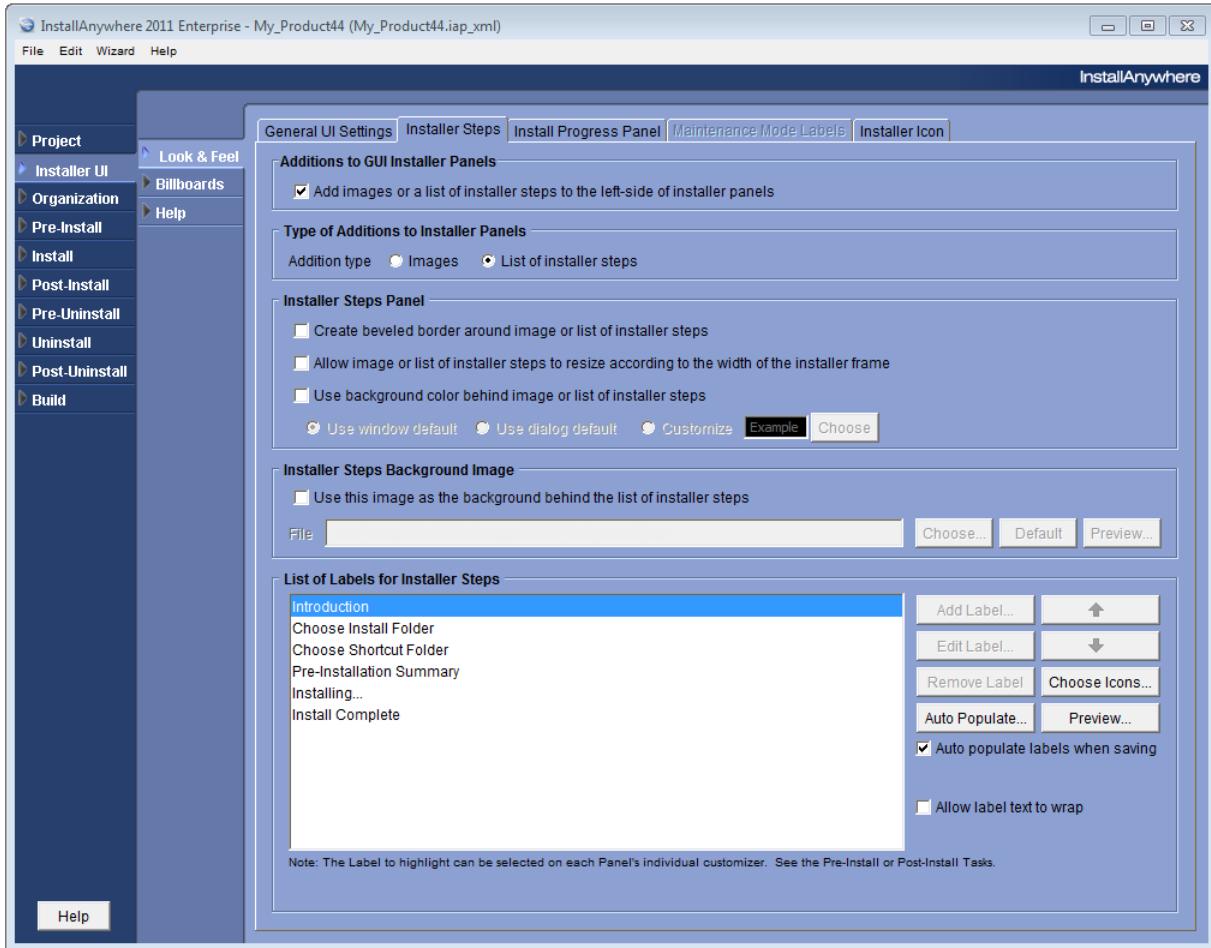


Figure 7-27: Installer Steps Tab of the Installer UI > Look & Feel Subtask

Additions to GUI Installer Panels

To have installer panels show installer progress, click **Add images or a list of installer steps to the left-side of installer panels**. This check box enables controls on the **Installer Steps** and **Install Progress Panel** tabs. If **Add images or a list of installer steps to the left-side of installer panels** is not selected, those controls are disabled.

Type of Additions to Installer Panels

There are two types of installer panel additions: **Images** and **List of Installer Steps**. Installers can use one or the other, not both.

If you choose **Images** as the type of addition, the **List of Labels for Installer Steps** controls are disabled and **Installer Steps Background Image** is replaced by **Default Installer/Uninstaller Panel Image**. Also, when **Images** is selected, the **Install Progress Panel Image** controls are enabled on the **Install Progress Panel** tab.

If you choose **List of Installer Steps** as the type of addition, you can customize panel labels in **List of Installer Steps**. Also, when **List of Installer Steps** is selected, the **Install Progress Label** controls are enabled on the **Install Progress Panel** tab.

Installer Steps Panel

The two controls in this section determine whether the Install Progress Panel has a beveled border, whether the panel resizes to fit the installer size, and what background color as the background.

Table 7-44 • Installer Steps Panel Options

Control	Description
Create Beveled Border Around Image or List of Installer Steps	Adds a beveled border around the Install Progress Panel.
Allow Image or List of Installer Steps to Resize According to the Width of the Installer Frame	Makes the size of the pane that holds the list of installer steps or image dynamically resize to fit the size of the installer frame.
Use Background Color Behind Image or List of Installer Steps	Uses the background color you choose as the background for the List of Installer Steps. Choose from <ul style="list-style-type: none"> • Use window default—Uses the default window color of the machine running the installer as the background color for the Installer Steps panel. • Use dialog default—uses the default dialog box color of the machine running the installer as the background color for the Installer Steps panel. • Customize—Uses the color you set in the Choose Color dialog box. (Click Choose to show the Choose Color dialog box.)

Installer Steps Background Image

The **Installer Steps Background Image** option enables developers to select a specific image to display in the background of the Installer Progress panel.

List of Labels for Installer Steps

The **List of Labels for Installer Steps** control the labels that appear as the installer progresses through the **Pre-Install**, **Install**, and **Post-Install** panels. It includes a list of labels and a set of buttons that enable developers to add, edit, remove, and re-order labels. These controls are only enabled when **List of Installer Steps** is the selected installer panel addition type.

- The **Auto Populate** button adds a label for every panel action added to the Pre-Install and Post-Install tasks.
- The **Choose Icons** opens the **Choose Label Icons** dialog box. Here you can alter the small graphics that precede the text labels. The default icons show an arrow for the active step, a check mark for completed steps, and a gray circle for steps pending.
- The **Auto populate labels when saving** automatically creates a list of labels that match the current set of panel actions each time the project is saved.
- The **Allow label text to wrap** avoids the possibility that any long Installer Step labels will be truncated in the installer. Instead, the label text will wrap around to the next line.
- The **Preview** shows a sample of the label icons as a set.

Install Progress Panel

Use the **Install Progress Panel** tab to configure settings for the images or labels on the progress panel. On the **Install Progress Panel** tab, either the **Install Progress Panel Image** controls or the **Install Progress Label** controls are enabled. The set of controls enabled on this tab depend on your choice for **Type of Additions to Installer Panels** on the **Installer Steps** tab.

- [Install Progress Panel Image Settings](#)
- [Install Progress Label Settings](#)
- [About Merge Module Progress](#)

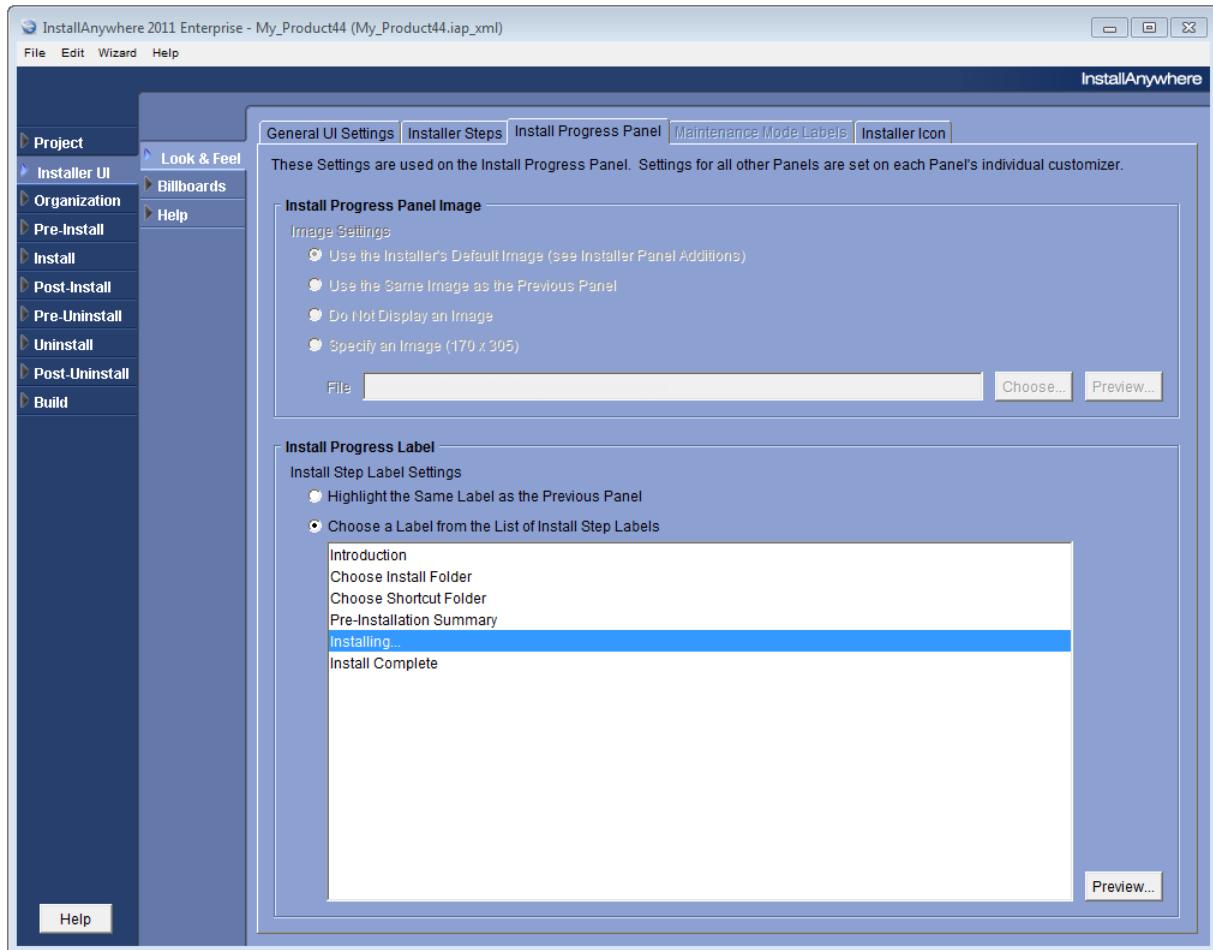


Figure 7-28: Install Progress Panel Tab of the Installer UI > Look & Feel Subtask

Install Progress Panel Image Settings

Use the **Image Settings** to determine what image your installers show in this panel. Select one of the following options:

Table 7-45 • Install Progress Panel Image Options

Option	Description
Use the Installer's Default Image	Select to use the image specified in Default Installer/Uninstaller Panel Image on the Installer Steps tab.
Use the Same Image as the Previous Panel	Select to use the image specified for the panel immediately prior to this panel.
Do not Display an Image	Select to show no additional images in this panel.
Specify an Image (170x305)	Select to use an image file you provide. To specify an image, click Choose . In the Select an Image File dialog box, navigate to the image you want to use and click Open . To verify the image appearance, click Preview .



Note • *The size of the install progress panel is 170 pixels wide by 305 pixels tall. Installer dimensions may change slightly by platform to better display text and different fonts.*

Install Progress Label Settings

Use the **Install Step Label Settings** to determine what label your installers show in the Install Progress Panel. Select one of the following options:

Table 7-46 • Install Step Label Settings Options

Option	Description
Highlight the Same Label as the Previous Panel	Select this option to keep the label the same as the previous panel during the install.
Choose a Label from the List of Install Step Labels	Select this option to display the selected label during the installation. To verify the label appearance, click Preview .

About Merge Module Progress



Edition • This feature is available in InstallAnywhere Enterprise Edition.

In releases prior to InstallAnywhere 2011, when a Merge Module was being installed, only the message [Installing Merge Module](#) was displayed on the **Install Progress** panel, with no indication of progress. Because large Merge Modules take a longer time to install, the end user was not informed of which part of the Merge Module was currently getting installed.

Starting with InstallAnywhere 2011, the installation progress of Merge Modules is merged into the main installer's progress information. Therefore, the end user can see the installation progress of a Merge Module reflected in the main installer's progress bar on the **Install Progress** panel.

There are two methods for showing the Merge Module installation progress on the **Install Progress** panel, depending upon the selection of the **Show Progress Dialog** option on the **Install Merge Module** action customizer:

- **Show Progress Dialog option selected**—If this option is selected, then the Merge Module installation progress is shown on a separate Merge Module progress bar.
- **Show Progress Dialog option not selected**—If this option is not selected, then the Merge Module installation progress is updated on the main installation project's progress bar.



Note • This feature does not support nested Merge Modules.

Maintenance Mode Labels

You can customize the text and images that are displayed on the **Maintenance Mode** panel of the Change Installation Launcher.

You can make these customizations on the **Maintenance Mode Labels Customizer** which is displayed when you select the **Maintenance Model Labels** tab of the **Installer UI > Look & Feel** subtask in the Advanced Designer.

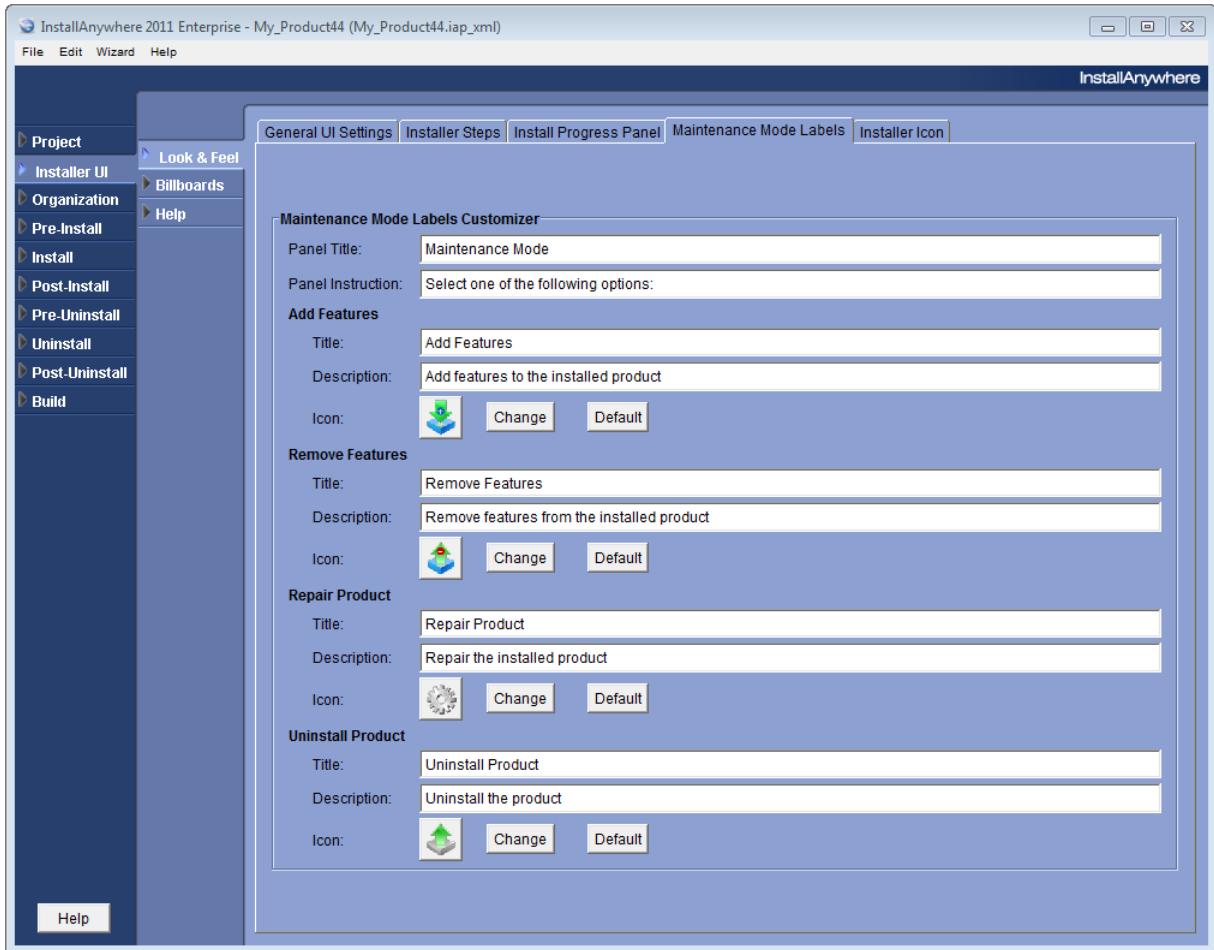


Figure 7-29: Maintenance Mode Labels Tab of the Installer UI > Look & Feel Subtask

The **Maintenance Mode Labels** tab includes the following options:

Table 7-47 • Maintenance Mode Labels Options

Option	Description
Panel Title	Enter the overall title of the Maintenance Mode panel. The default value is Maintenance Mode .

Table 7-47 • Maintenance Mode Labels Options

Option	Description
Panel Instruction	Enter the instructional sentence that appears directly under the Panel Title . The default value is Select one of the following options :
Add Features	Set the following properties to define the selection options on the Maintenance Mode panel:
Remove Features	
Repair Product	
Uninstall Product	<ul style="list-style-type: none"> • Title—Label which identifies the selection. This text appears in bold. • Description—Text which describes the selection. This text appears in plain text directly under the Title. • Icon—The default icon that represents the selection. To select a different icon, click Change and select an image (.gif, .jpg, .jpeg, or .png). To revert back to the default image, click Default.

Installer Icon

On the **Installer Icon** tab of the **Installer UI > Look & Feel** subtask, you can specify a custom Windows icon file (.ico) to be used for the installer.



Note • You can specify an icon (.ico) file for the installer on Windows platforms only.

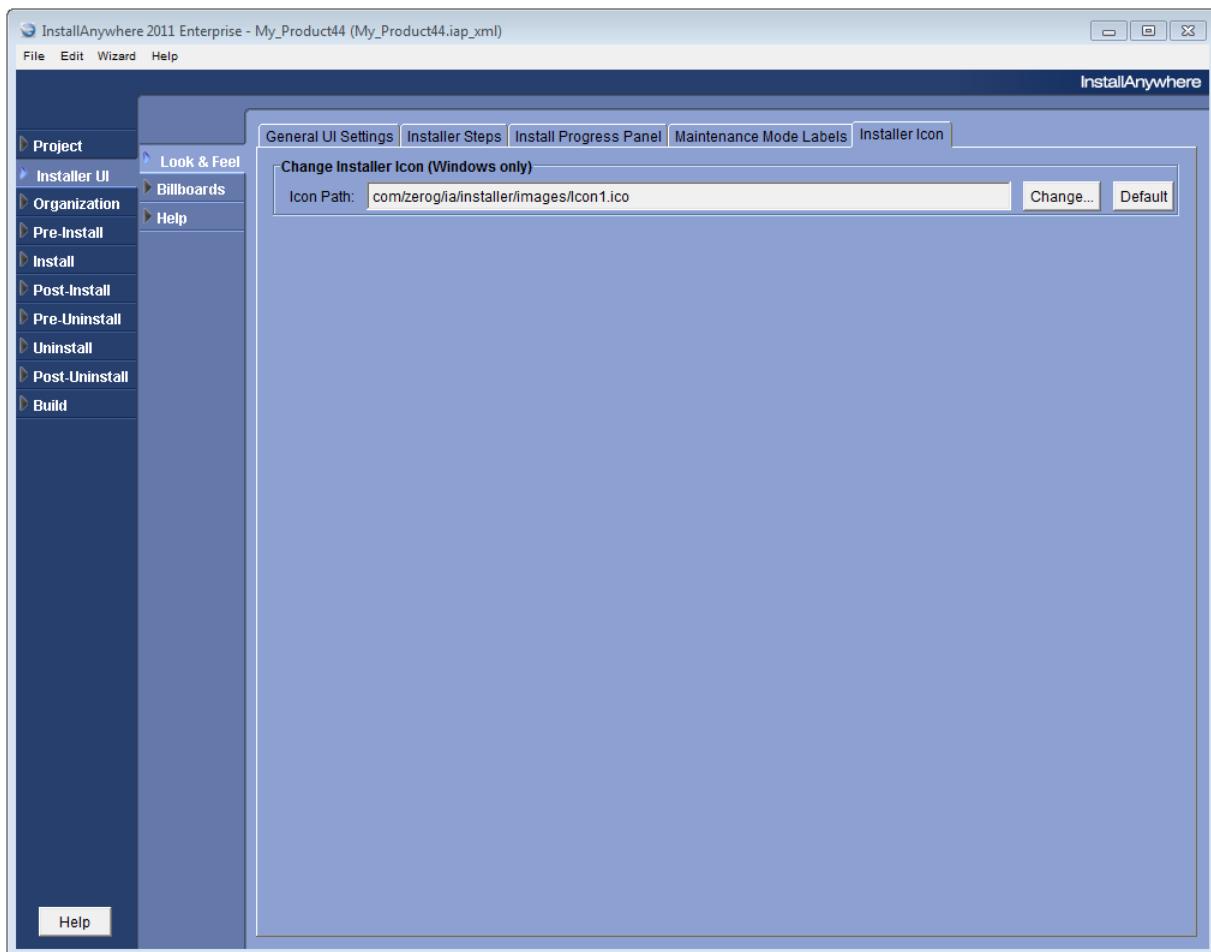


Figure 7-30: Installer Icon Tab of the Installer UI > Look & Feel Subtask

The **Installer Icon** tab includes the following controls:

Table 7-48 • Installer Icon Tab of Installer UI > Look & Feel Subtask

Control	Description
Icon Path	After you click Change and select an icon file, the icon file name and path is listed in this field.
Change	Click Change to open the Select an Icon File dialog box and select an icon (.ico) file.

Table 7-48 • Installer Icon Tab of Installer UI > Look & Feel Subtask

Control	Description
Default	Click to clear the selected icon file and return to using the default installer icon, which is: com/zerog/ia/installer/images/Icon1.ico



Note • The **Installer UI > Look & Feel** task does not display a preview of the selected .ico file.



Note • Because the installer icon change happens at build time by changing the icon of self_extractor .exe files in the InstallAnywhere installed location, it is recommended that you build one project at a time when using the **Change Installer Icon** option to change the installer icon. If you launch two instances of the same InstallAnywhere installation and build two different projects at the same time, each of which has a different icon selected on the **Installer Icon** tab, the correct icon for both projects may not be used.

Billboards

Billboards are images that appear in the large right hand pane of the installer while files are being installed. Use the **Billboards** subtask to add billboards to the installer panels.

Several billboard graphics may be added for larger (and longer) installations. For small installations, like the tutorial OfficeSuite example, only one billboard will show.

Billboards may also be assigned to features, and will only be displayed if the feature they are associated with installs.

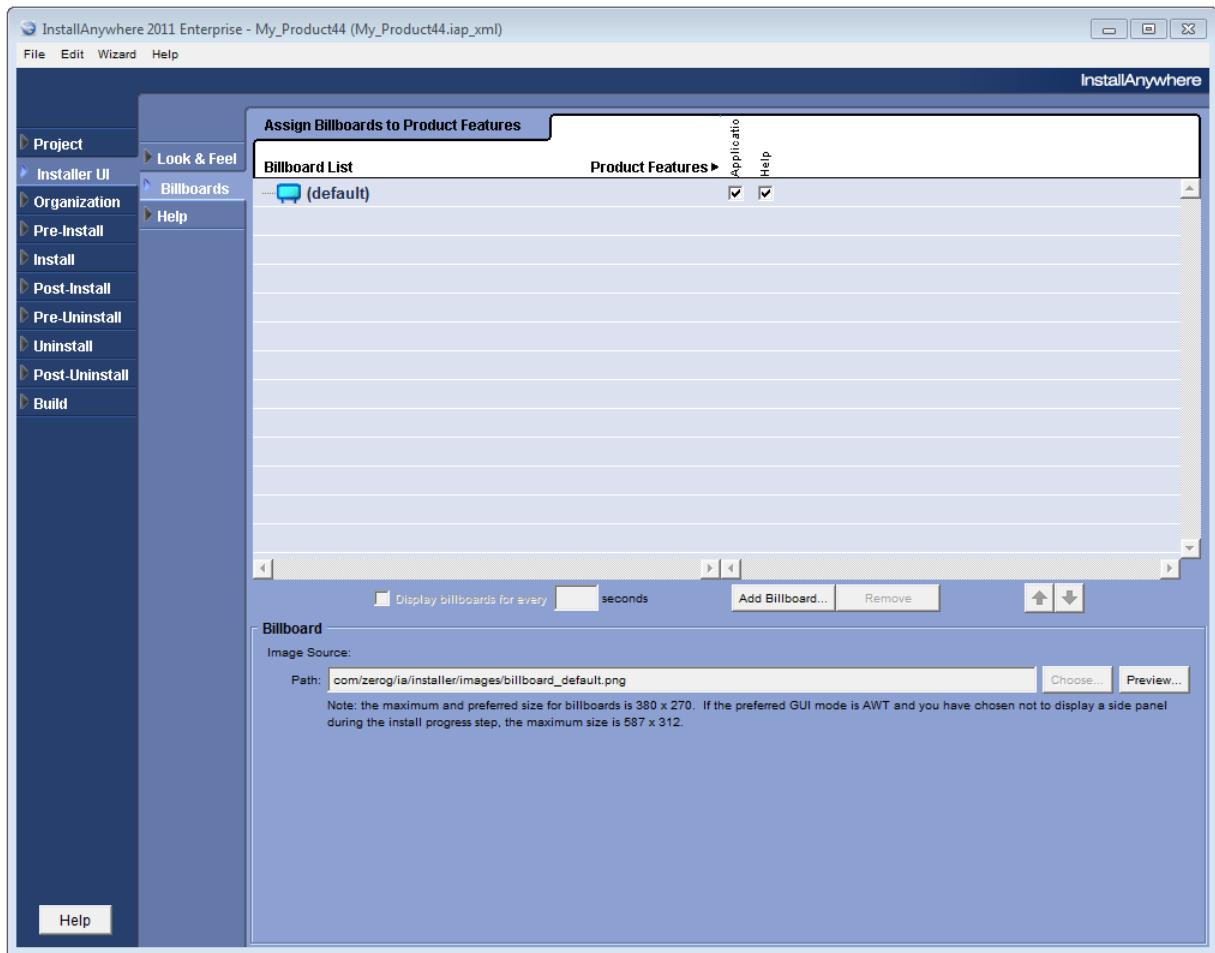


Figure 7-31: Installer UI > Billboards Subtask

When adding multiple billboards, the billboards appear in the order they are shown in the **Billboard List**.



Note • The maximum and preferred size for billboards is 380 x 270. If the preferred GUI mode is AWT and you have chosen not to display a side panel during the Install Progress step, the maximum size is 587 x 312.

The **Installer UI > Billboards** subtask includes the following controls:

Table 7-49 • Installer UI > Billboards Subtask Controls

Control	Description
Billboard List	List of all of the billboards that have been added to the project. Use the check boxes in the Product Features area to assign billboard to features.

Table 7-49 • Installer UI > Billboards Subtask Controls

Control	Description
Display billboards for every nn seconds	Select this option to display a billboard for a specified number of seconds, and then enter a number in the seconds box. For more information, see Billboard Timers .
Add Billboard	Click to add a billboard to the list. The Choose an Image File dialog box opens, prompting you to select the billboard image. 
	Note • <i>The maximum and preferred size for billboards is 380 x 270. If the preferred GUI mode is AWT and you have chosen not to display a side panel during the Install Progress step, the maximum size is 587 x 312.</i>
Remove	Click to remove the selected billboard.
Up and Down Arrows	When adding multiple billboards, the billboards appear in the order they are shown in the Billboard List . Use the arrows to specify the billboard order.
Image Source Path	Lists the file name and path of the selected billboard image. Click Preview to preview the billboard.

Billboard Timers

When multiple billboards are specified, you can control how much time each billboard will be displayed while the installation is in progress by using the **Display billboards for every nn seconds** option.

- **Default time period**—The default time for a billboard to be displayed is 0.5 seconds.
- **Specifying a time period**—To change this time period, select the **Display billboards for every nn seconds** option and enter a number in the **seconds** box.
- **Repeating billboards**—If the install phase is still in progress after all of the billboards have been displayed for the specified time period, the billboards “loop” back to the beginning of the list.
- **Short installation**—If the install phase completes or is cancelled before all billboards are displayed for the specified time period, some of the specified billboards may not be displayed.

Help

The **Help** subtask enables a Help feature for the installer program. Click the **Enable installer help** check box to enable a Help feature for the installer. Selecting **HTML** allows for greater formatting control of the message. Developers may set a single help message which needs to be defined in this window.

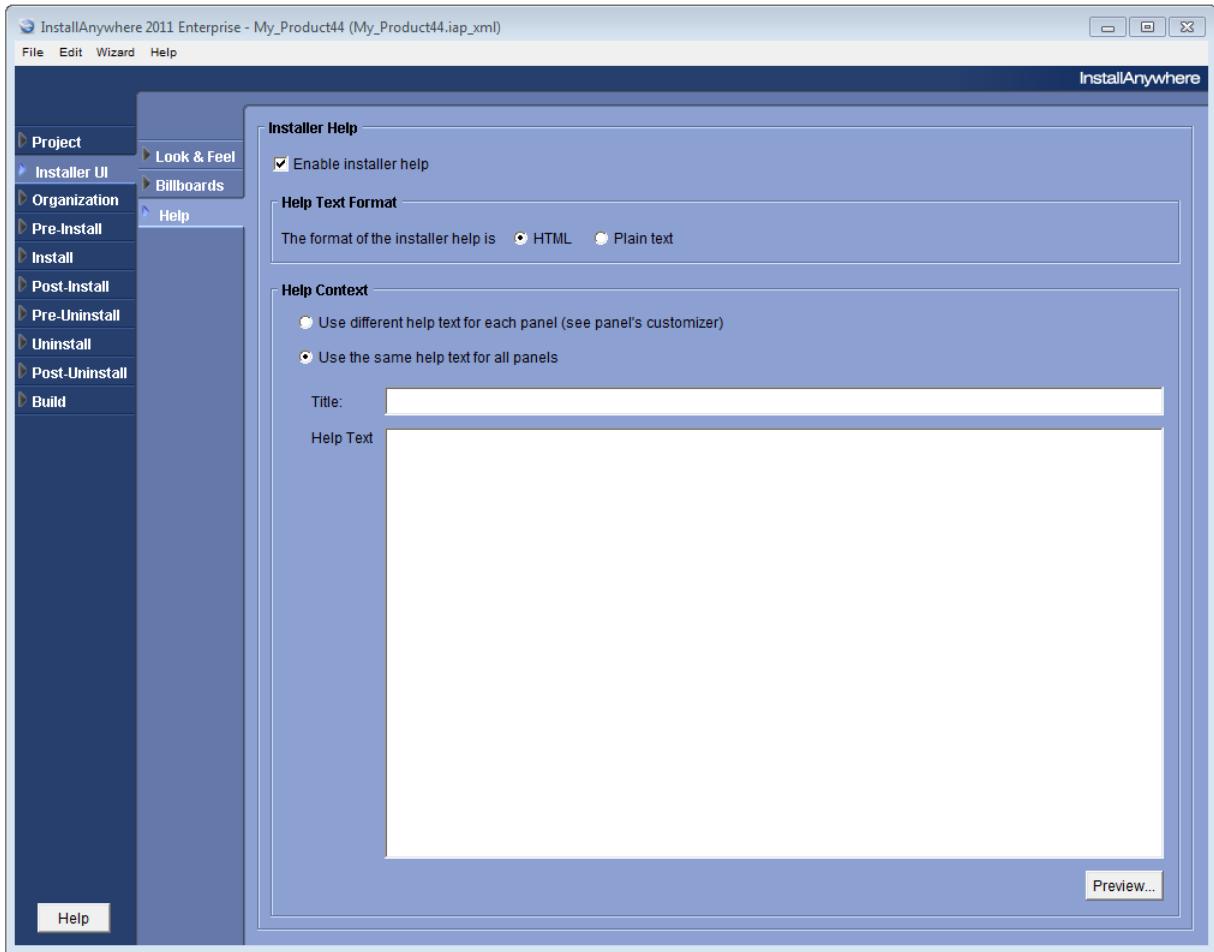


Figure 7-32: Installer UI > Help Subtask

To have a different help instruction for each installer panel, select **Use different help for each text panel**. The action customizer, available at the bottom of the **Install**, **Pre-Install**, **Post-Install**, **Pre-Uninstall**, and **Post-Uninstall** tasks, has a **Help Settings** tab for adding or modifying HTML or text for each installer panel. When **Use different help for each text panel** is selected, the **Help Settings** tab in the action customizer is enabled.

Organization

The **Organization** task enables developers to arrange Install Sets, Product Features, Components, and Merge Modules. Install Sets and Features allow for levels of installation options for the end user of the installer. Components are the smallest widget that can be selected by a Feature set. Install Sets are groupings of Features, and are an organizational tool for the developer of the installer. Components may be much more than files, they can be sophisticated actions that are required to install and run applications or features properly.

There is an interaction between the **Install Sets**, **Features**, and **Components** subtasks, as well as the **Install** task. If an install set is added in the **Install Sets** subtask, features can be assigned to that install set in the **Features** subtask. If a feature is added in the **Features** subtask, components can be assigned to that feature in the **Components** subtask. If a component is added in the **Components** subtask, files and actions can be assigned to that component after the files or actions are added in the **Install** task.

The **Organization** task includes the following subtasks:

- [Install Sets](#)
- [Features](#)
- [Components](#)
- [Modules](#)
- [DIM References](#)
- [Hosts](#)



Note • For more information on installer organization, see [Organizing Features and Components](#).

Install Sets

Install Sets are a set of product features. The **Install Sets** subtask allows developers to add, name, remove, or order Install sets in the installer. In the **Install Set List**, developers define which install set (or sets) to use as the default option to provide to the end user. Features are assigned to install sets in the **Features** subtask.

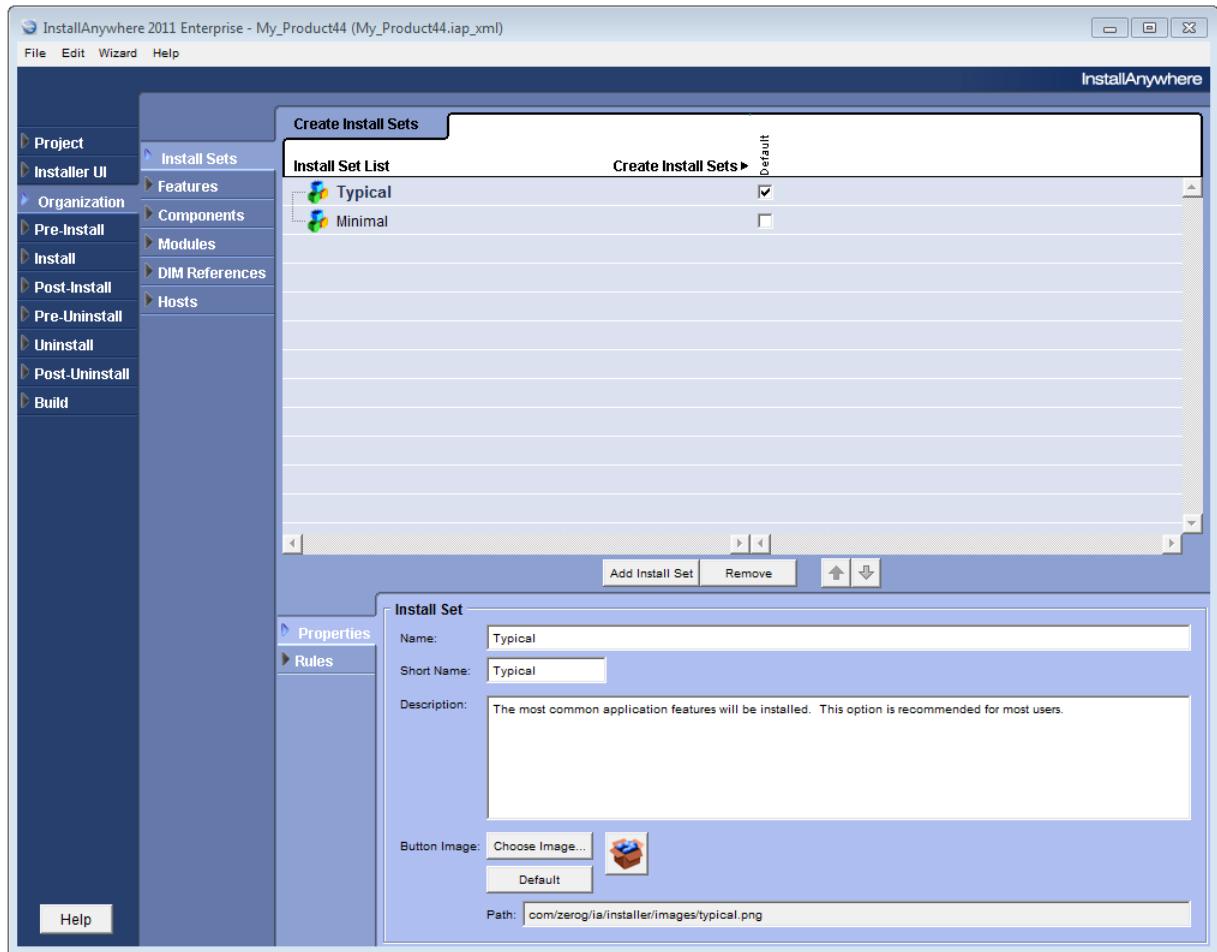


Figure 7-33: Organization > Install Sets Subtask

When the installer requests install set information, each install set is represented by a graphic element. The **Choose Image** button enables developers to select the graphic element.

Rules may be associated with an install set, and that association is created by selecting **Rules** in the customizer and adding rules. The rules for install sets are evaluated before the install set is installed. If the rules on the Install Set evaluate to false, the Install Set will not be displayed.

Features

Features are meant to identify distinct parts of the product so the end user may choose whether or not to install them. InstallAnywhere Product Features are groupings of components. It is up to the developer to define the logical grouping of components into features by assigning components to the features in the **Components** subtask.

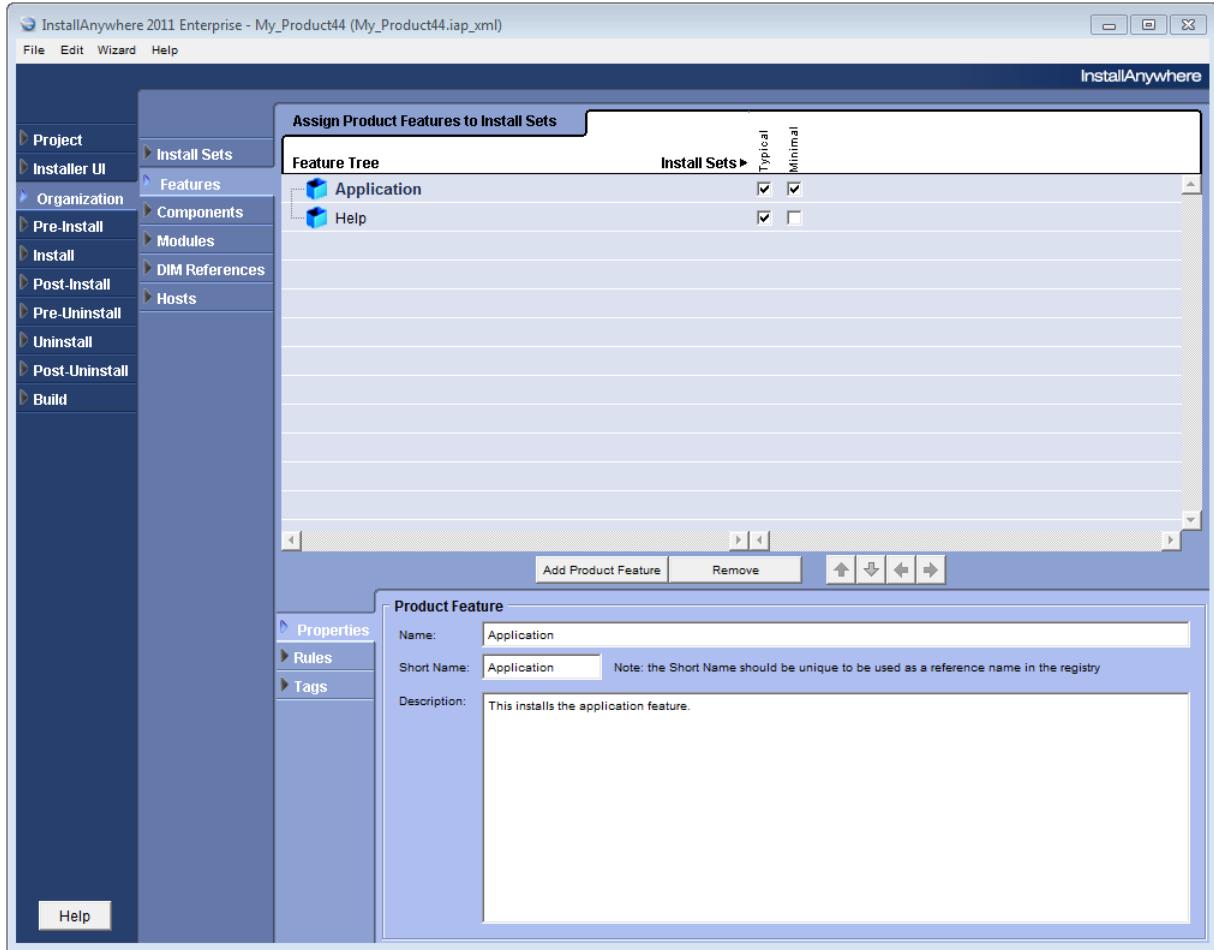


Figure 7-34: Organization > Features Subtask

The **Features** subtask enables developers to add, name, remove, or order features.

Rules may be associated with a feature set, and that association is created by selecting **Rules** in the customizer and adding rules. The rules for feature sets are evaluated before the feature set is installed. If the rules on the Feature evaluate to false, the Feature will not be displayed.

Components

Components are the smallest unit of organization when creating an installer. Unlike Features and Install Sets, Components may be versioned. Components are uniquely identified so developers may update a specific component or use the Find Component in Registry action to locate a particular component.

You add components to a project in the **Organization > Components** task. In the **Components** task, you add and remove components, set component properties, and associate components with features.

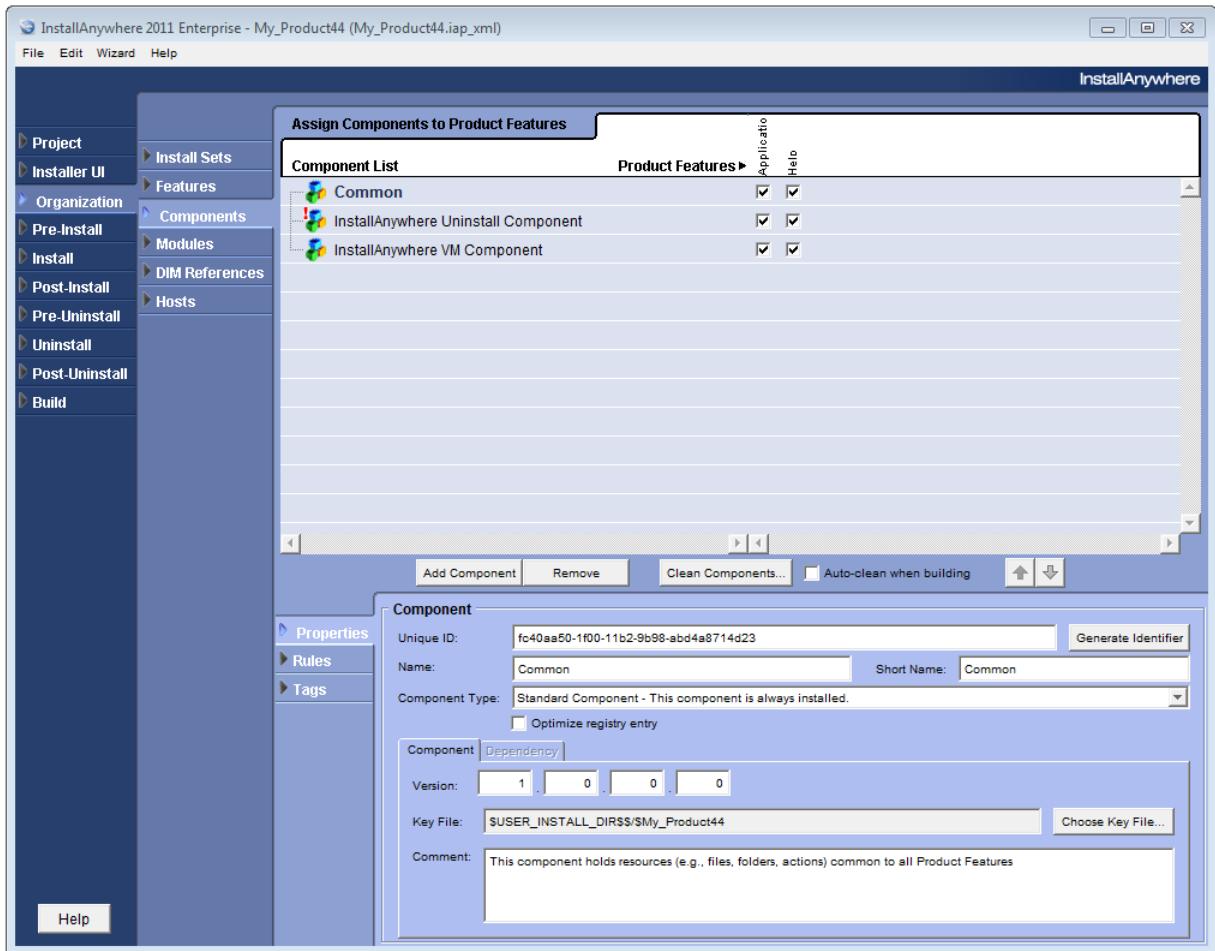


Figure 7-35: Organization > Components Subtask

You can use shared components and component dependencies to create installers that leverage external components, either developed in-house or by a third party, as part of your software distribution strategy. These components can be included as part of a suite installer, be defined as prerequisite dependencies for your package, and can even enable multiple applications to share common components across a system.

Properties Tab

The **Properties** tab of the **Organization > Components** subtask includes the following options:

Table 7-50 • Organization > Components | Properties Tab

Option	Description
Unique ID	<p>When you add a new component, an alphanumeric string is automatically entered to uniquely identify this component. Click Generate Identifier to generate a new ID.</p> <p>The Unique ID value is a UUID that must be different from the UUID of any other component, except for a different version of the same component.</p>  <p>Note • The Find Component in Registry action enables you to detect a component based on its UUID and version.</p>
Name	Enter a name to identify this Component in the InstallAnywhere user interface.
Short Name	Enter a short reference name, which will be used to uniquely identify the Component in the Registry.
Component Type	<p>Select one of the following options to identify this component's type:</p> <ul style="list-style-type: none"> • Standard Component—This component will always be installed, regardless of previous installations or dependencies. The uninstaller for the product with which a standard component is installed will remove the standard components. • Shared Component—This component will be installed if it has not already been installed on the system. A shared component can be made available to other applications or installations. At uninstall, a shared component will be removed only if no other installed application references it. The uninstaller for the last product referencing the shared component will uninstall it. • Dependency—If a component is specified as a Dependency, instead of installing this component, the installer will require that this component has already been installed on the system. Dependencies are not elements of your install per se, but rather are prerequisite requirements that the installer will enforce.

Table 7-50 • Organization > Components | Properties Tab

Option	Description
Optimize registry entry	<p>Use this option to instruct InstallAnywhere how to update the global registry when installing a new version of a previously installed component.</p> <ul style="list-style-type: none">• Selected—Only the latest version of the component is stored in the global registry. For example, if you install version 1.0 of Component A and then install version 2.0 of Component A, the global registry would only contain an entry for version 2.0 of Component A, the latest version.• Not selected—Both versions of the component are stored in the global registry. For example, if you install version 1.0 of Component A and then install version 2.0 of Component A, the global registry would contain two entries for Component A: one for version 1.0 and one for version 2.0.

Component Subtab

The **Component** subtab of the **Properties** tab of the **Organization > Components** task includes the following options:

Table 7-51 • Organization > Components | Properties Tab / Component Subtab

Option	Description
Version	Enter a 4-digit version.
Key File	<p>A key file is a single file that identifies the component. A key file should always be included in a component.</p> <p>You can enter the name and location of the key file or click Choose Key File to select one of the project files to be the key file.</p>  <p>Note • A component's key file must be present in all subsequent versions of the component. The key file is used to define the component's location when the Find Component in Registry action is used.</p>
Comment	Enter a comment to identify the purpose of this component.

Dependency Subtab

The **Dependency** subtab of the **Properties** tab of the **Organization > Components** task includes the following options:

Table 7-52 • Organization > Components | Properties Tab / Dependency Subtab

Option	Description
Matches Key File Location	Enter the location of the key file of the dependent component. A key file is a single file that identifies the component.
Version at least Version at most	Optionally, specify a version number range to make the dependency search more specific.
Matched Key File Location	This variable will contain where the dependency is installed.
Dependency State Variable	<p>You can use the this variable to see if the search was successful.</p> <ul style="list-style-type: none"> • If the dependency check passes, the Dependency State Variable will be an empty string. • If the dependency check fails, the Dependency State Variable will contain the Dependency Failed Message.
Dependency Failed Message	If the dependency check fails, the Dependency State Variable will contain the message entered in this field.



Note • For more information about dependencies, review the *Sample Dependencies Template* that comes with InstallAnywhere. It is sample project that helps illustrate how dependencies work, and what variables are set.



Note • You can control when the installer searches for dependencies explicitly by using the **Evaluate Dependencies** action. See [Evaluate Dependencies](#).

Rules and Tags Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—On the **Rules** tab of the **Component** customizer, you can associate rules with the component. The rules for a component are evaluated before the component is installed. For more information, see [Rules Reference](#).
- **Tags**—On the **Tags** tab of the **Component** customizer, you can associate **Build Configuration** Tags with the component. See [Assigning Tags to Project Elements](#).

Modules

The **Modules** subtask imports Merge Modules into an InstallAnywhere project. Merge Modules are created as a Distribution option in the **Build** task.

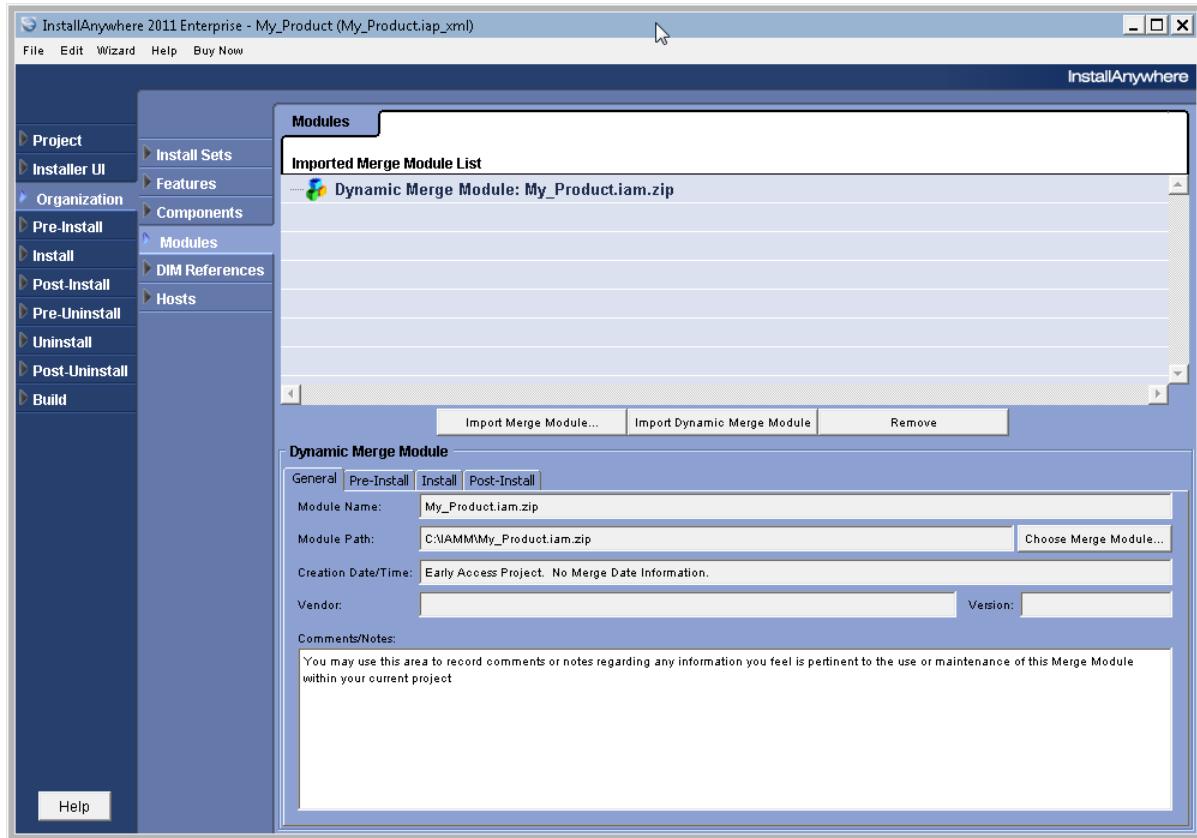


Figure 7-36: Organization > Modules Subtask

The **Organization > Modules** subtask includes the following options:

Table 7-53 • Organization > Modules Subtask

Option	Description
Imported Merge Module List	List of merge modules that have been imported to this project.

Table 7-53 • Organization > Modules Subtask

Option	Description
Import Merge Modules	<p>Click to select a static Merge Module to merge into the current installer, and then customize it on the Merge Module Customizer.</p> <ul style="list-style-type: none"> • All of the Merge Module's features, components, files, actions, and panels (optionally) will be combined into the current project, allowing developers to further customize any settings. • Static Merge Modules are recommended if you want import features and components and if you want more freedom in the place actions. • Changes made to a static Merge Module will not be noted in the parent project.
Import Dynamic Merge Module	<p>Click to select a dynamic Merge Module to merge into the current installer, and then customize it on the Dynamic Merge Module Customizer.</p> <ul style="list-style-type: none"> • All of the Merge Module's files, actions, and panels (optionally) will be combined into current project. • The actions will appear as in an Action Group in Pre-Install and Post-install. <p>Dynamic Merge Modules are recommended if you have Merge Modules whose contents constantly change. A parent project will automatically refresh dynamic Merge Modules when the parent project is loaded and built.</p>
Remove	Select a merge module in the Imported Merge Module List and click Remove to delete the merge module from the project.

Merge Module Customizer

When a Merge Module is added to the Imported Merge Modules list, first the [InstallAnywhere Merge Module Import Assistant Dialog Box](#) opens, and then the Merge Module customizer opens, which includes the following controls:

Table 7-54 • Merge Module Customizer

Control	Description
Module Name	Enter a name to identify the Merge Module.
Module Path	Lists the path to the selected Merge Module.
Creation Date/Time	Date and time Merge Module was created.
Vendors	Name of vendor that created the Merge Module
Version	Version of the selected Merge Module.
Comments/Notes	Enter any information that you feel is pertinent to the use or maintenance of this Merge Module within this project.

Dynamic Merge Module Customizer

When a Dynamic Merge Module is added to the **Imported Merge Modules** list, the **Dynamic Merge Module** customizer opens, which includes the controls on the following tabs:

General

The **General** tab of the Dynamic Merge Module customizer includes the following controls:

Table 7-55 • General Tab / Dynamic Merge Module Customizer

Control	Description
Module Name	Enter a name to identify the Merge Module.
Module Path	Lists the path to the selected Merge Module.
Creation Date/Time	Date and time Merge Module was created.
Vendor	Name of vendor that created the Merge Module.
Version	Version of the selected Merge Module.
Comments/Notes	Enter any information that you feel is pertinent to the use or maintenance of this Merge Module within this project.

Pre-Install

The **Pre-Install** tab of the Dynamic Merge Module customizer includes the following controls:

Table 7-56 • Pre-Install Tab / Dynamic Merge Module Customizer

Control	Description
Import Pre-Install Actions	Select this option to import the Pre-Install actions in this Merge Module into the main project.
Input Variables	Listing of the input variables in this Merge Module that can be set.
Parent Variables	Listing of the parent variables in the main project that are set by this Merge Module.

Install

The **Install** tab of the Dynamic Merge Module customizer includes the following controls:

Table 7-57 • Install Tab / Dynamic Merge Module Customizer

Control	Description
Import Install Actions	Select this option to import the Install actions in this Merge Module into the main project.

Table 7-57 • Install Tab / Dynamic Merge Module Customizer

Control	Description
Uninstall Merge Module when parent is uninstalled	<p>Select this option to uninstall this merge module when the main project gets uninstalled.</p>  <p>Note • The point at which merge modules are uninstalled can also be configured using the Uninstall Merge Modules action in the Uninstall task.</p>  <p>Note • This option is selected by default for new InstallAnywhere 2011 projects. For projects made with previous releases of InstallAnywhere and opened in InstallAnywhere 2011, this option will not be selected.</p>
Add merge module log to parent log	<p>Select this option to append the installation log of this Merge Module to the main project log. By default, this option will not be selected.</p>  <p>Note • For this option to work properly, it is mandatory that logging is enabled for both the parent and the merge module. The same applies for appending the stderr and stdout entries to the log.</p>  <p>Note • You can only append merge module logs to the parent log if Plain text format is selected under Log Format on the Project > Log Settings task. XML log format is not supported.</p>
Input Variables	Listing of the input variables in this Merge Module that can be set.
Parent Variables	Listing of the parent variables in the main project that are set by this Merge Module.
Send stdout to variable Send stderr to variable	To set the stderr and stdout of this Merge Module to InstallAnywhere variables, select these options and enter a variable name in the box. This is useful if you want to debug a Merge Module.

Post-Install

The **Post-Install** tab of the Dynamic Merge Module customizer includes the following controls:

Table 7-58 • Post-Install Tab / Dynamic Merge Module Customizer

Control	Description
Import Post-Install Actions	Select this option to import the Post-Install actions in this Merge Module into the main project.

Table 7-58 • Post-Install Tab / Dynamic Merge Module Customizer

Control	Description
Input Variables	Listing of the input variables in this Merge Module that can be set.
Parent Variables	Listing of the parent variables in the main project that are set by this Merge Module.

DIM References

The **DIM References** subtask supports referencing Developer Installation Manifests (DIMs) in installation projects.

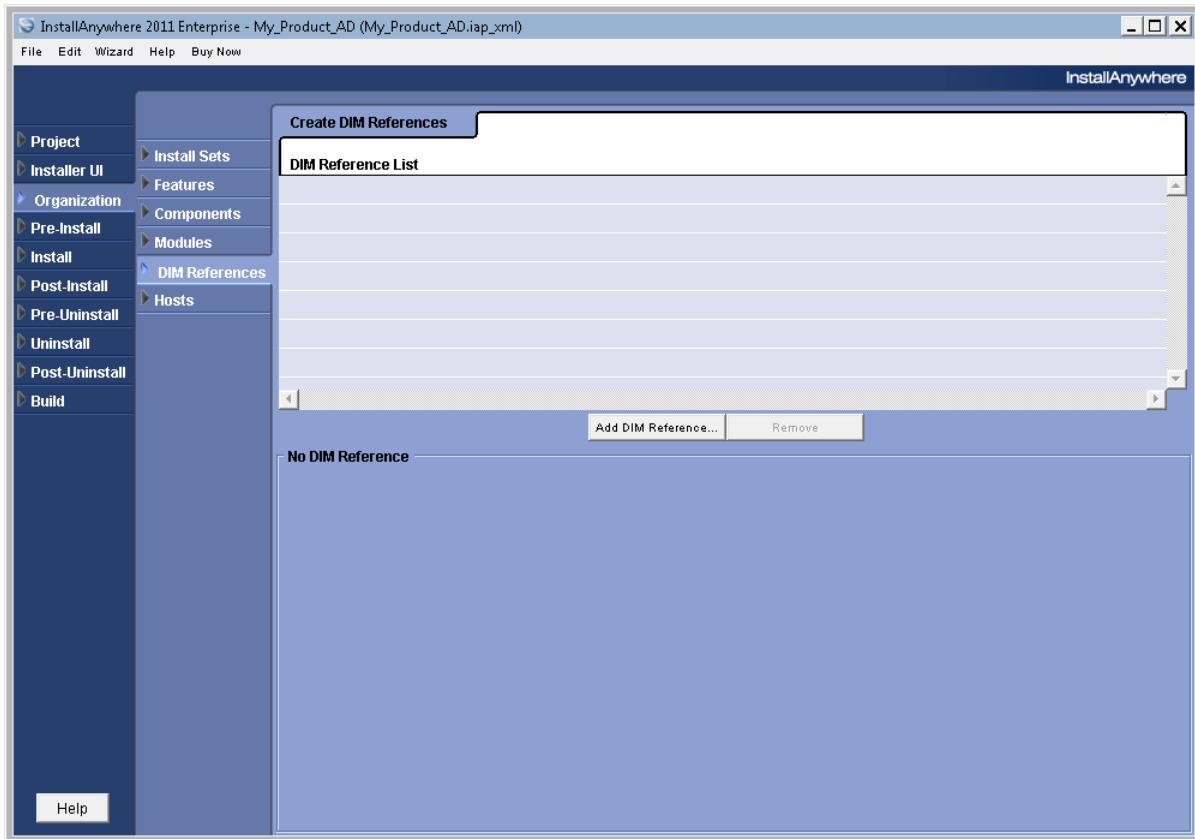


Figure 7-37: Organization > DIM References Subtask

The **Organization > DIM References** subtask includes the following controls:

Table 7-59 • DIM References Subtask Controls

Control	Description
DIM Reference List	Lists DIM references. Click a reference in this list to show the DIM Reference customizer.

Table 7-59 • DIM References Subtask Controls (cont.)

Control	Description
Add DIM Reference	Opens the Add DIM Reference to Project dialog box. In this dialog box, you locate the DIM you want to reference. Navigate to the DIM file and click Open .
Remove	Deletes the currently selected DIM from the DIM Reference list.

DIM Reference Customizer

When a DIM is selected in the **DIM Reference List**, the DIM Reference customizer is displayed. The DIM Reference customizer includes the following tabs:

- General Tab
- Build Variables Tab
- Runtime Variables Tab

General Tab

The **General** tab of the DIM Reference customizer includes the following options:

Table 7-60 • DIM Reference Customizer / General Tab

Option	Description
Name	Name of the selected DIM (read-only).
Source Path	Location of the selected DIM (read-only). Click Choose DIM Reference to open the Add DIM Reference to Project dialog box and select a DIM file.
Unique ID	GUID for the selected DIM (read-only).
Version	Version number of the selected DIM (read-only).
Destination Path	Installation directory (typically a variable) for the contents of the DIM.
Meta Information	Tabular summary of meta tags associated with the selected DIM (read-only).

Build Variables Tab

The **Build Variables** tab of the DIM Reference customizer lists the **Name**, **Description**, and **Value** of each build variable exposed by the selected DIM.

Runtime Variables Tab

The **Runtime Variables** tab of the DIM Reference customizer lists the **Name**, **Description**, and **Value** of each runtime variable exposed by the selected DIM.

Hosts



Edition • The Hosts subtask is only available in the Enterprise Edition. Standard Edition uses only the Operating System host, which requires no additional configuration in the Organization task.

The **Organization > Hosts** subtask allows Enterprise Edition users to define hosts other than the default operating system host. On the Hosts subtask, users can define **Application Server** and **Database Server** hosts. You can use these hosts to target application servers and database servers with your InstallAnywhere installers.

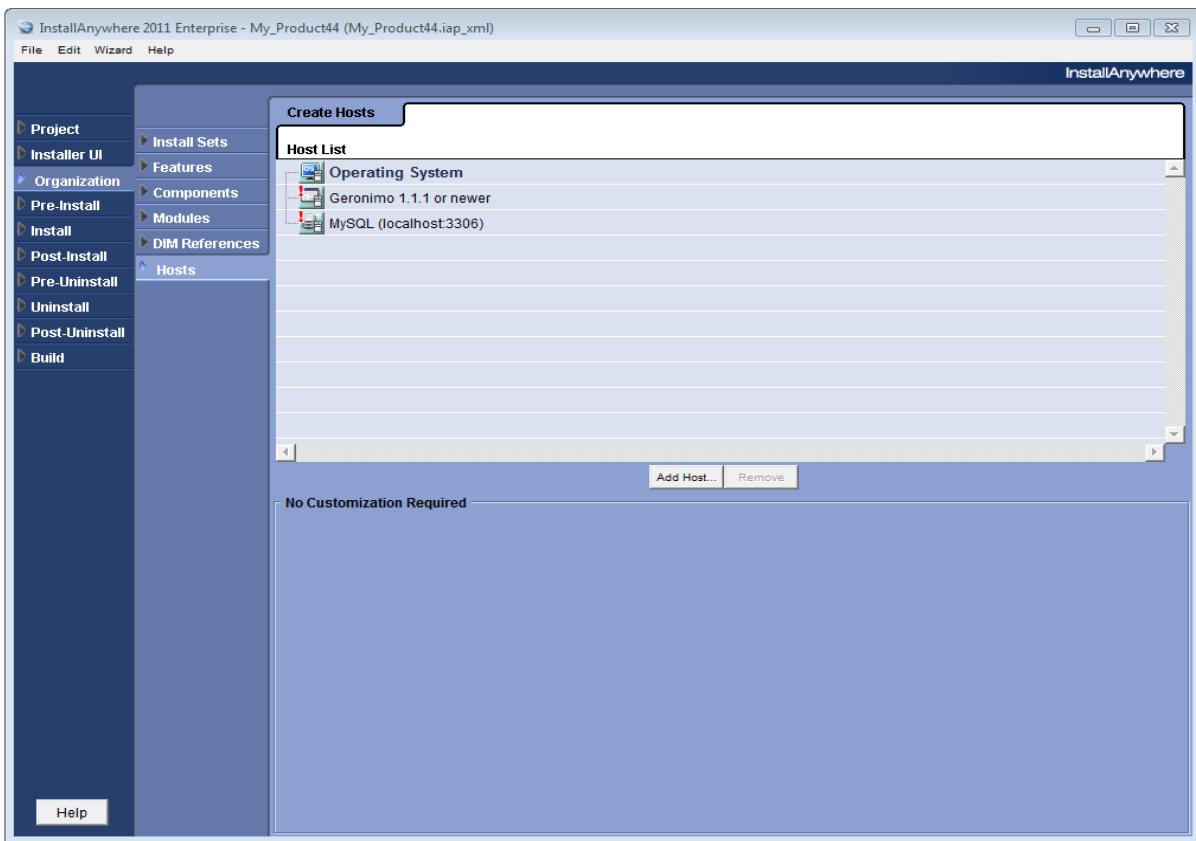


Figure 7-38: Organization > Hosts Subtask

The hosts you define here appear on the Install task at the root level of the visual tree. On the Install task, you can assign files, launchers, and actions to your Operating System, Application Server, and Database Server hosts.

Table 7-61 • Hosts subtask controls

Control	Description
Host List	Lists Operating System , Application Server , and Database Server hosts. Every project includes a default Operating System host that cannot be removed.

Table 7-61 • Hosts subtask controls (cont.)

Control	Description
Add Host	Opens the Choose a Host dialog box. This dialog box lists Application Server and Database Server hosts. Select a host type and click Add .
Remove	Removes the currently selected host from the Host list.
Host Customizer	Shows configuration controls for the currently selected host. (Operating System hosts require no configuration.) <ul style="list-style-type: none">• For details about the Application Server customizer, see Application Server Hosts.• For details about the Database Server customizer, see Database Server Hosts.



Note • **Password** values provided on the customizers for both the Application Server and Database Server hosts are automatically encrypted according to the **Security** settings on the **Project > Variables** subtask.

Application Server Hosts

Application Server hosts can be customized with the Application Server customizer on the Hosts subtask.



Note • *InstallAnywhere supports remote deployment for Geronimo, Sun Application Server, and WebLogic application servers. Using remote deployment, you can create an installer that runs from a machine with no server to deploy software to application servers on connected target systems.*

The Application Server customizer includes the following tabs:

- [General Settings](#)
- [Optional Settings](#)

General Settings

This tab lists the **Server Type**, **Server Path**, and optionally, the **Host Name**, **Port**, and **Bundle Connection Libraries from Local Server Install** controls.

Table 7-62 • General Settings Tab - Application Server Customizer

Control	Description
Server Type	<p>Identify the type of application server to which you want to deploy. Choose from</p> <ul style="list-style-type: none"> • Geronimo • WebSphere • JBoss • WebLogic • TomCat • Sun Application Server • Resin
Server Path	<p>Enter the file path to the server.</p> <ul style="list-style-type: none"> • If Bundle Connection Libraries from Local Server Install is checked, this value is the path to the server install on the build system. (The Server Path value can include source path variables (Access Path Names) to be resolved at build time.) • If Bundle Connection Libraries from Local Server Install is not checked, this value is the path to a server install on the target system. (The Server Path value can include InstallAnywhere variables to be resolved at install time.) • If Server Type is WebSphere, then Server Path is the file path to the active profile.  <p>Note • Only Geronimo, WebLogic, and Sun Application Server support remote WAR/EAR deployment (bundling connection libraries at build time).</p>
Host Name	<p>Enter the host name of the server to which you want to deploy software. Host Name must be an IP address or resolvable name. For example, localhost, 127.0.0.1, or mycompany.com.</p>  <p>Note • Host Name is only shown when the Server Type is Geronimo, WebLogic, WebSphere, and Sun Application Server. For WebSphere, Host Name is read-only and set, by default, to localhost.</p>

Table 7-62 • General Settings Tab - Application Server Customizer (cont.)

Control	Description
Port	<p>Enter the port for the application server to which you want to deploy. For example, 1099, 8880, 7001, or 4848.</p>  <p>Note • For WebSphere, the Port text box is active even though InstallAnywhere does not support remote deployment for WebSphere. Interactions with WebSphere require a port number.</p>
Bundle Connection Libraries from Local Server Install	<p>Click to bundle the connection libraries from this server at build time.</p> <p>Bundling the connection libraries enables your installer to perform a remote deployment of WAR or EAR files to a target machine that does not have an application server already installed. However, bundling connection libraries in the installer can significantly increase installer size.</p> <ul style="list-style-type: none"> • If Bundle Connection Libraries from Local Server Install is checked, your build system must have access to an installation of the application server at build time. The application server does not need to be running at build time. • If Bundle Connection Libraries from Local Server Install is not checked, the installer looks, at install time, for the libraries based on the path you specified for Server Path. If the connection libraries exist there, the installer uses them to handle the WAR/EAR deployment. The application server does not need to be running at install time.

Optional Settings

This tab shows the authentication controls: **Authenticate with Server**, **Username**, and **Password**.

Table 7-63 • Optional Settings Tab - Application Server Customizer

Control	Description
Authenticate with Server	This check box enables the server authentication controls on the Optional Settings tab. Click Authenticate with Server if your want your installer to handle authentication tasks.
Username	Provide a user name for an account with sufficient privileges to perform the WAR/EAR deployment tasks your installer requires.
Password	Provide a password for the account specified by Username .



Note • For more information about deploying WAR and EAR files to an application server, see the **Deploy WAR/EAR Archive** action in [Install Actions](#).

Database Server Hosts

Database Server hosts can be customized in the **Database Server** customizer on the **Organization > Hosts** subtask.

Table 7-64 • Database Server Customizer Controls

Control	Description
Server Type	<p>Identify the type of database server with which your installer must interact. Choose from</p> <ul style="list-style-type: none"> • MySQL • MS SQL Server • Oracle • DB2 • Interbase • Firebird • PostgreSQL • Sybase ASE • Generic JDBC Connection.  <p>Note • Generic JDBC connections allow you to interact with database servers InstallAnywhere does not support.</p>
Server Host	<p>Type a URL for your database server. This value must be a valid IP address. For example, localhost, 127.0.0.1.</p>  <p>Note • When the Server Type is Generic JDBC Connection, InstallAnywhere shows the Custom Connection String field instead of the Server Path, Server Port, and Database Name controls. For generic JDBC connections, you must provide a connection string that works for your database server's syntax requirements.</p>
Server Port	<p>Type the port number on which your database server is operating. InstallAnywhere automatically inserts the most common port number for the database server type selected in Server Type.</p>
Database Name	<p>Type the name of the database with which your installer will interact.</p>
Username	<p>Provide a user name for an account with sufficient privileges to complete the database tasks your installer performs.</p>
Password	<p>Provide a password for the database account indicated by Username.</p>

Table 7-64 • Database Server Customizer Controls (cont.)

Control	Description
Custom Driver Settings	This check box enables you to provide custom values for JDBC Driver Class and Dependencies . InstallAnywhere automatically bundles the default driver and dependencies for the database server selected in Server Type .
JDBC Driver Class	Shows the class for a JDBC driver the database server supports. If Custom Driver Settings is checked, you can edit the value to provide a custom JDBC driver class for your Database Server host.
Dependencies	Shows the libraries currently bundled as dependencies for this Database Server host. If Custom Driver Settings is checked, you can add or remove libraries from the Dependencies list.
Add JAR or ZIP	Opens the Choose Code Archive dialog box. To add a library to the Dependencies list, browse to a JAR or ZIP file that you want to add as a dependency and click Open .
Remove	Deletes the currently selected file from the Dependencies list.



Note • For details on locating database drivers for use with Database Server hosts and Run SQL Script actions, see the following article in the InstallAnywhere Knowledge Base:

<http://support.installshield.com/kb/view.asp?articleid=Q113500>



Note • For more information about running a SQL script on a database server, see the **Run SQL Script** action in **Install Actions**.

Pre-Install

On the **Pre-Install** task, you can specify an ordered sequence of panels and actions that occur before file installation. For more information, see the following procedures:

- [Customizing the Pre-Install Panels and Actions](#)
- [Adding Pre-Install Actions](#)

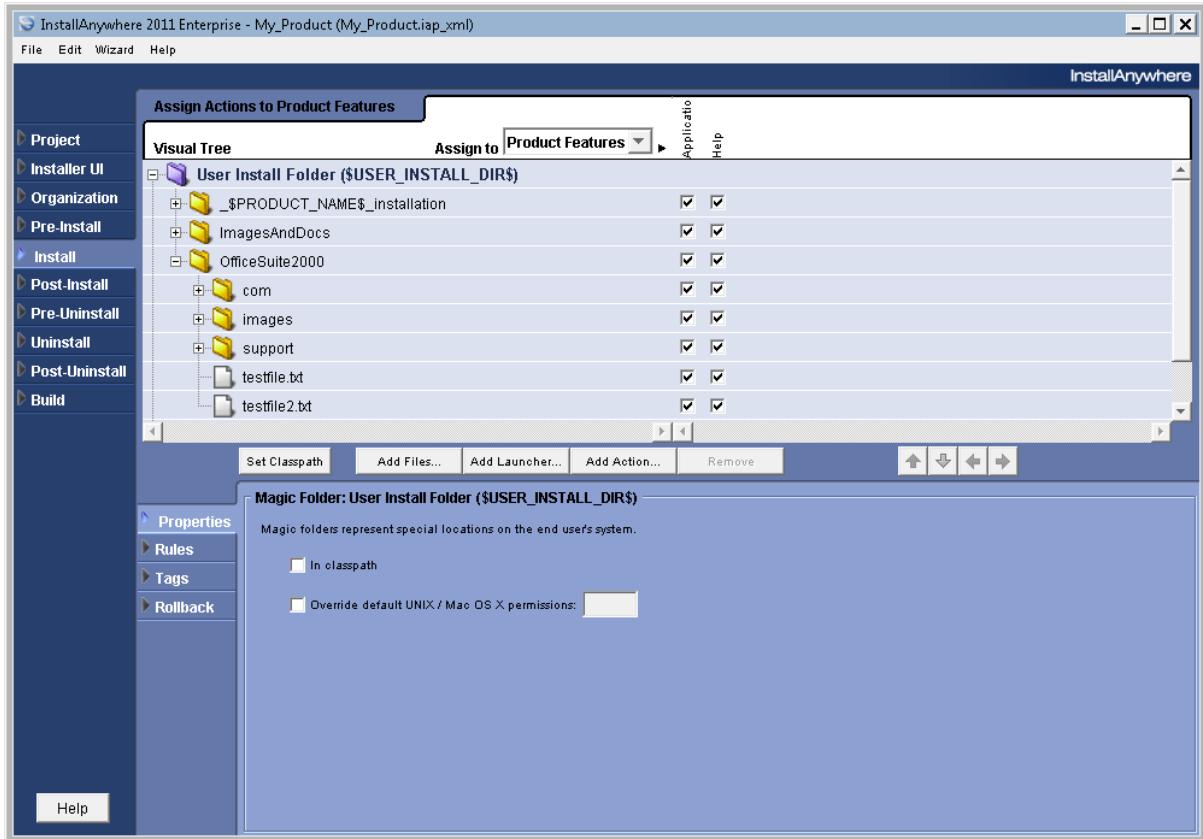


Figure 7-39: Pre-Install Task

Install

The Install task displays files and directories in the way the installer will deploy them on the target system. The controls on the Install task allow install developers to assign files, directories, and actions to either components or product features.

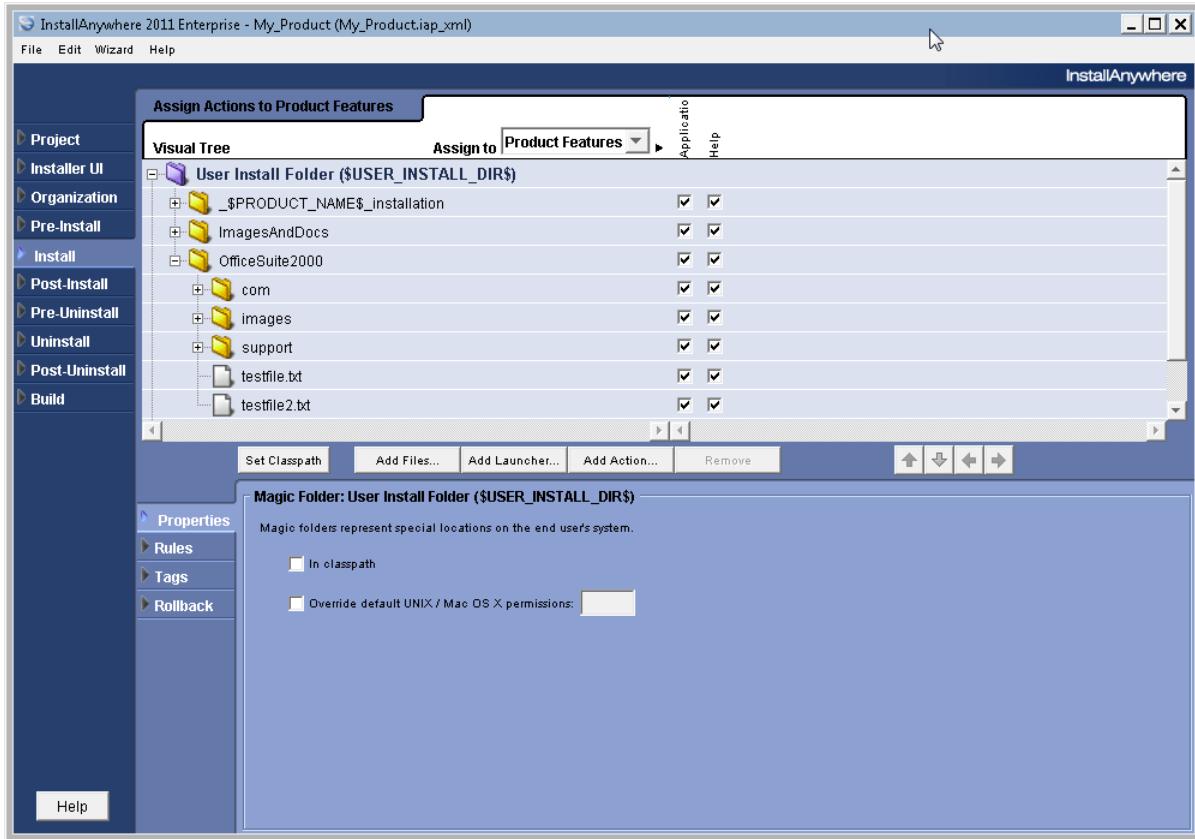


Figure 7-40: Install Task



Important • Only actions (including files, launchers, folders, and other actions) that occur in the *Install* task will be uninstalled by the project's uninstaller. Actions that occur in the *Pre-Install* and *Post-Install* tasks are not automatically uninstalled.

The **Install** task includes the following controls:

Table 7-65 • Install Task Controls

Control	Description
Visual Tree	<p>The Visual Tree displays the hierarchy of hosts, files, folders, and actions that make up your install.</p>  <hr/> <p>Note • <i>Multiple hosts are available only in the Enterprise Edition.</i></p> <p>You can add files to the Visual Tree by clicking Add Files or dragging and dropping directly from your file manager. (Windows Explorer, the Mac OS X Finder, and some Unix/Linux File Managers supported.)</p> <p>Once in the Visual Tree, items may be moved in the hierarchy by dragging and dropping or by clicking the arrow buttons.</p> <p>Right-clicking in the Visual Tree shows a context-sensitive menu for managing the files and actions contained there. For example, right-clicking enables developers to specify a magic folder where they want to place a file.</p>

Table 7-65 • Install Task Controls

Control	Description
Assign To	<p>Along with the files, folders, and actions, the Visual Tree shows either how the files, folders, and actions are assigned to Product Features, or how the files, folders, and actions are assigned to Components. The Assign To menu has two choices—Product Features and Components.</p> <p>Product Features</p> <p>The Assign To Product Features view shows a column for each feature that has been created in the Organization > Features task. The Visual Tree shows a check box in each Feature column for every file, sub-folder, and action.</p> <p>Folders reflect the feature assignments of the files and actions they contain:</p> <ul style="list-style-type: none"> • Check—All files within the folder are assigned to the feature. • Dash—Some, but not all, files in the folder are assigned to the feature. • Blank—No items in the folder belong to the feature. <p>Files or actions will either have a check if it is associated with a feature or blank if it is not associated with a feature.</p> <p>Components</p> <p>The Assign To Components view of the Install task works somewhat differently from the Assign To Product Features view. A file may only belong to one component. If there are three Components—Comp 1, Comp 2, and Comp 3—selecting Comp 1 for a file means that file cannot belong to any other Components.</p> <p>Components are assigned automatically when adding the files, but these assignments may be modified.</p> <p>Folders reflect the component assignments of the files and actions they contain:</p> <ul style="list-style-type: none"> • Check—All files within the folder are assigned to the checked component. • Dash—Files within the folder are assigned to multiple components. • Blank—No items in the folder belong to the component.
Set Classpath	<p>The Set Classpath button determines the folders and archives which should be on the project's classpath. If the classpath has been set previously, this command overrides those settings. This feature works for any .zip, .jar, or class files in the project.</p>

Table 7-65 • Install Task Controls

Control	Description
Add Files	<p>The Add Files button opens the Add Files to Project dialog box. In this dialog box, you can add files or directories to your project.</p> <p>The files and directories you select in the Add Files to Project dialog box are added to the Visual Tree:</p> <ul style="list-style-type: none"> • If a file is currently selected in the Visual Tree, the added files or directories are inserted immediately after the selected file. • If a top-level folder is selected in the Visual Tree, the added files or directories are inserted into that folder (after the last file in that folder). • If a sub-folder is selected, the location of the added files depends on whether the subfolder is expanded or collapsed: The files are inserted into the sub-folder when the sub-folder is expanded but they are inserted into the parent folder if the sub-folder is collapsed. <p>For details about the Add Files to Project dialog box, see Add Files to Project Dialog Box. For instructions about adding files to your project, see Adding Files to a Project.</p>
Add Launcher	<p>The Add Launcher button adds a Create LaunchAnywhere for Java Application action (and adds a shortcut that point to the launcher) to the Visual Tree. This action creates a LaunchAnywhere application for the Java application your installer deploys. L</p>  <p>Note • <i>LaunchAnywhere applications make it easy for your end users to launch your Java application because they allow end users to invoke your application in a way that's natural for their platform.</i></p>  <p>Note • <i>A LaunchAnywhere application must point to a file with a main class.</i></p>  <p>Note • <i>For more information about LaunchAnywhere, see LaunchAnywhere and Creating Launchers for Java Applications.</i></p>
Add Action	<p>The Add Action button opens the Choose an Action dialog box.</p> <p>For more information about actions, see Actions, Customizing the Pre-Install Panels and Actions, and Action Customizers and the Action Execution Sequence.</p>
Remove	<p>The Remove button deletes the currently selected action, file, or directory from the Visual Tree.</p>

Table 7-65 • Install Task Controls

Control	Description
Properties Tab	Displays the customizers for the selected project element.
Rules Tab	Use to add rules to the selected project element.
Tags Tab	Use to add Build Configuration Tags to the selected Uninstall Category or action. See Assigning Tags to Project Elements .
Rollback Tab	If the Enable Rollback option is selected on the Project > Advanced task, you can select the options on this tab to specify rollback options for the selected project element: <ul style="list-style-type: none">• If rollback is triggered at install time, uninstall this action/resource—Select this option if you want the selected project element to be uninstalled if the product installation is cancelled or encounters a fatal error. If this option is not selected and a rollback occurs, the project element will not be uninstalled.• If a fatal error occurs on this action/resource at install time, trigger uninstall—Select this option if you want a rollback to be triggered if the selected project element (execution of the action results in a fatal error) fails to execute properly or to be installed properly. If this option is not selected and the execution of this action results in a fatal error during installation, a rollback will not be triggered.

Post-Install

On the **Post-Install** task, you can specify an ordered sequence of panels and actions that occur after file installation. For more information, see the following procedures:

- [Customizing the Post-Install Task](#)
- [Adding Post-Install Actions](#)

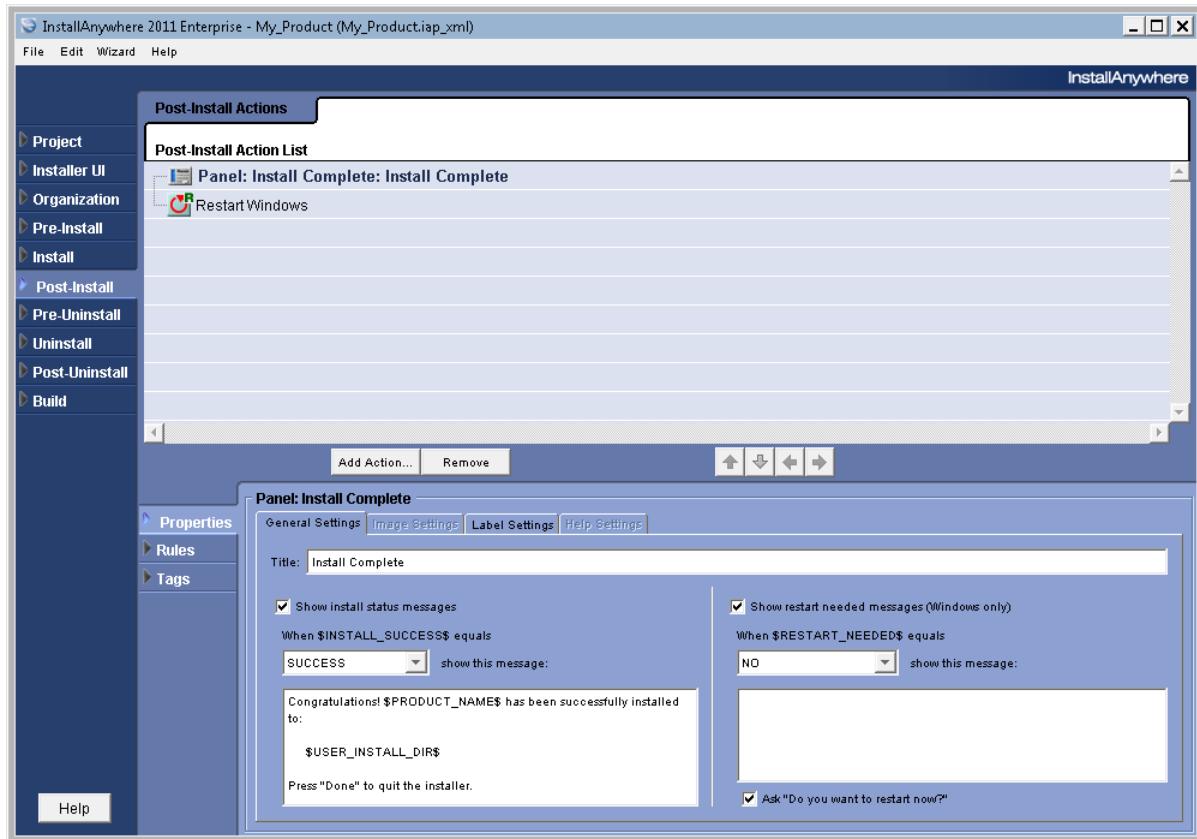


Figure 7-41: Post-Install Task

The **Post-Install** task includes the following controls:

Table 7-66 • Post-Install Task Controls

Control	Description
Action List	<p>The Action List displays the hierarchy of action groups and actions that make up the Post-Install task.</p> <p>You can add action groups and actions to the Action List by clicking the Add Action button. Once in the Action List, items may be moved in the hierarchy by dragging and dropping or by clicking the arrow buttons.</p> <p>Right-clicking in the Action List shows a context-sensitive menu for managing the action groups and actions.</p>
Add Action	<p>The Add Action button opens the Choose an Action dialog box, where you can choose to add action groups and actions.</p> <p>For more information about actions, see General Actions, Panel Actions, and Console Actions, System i (i5/OS) Actions, and plug-ins</p>
Remove	The Remove button deletes the currently selected action, file, or directory from the Visual Tree.
Properties Tab	Displays the customizers for the selected project element.
Rules Tab	Use to add rules to the selected project element.
Tags Tab	Use to add Build Configuration Tags to the selected Uninstall Category or action. See Assigning Tags to Project Elements .

Pre-Uninstall

On the **Pre-Uninstall** task, you can specify an ordered sequence of panels and actions that occur before file uninstallation.

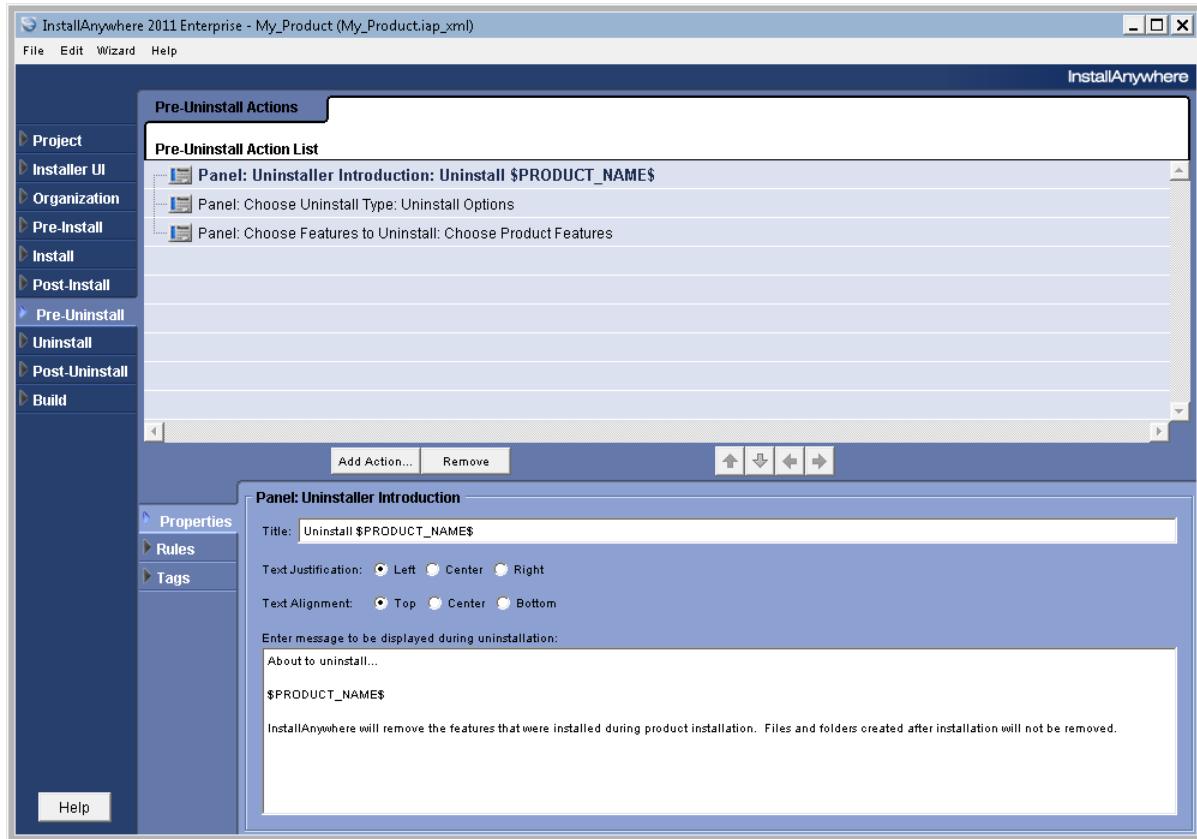


Figure 7-42: Pre-Uninstall Task

Uninstall



Edition • This feature is included with InstallAnywhere Enterprise Edition.

InstallAnywhere automatically creates an uninstaller for the project. The uninstaller, much like the installer, is a collection of panels, consoles, and actions. The standard uninstaller uninstalls the application by executing each action's uninstallation procedure.

However, in some situations, you may want additional flexibility and have more control over how the uninstallation is performed. Therefore, you may want to use the **Uninstall** task to customize the Uninstaller by adding, removing or changing some of the uninstall actions.

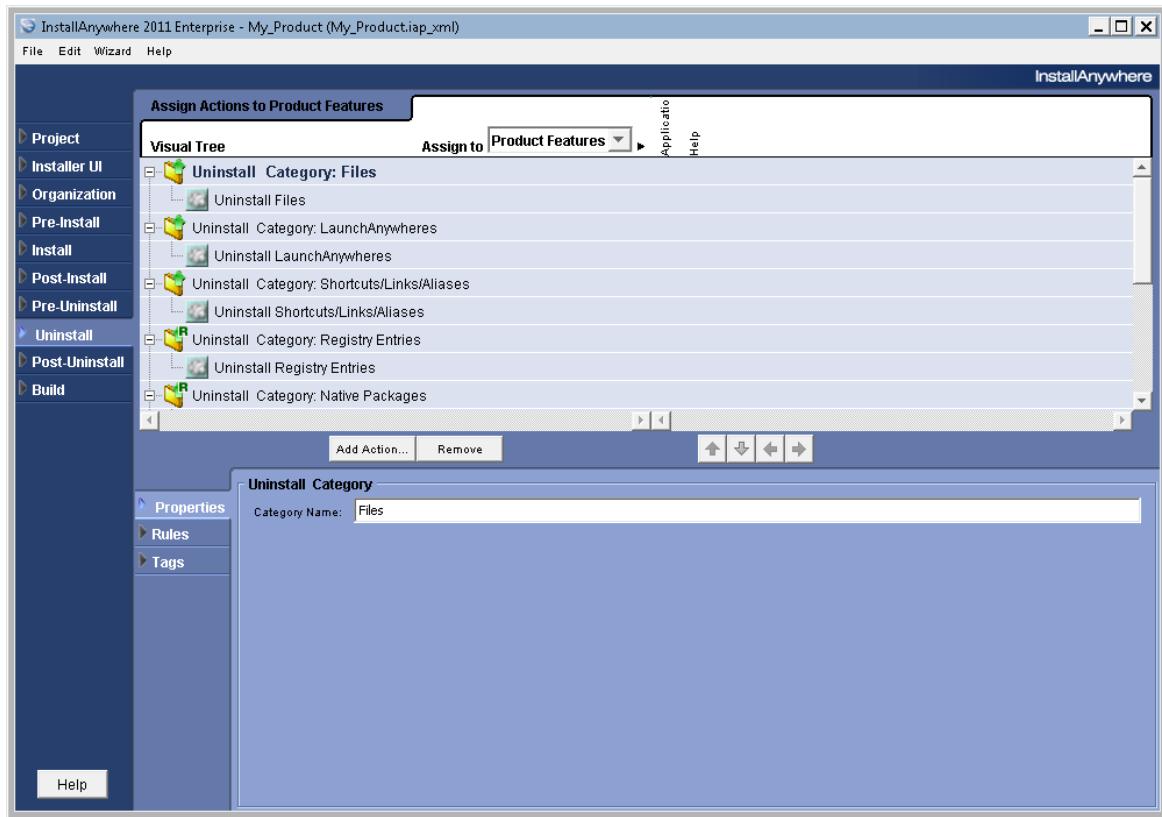


Figure 7-43: Uninstall Task

The Uninstall task includes the following options and controls:

Table 7-67 • Uninstall Task Controls

Control	Description
Visual Tree	<p>The Visual Tree displays the hierarchy of Uninstall Categories and actions that make up the Uninstaller. By default, the following Uninstall Categories are created:</p> <ul style="list-style-type: none"> • Files • LaunchAnywheres • Shortcuts/Links/Aliases • Registry Entries • Native Packages • Folders • Others Category <p>By default, each of these Uninstall Categories contains one Uninstall action that would run the uninstallation of the category.</p> <ul style="list-style-type: none"> • Adding actions—By default, all Uninstall actions are listed in the visual tree. Click Add Action to add additional General Actions to the visual tree or to re-add deleted Uninstall actions. • Moving items—Once in the Visual Tree, items may be moved in the hierarchy by dragging and dropping or by clicking the arrow buttons. • Adding Uninstall categories—To add custom uninstallation behavior, you can add additional custom Uninstall Categories by clicking Add Action and selecting Uninstall Category from the Uninstall tab. • Removing items—Click Remove to remove actions or categories from the Visual Tree. <p> <i>Tip • Right-clicking in the Visual Tree shows a context-sensitive menu for managing the files and actions.</i></p>

Table 7-67 • Uninstall Task Controls

Control	Description
Assign to	<p>On the Uninstall task, you can assign General actions to Product Features or Components. The Assign To menu has two choices—Product Features and Components.</p> <ul style="list-style-type: none"> • Product Features—When you select this option, a column of check boxes for each feature that has been created in the Organization > Features task is displayed. To assign a General action to Product Features, select the check boxes next to the appropriate Product Features. You can select one or more than one Product Feature. • Components—When you select this option, a column of radio buttons for each Component that has been created in the Organization > Components task is displayed. A General action may only belong to one Component. To assign a General action to a Component, select the radio button next to the appropriate Component.  <p>Important • While you are permitted to assign General actions in the Uninstall task to Components or Features, you are not permitted to assign Uninstall Categories or Uninstall Actions to Components or Features. This is because Uninstall Categories and Execute Uninstall Actions are special actions which are not Feature/Component-specific. These are shared across all Components and Features.</p>
Add Action	Click to open the Choose an action dialog box, where you can select an action to add to the Visual Tree. In the Uninstall task, Uninstall Actions and General Actions are available.
Remove	The Remove button deletes the currently selected category or action from the Visual Tree.
Properties Tab	Displays the customizers for the selected Uninstall Category, Uninstall action, or General action. For more information on these customizers, see Uninstall Actions and General Actions .
Rules Tab	Use to add rules to the selected Uninstall Category or action. For more information, see Assigning a Rule to an Action and Rules Reference .
Tags Tab	Use to add Build Configuration Tags to the selected Uninstall Category or action. See Assigning Tags to Project Elements .

Post-Uninstall

On the **Post-Uninstall** task, you can specify an ordered sequence of panels and actions that occur after file uninstallation.

The **Post-Uninstall** task follows the same organization and have the same options as the **Post-Install** task. The differences between the tasks are determined solely by the time of their occurrence in the installation or uninstallation process.

Using the **Post-Uninstall** task, you can add, remove, or order actions. The actions will occur in the order they appear in the action list. Actions may be moved by dragging and dropping, or by selecting and using the ordering arrows. Use the **Post-Uninstall** task to add the Panel, Console, and General actions needed by your uninstaller.

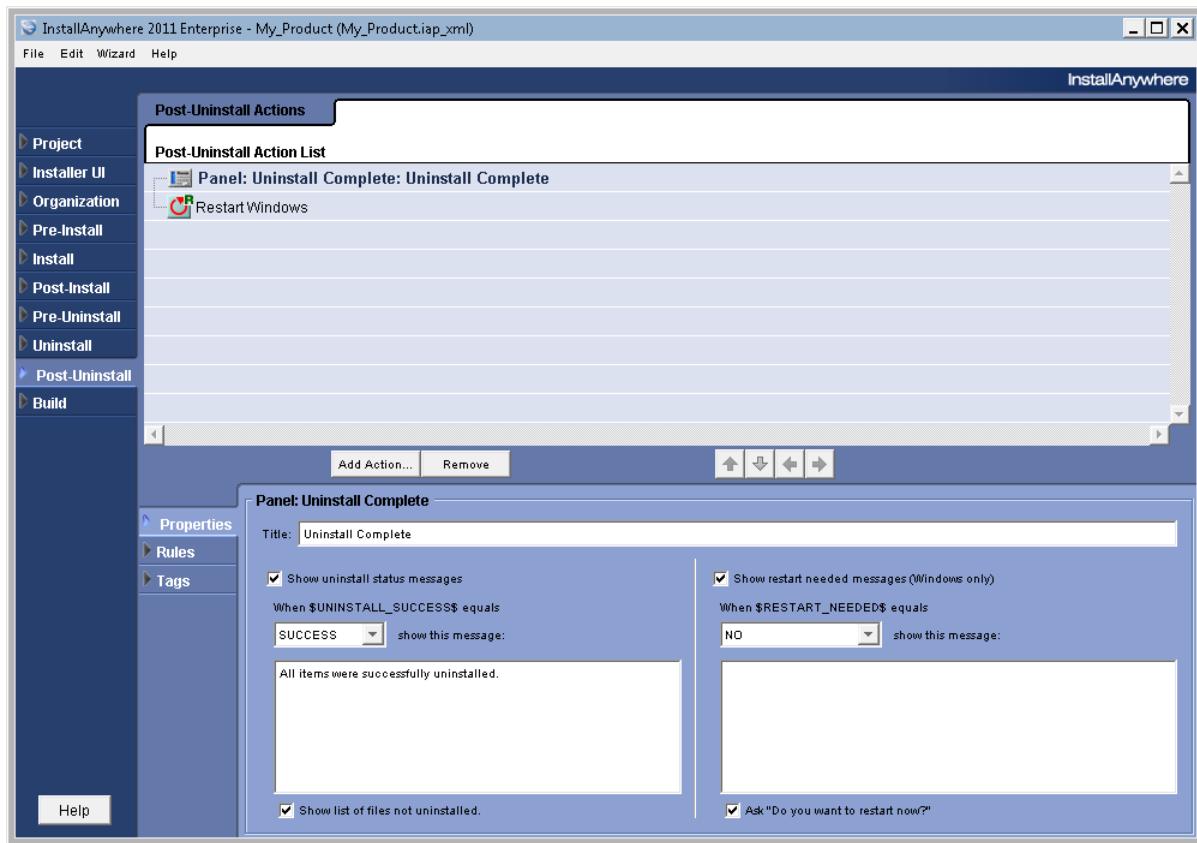


Figure 7-44: Post-Uninstall Task

Build

The **Build** task provides the options for building an installer to multiple targets in multiple Build Configurations. The **Build** task includes the following tabs:

Table 7-68 • Build Task Tabs

Tab	Description
Build Configurations Tab	Create and modify Build Configurations, each of which represents how the installer will be built for a particular set of locales, platforms, files, build distributions, JVM's, locales, and other settings.
Build Log Tab	Reports the results of the most recent build.

Build Configurations Tab



Edition • The Build Configurations feature is included in the InstallAnywhere Enterprise Edition.

The Build Configuration feature enables you to have total control over which software objects are included in a build. Each InstallAnywhere project can have multiple Build Configurations, each representing how the installer will be built for particular set of locales, platforms, files, build distributions, JVM's, and other settings. You create and modify Build Configurations on the **Build Configurations** tab of the Advanced Designer's **Build** task. On this tab, you can add, edit, or delete a Build Configuration.

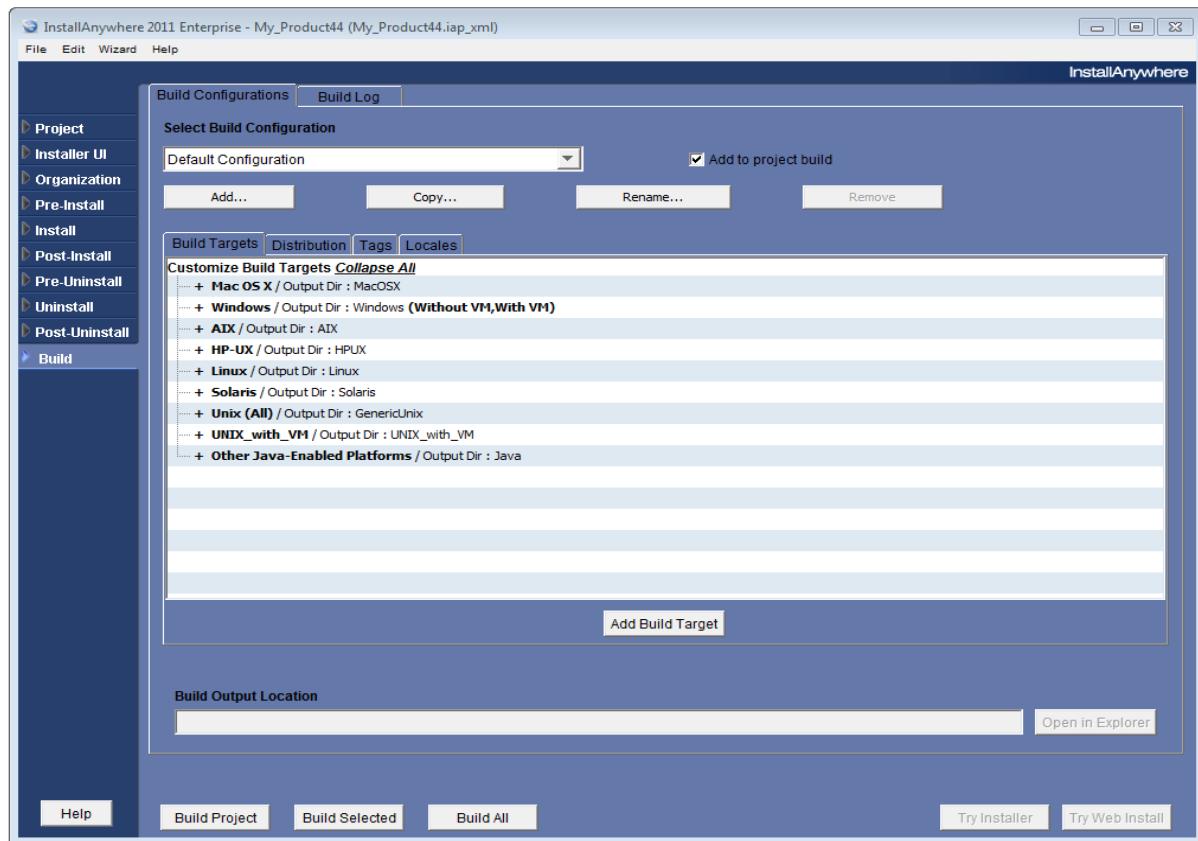


Figure 7-45: Build Configurations Tab of the Build Task

The **Build Configurations** subtab includes the following controls:

Table 7-69 • Build Configurations Subtab

Control	Description
Select Build Configuration	<p>Select a Build Configuration from this list and then perform one of the following actions:</p> <ul style="list-style-type: none"> • Select the Add to project build option and then select Save on the File menu to add the selected Build Configuration to the project build. Then, each time you click the Build Project button, this Build Configuration will be built. • Make edits on the Build Targets, Distribution, Tags, and Locales subtabs and then select Save on the File menu to save those edits. • Click Build Selected to build just the selected Build Configuration. • Click Copy to copy the selected Build Configuration using a new name. • Click Rename to edit the name of the selected Build Configuration. • Click Remove to delete the selected Build Configuration. <p>In addition to new Build Configurations that you create, the following two values may be listed in the Select Build Configuration list:</p> <ul style="list-style-type: none"> • Default Configuration—When a new InstallAnywhere project is created, from any of the templates, it is assigned the default Build Configuration, which is named Default Configuration. The Add to project build option is automatically selected for this configuration. All the build settings defined in the template are copied to the Default Configuration. • Migrated Configuration—Projects migrated from the older InstallAnywhere versions will have only one Build Configuration, Migrated Configuration, which will have the same build settings that were found in the old project.
Add to project build	<p>Select this option to include the selected Build Configuration to the Project build. Then, each time you click the Build Project button, this Build Configuration will be built.</p> <p>When a new project is created from any of the Templates, there will be only one Build Configuration with the name Default Configuration. The Add to project build option will be enabled automatically for this configuration.</p>
Add	Click to create a new Build Configuration. The Add Configuration dialog box opens, prompting you to enter a name for the new Build Configuration.
Copy	Click to copy an existing Build Configuration. The Copy Configuration dialog box opens, prompting you to enter a name for the new Build Configuration.
Rename	Click to edit the name of an existing Build Configuration. The Rename Build Configuration dialog box opens, prompting you to enter a new name for the Build Configuration.

Table 7-69 • Build Configurations Subtab

Control	Description
Remove	Click to delete the selected Build Configuration. You will be prompted to confirm the deletion.
Build Targets Tab	Use to define the target platforms for which the project builds installers and determines whether or not each build target includes a VM pack. See Build Targets Subtab .
Add Build Target	Click to create a new build target. See Build Targets Subtab .
Distribution Tab	Use to specify whether, when this Build Configuration is built, InstallAnywhere will build Web installers, CD/DVD installers, or Merge Modules. See Distribution Subtab .
Tags Tab	Use to associate Tags with the selected Build Configuration. See Tags Subtab .
Locales Tab	Use to define the languages to include in each Build Configuration. See Locales Subtab .
Build Output Location	For All the Build Configurations in the project (regardless of how many are being built at the moment), there will be one sub directory for each Build Configuration under the Build Output Location. These subdirectories will be named after the Build Configuration Name. For Migrated Projects, the Build Output location of the “Migrated” Configuration will be \$IA_PROJECT_DIR\$/\$IA_PROJECT_NAME\$_Build_Output
Open in Explorer	Opens a window to show the build output directory.
Build Project	Click to initiate the building of installers based on all of the Build Configurations that have been added to the Project build (by having the Add to project build option selected).
Build Selected	Click to build the selected Build Configuration.
Build All	Click to build all Build Configurations that have been added to the Project.

Table 7-69 • Build Configurations Subtab

Control	Description
[Target Installer Selection List]	<p>This list appears between the Build All and Try Installer buttons if, after the project has been built, there are more than one supported target installers available for the authoring platform. If there are none or only one supported target installer, this list will be hidden.</p> <p>To launch an installer, select a target installer from this list and click Try Installer.</p>  <p>Note • <i>The name that is displayed for each Target Installer uses one of the following formats:</i></p> <ul style="list-style-type: none">• Without VM—OutputFolderName_BuildConfigurationName• With VM—OutputFolderName_BuildConfigurationName - BundledVMName
Try Installer	Click to run the default installer for your operating system. If two or more installers are built for your operating system, InstallAnywhere shows a control that lists the installers (by their Output label and bundled VM) you can try.
Try Web Install	Click to test the installation by launching a Web browser and the InstallAnywhere Web Install Page generated by the build process.

Build Targets Subtab



Edition • This topic describes the features and controls of the Enterprise Edition. InstallAnywhere Standard Edition does not provide the ability to change the platforms listed as build targets. Therefore, that edition does not include the **Add** button, the **Delete Build Target** button, or the **Platform** list.

The **Build Targets** subtab defines which target platforms the installer will be built for. Installers may be created either with a developer-selected Java virtual machine bundled with it, or without a Java VM. To build for a set of platforms, enable the combination of platforms desired, and specify **VM Search Instructions** and **JVM Search Settings** for that platform.

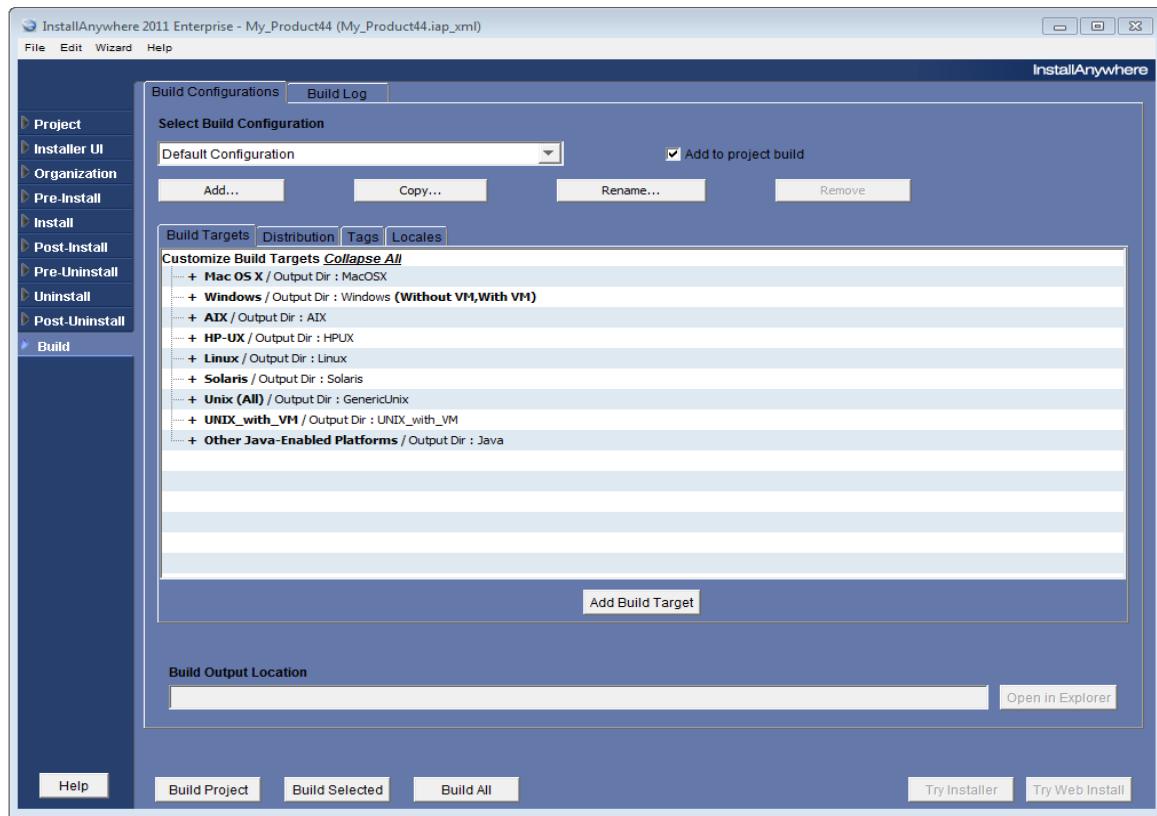


Figure 7-46: Build Configurations Tab / Build Targets Subtab

The **Build Targets** subtab lists available build targets in a collapsible list. When you expand a build target for one of the platforms, a customizer for that platform opens.

- Windows and UNIX Platform Build Targets Customizer
- Mac OS X/Other Java-Enabled Platforms Build Target Customizer

Windows and UNIX Platform Build Targets Customizer

The build target customizer for Windows and UNIX platforms includes controls that you can use to specify **VM Search Instructions** and **JVM Search Settings**.

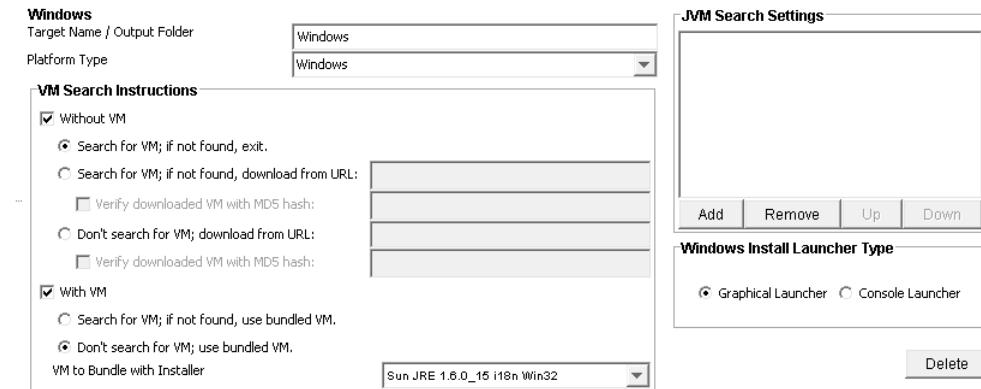


Figure 7-47: Windows and UNIX Platforms Build Target Customizer



Note • See also *Unix-Based Platforms Requirements*.

The Windows and UNIX platforms build target customizer includes the following controls:

Table 7-70 • Windows and UNIX Platforms Build Target Customizer

Control	Description
Target Name / Output Folder	Enter the name for the build target with a description (typically the platform name) that distinguishes it from other installers the project builds. <ul style="list-style-type: none"> • Each build target label must have a unique name. • If you have two build targets for the same platform with the same label, InstallAnywhere appends <u>_1</u> to the label of the duplicate build target.
Platform Type	The platform type is automatically selected for each of the listed build targets, but if you are adding a new build target, you can select a different platform from the list.

Table 7-70 • Windows and UNIX Platforms Build Target Customizer

Control	Description
Without VM	<p>Select this option if you do not want to include a Java virtual machine with this build target. If you select this option, you also need to select one of the following options to specify how the installer will find the VM it will use:</p> <ul style="list-style-type: none"> • Search for VM; if not found exit—Select this option to instruct the installer to search the target machine for a VM. If one is not found, the installer will exit. The search will be performed using the JVM spec file(s) listed in the in JVM Search Settings list. • Search for VM; if not found, download from URL—Select this option to instruct the installer to search the target machine for a VM. If one is not found, the installer will download a VM from the URL specified in the text box. The search will be performed using the JVM spec file(s) listed in the in JVM Search Settings list. <p> Optionally, you can also select the Verify downloaded VM with MD5 hash and enter the MD5 hash (in lower case letters) in the text box. After the installer is built and run, you will then be able to check the integrity of the downloaded VM.</p> <ul style="list-style-type: none"> • Don't search for VM; download from URL—Select this option to instruct the installer to always download a VM from the URL specified in the text box. <p> Optionally, you can also select the Verify downloaded VM with MD5 hash and enter the MD5 hash (in lower case letters) in the text box. After the installer is built and run, you will then be able to check the integrity of the downloaded VM.</p> <p></p> <p>Important • You can obtain VM packs by downloading them from the Flexera Software Web site (from the page that opens when you select Download Additional VM Packs from the Help menu). However, these VM packs are provided for your convenience only and should not be referenced by your installation project.</p> <p>Therefore, if you select the Search for VM; if not found, download from URL or Don't search for VM; download from URL options, do not specify a URL pointing to one of the VM packs on the www.flexerasoftware.com website. If you do, the build will fail. Instead, you should host the VM pack on your own file server or FTP server and specify the URL to that location.</p>

Table 7-70 • Windows and UNIX Platforms Build Target Customizer

Control	Description
With VM	<p>Select this option if you want to include a Java virtual machine with this build target. If you select this option, you also need to select one of the following options to specify whether the installer should use the bundled VM if a VM is found on the target machine.</p> <ul style="list-style-type: none"> • Search for VM; if not found, use bundled VM—Select this option to instruct the installer to search the target machine for a VM. If one is not found, the installer will use the bundled VM. The search will be performed using the JVM spec file(s) listed in the in JVM Search Settings list. • Don't search for VM; use bundled VM—Select this option to instruct the installer to always use the bundled VM, even if a VM exists on the target machine.
VM to Bundle with Installer	<p>If you select the With VM option, select a bundled VM from this list.</p>
JVM Search Settings	<p>If the installer searches for a VM, it uses the JVM search instruction files specified in this list.</p> <ul style="list-style-type: none"> • Add—Click to open the Choose JVM Spec dialog box, where you are prompted to select a JVM spec from the list. • Remove—Click to remove the selected JVM spec from the list. • Up and Down—If more than one JVM spec is listed, use these buttons to list the specs in order of preference. <p> Note • <i>InstallAnywhere provides several pre-written JVM Specs, which are listed on the Choose JVM Spec dialog box. You can also write your own JVM spec. For more information, see JVM Spec Files.</i></p> <p> Note • <i>Starting with InstallAnywhere 2011, the Download Additional VM Packs... button has been moved from the Build Targets subtab to the Help menu.</i></p>
Windows Installer Launcher Type	<p>(Windows only) Identify the type of launcher you want to use. Choose from Graphical Launcher and Console Launcher.</p> <p> Note • <i>InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers will not run under those locales.</i></p>
Delete	<p>Click to delete the build target.</p>



Important • If you are building an installer that is targeting the 32-bit and 64-bit portions of a 64-bit Windows system, you do not need to select the 64-bit Java VM. The 32-bit Java VM targets both portions of the registry.



Note • The download functionality is supported only for simple FTP and HTTP protocols. The build will fail if any other protocol is used.

Unix-Based Platforms Requirements

The following are required on Unix-based platforms:

- **Unzip tool**—It is mandatory to include an unzip tool on Unix-based platforms. If it is not present, the JVM search settings configured in the InstallAnywhere project will not be respected, and the JVM search will be performed as it was in releases prior to InstallAnywhere 2011.
- **wget tool**—If the user chooses to download a JVM on a Unix-based platform, a wget tool is required.
- **tar, gzip, and unzip tools**—For extracting the downloaded JVM on Unix-based platforms, the tar, gzip, and unzip tools are required.
- **md5sum tool**—For MD5 checksum verification of a downloaded JVM on Unix-based platforms, the md5sum tool is required.

Mac OS X/Other Java-Enabled Platforms Build Target Customizer

The Mac OS X/Other Java-Enabled Platforms build target customizer includes controls to define a Mac OS X/Java-enabled platform build target.

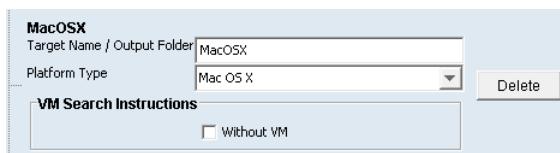


Figure 7-48: Mac OS X/Other Java-Enabled Platforms Build Target

The Mac OS X/Other Java-Enabled Platforms build target customizer includes the following controls:

Table 7-71 • Mac OS X/Other Java-Enabled Platforms Build Target

Control	Description
Target Name / Output Folder	<p>Enter the name for the build target with a description (typically the platform name) that distinguishes it from other installers the project builds.</p> <ul style="list-style-type: none"> • Each build target label must have a unique name. • If you have two build targets for the same platform with the same label, InstallAnywhere appends <u>_1</u> to the label of the duplicate build target.

Table 7-71 • Mac OS X/Other Java-Enabled Platforms Build Target

Control	Description
Platform Type	<p>The platform type is automatically selected for each of the listed build targets, but if you are adding a new build target, you can select a different platform from the list.</p> 
	<p>Note • If you change the Platform Type to another type, the controls displayed on the customizer will change.</p>
Without VM	Select this option if you do not want to include a Java virtual machine with this build target.
Delete	Delete this build target from the project.

Distribution Subtab

Use the **Distribution** subtab to build and optimize an installer for use over the Internet, launched from an HTML file. You can also build and optimize an installer on a single or multiple CD-ROM discs. The **Distribution** subtab also enables developers to build and optimize Merge Modules and Templates.

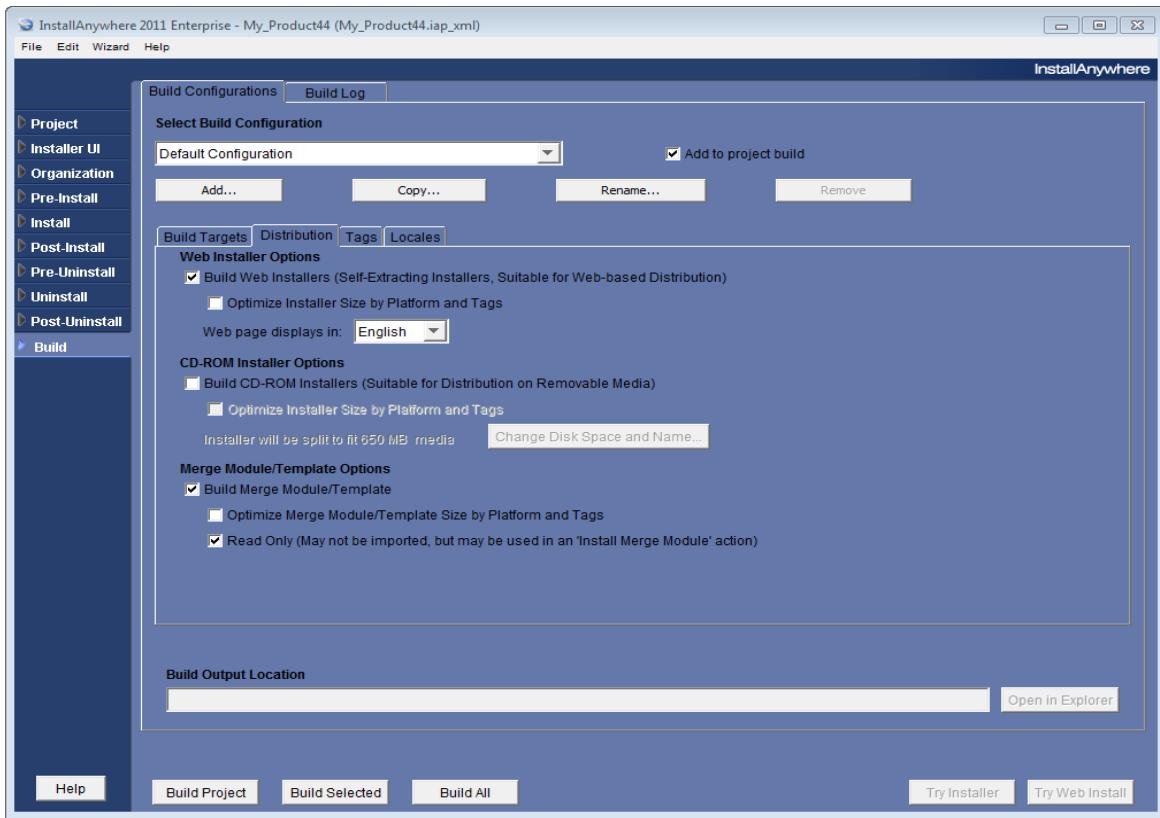


Figure 7-49: Build Configurations Tab / Distribution Subtab

The **Distribution** subtab of the **Build Configurations** tab includes the following options:

Table 7-72 • Distribution Tab Controls

Control	Description
Build Web Installers (Self-Extracting Installers, Suitable for Web-based Distribution)	Select this option to build a Web installer, which is a single executable that contains all of the necessary installation logic. Building the Web installer also generates an HTML page and embedded Java applet to make downloading the installer over the web easy.
Optimize Installer Size by Platform and Tags	Select this option to minimize the size of the final installers by excluding platform-specific resources (as determined by evaluating Check Platform rules) and excluding resources associated with Build Configuration tags that are not associated with the selected Build Configuration.

Table 7-72 • Distribution Tab Controls (cont.)

Control	Description
Web page displays in	Select in which language to build the target web page.
Build CD-ROM Installers (Suitable for Distribution on Removable Media)	<p>These installers consist of multiple files meant to be burned onto a CD, DVD, or multiple CDs or DVDs. They can also be placed on network volumes to provide easier access to large installers. The output of this build process can be directly burned onto disk.</p> <p>InstallAnywhere CD/DVD Installers have the ability to span multiple CDs/DVDs. By default, the installer will automatically segment the installer into a new disk if the size of the installer grows to 650 MB. To control when InstallAnywhere spans to new disks, configure your disk names by clicking the Change Disk Space and Name button to open the Change Disk Space and Name Dialog Box, where you can set the size for each disk, as well as set its name. The name is displayed during the install process when the installer asks for the next disk in a set.</p>  <p>Note • For more information, see About CD-ROM Installers.</p>
Optimize Installer Size by Platform and Tags	Select this option to minimize the size of the final installers by excluding platform-specific resources (as determined by evaluating Check Platform rules) and excluding resources associated with Build Configuration tags that are not associated with the selected Build Configuration.
Build Merge Module/Template	InstallAnywhere can integrate Merge Modules into an installer project at design time. To import a module, select the Import Merge Module button and select the module you wish to import. You will then be given the option to select which tasks' actions. These imported actions will be placed at the bottom of the action list.
Optimize Merge Module/Template Size by Platform and Tags	Select this option to minimize the size of the final merge module or template by excluding platform-specific resources (as determined by evaluating Check Platform rules) and excluding resources associated with Build Configuration tags that are not associated with the selected Build Configuration.
Read Only (May not be imported, but may be used in an 'Install Merge Module' action)	<p>Choose this option to create a merge module that is read-only, meaning that it can be installed but not altered.</p> <p>If a merge module is read-only, it cannot be opened, used as a template, or merged into an installer. A read-only merge module can only be installed as a self-contained installer unit.</p>

About CD-ROM Installers

The directory structure for CD-ROM installers is

```
Platform1/Disk1/InstData/
| -...
| -MediaId.properties
| -Resource1.zip
Platform1/Disk2/InstData/
| -MediaId.properties
| -Resource2.zip
Platform1/Diskn/InstData/
| -MediaId.properties
| -ResourceN.zip
...
```

Disk 1 typically contains an installer binary, often inside a VM or No VM directory. For example, when the **Without VM** and **With VM** options for Windows are both checked on the **Build Targets** subtab, InstallAnywhere places both VM and No VM subdirectories, each with an `install.exe`, inside Disk1/InstData. So the directory structure for Disk 1, in this case, is:

```
Windows/Disk1/InstData/
| -No VM
|   |-install.exe
| -VM
|   |-install.exe
|   |-MediaId.properties
|   |-Resource1.zip
```

However, on other platforms, such as Mac OS X, Unix (All), and Other Java-Enabled Platforms, the install binary is contained in the Disk1/InstData directory. On Mac OS X, for example, the directory structure for Disk 1 is:

```
MacOSX/Disk1/InstData/
| -install.app
| -MediaId.properties
| -Resource1.zip
```

When burning CDs or DVDs, the developer needs to make sure that the folders—Disk1, Disk2, and so on—are burned as is to the disk. Burning *only* the contents of these folders causes installers to work incorrectly. The directory structure for the disk burning application should look like:

```
<ISO CD NAME>
| -Disk1
| -Disk2
```

If you have chosen to split the installer into multiple CD-ROMs, a dialog box will open to prompt the user to insert or locate the subsequent CDs. This dialog box will also open during Maintenance Mode when a feature is being added or repaired.

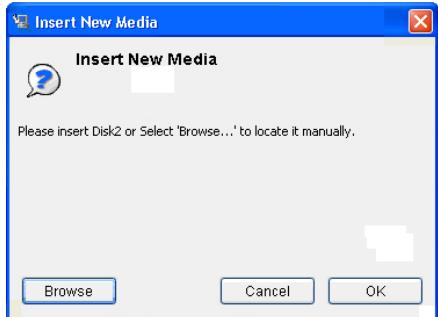


Figure 7-50: Insert New Media Dialog Box

Tags Subtab



Edition • This feature is included with InstallAnywhere Enterprise Edition.

You can use Tags to bundle different sets of actions, panels, features, and components into Build Configurations. After you define a set of Tags for a project, you then assign an appropriate Tag to all of those project elements that you want to include in some Build Configurations but exclude from others. Then, on the **Tags** subtab of the **Build Configurations** tab, you specify which Tags you want to include in the selected Build Configuration and which Tags you want to exclude from it.

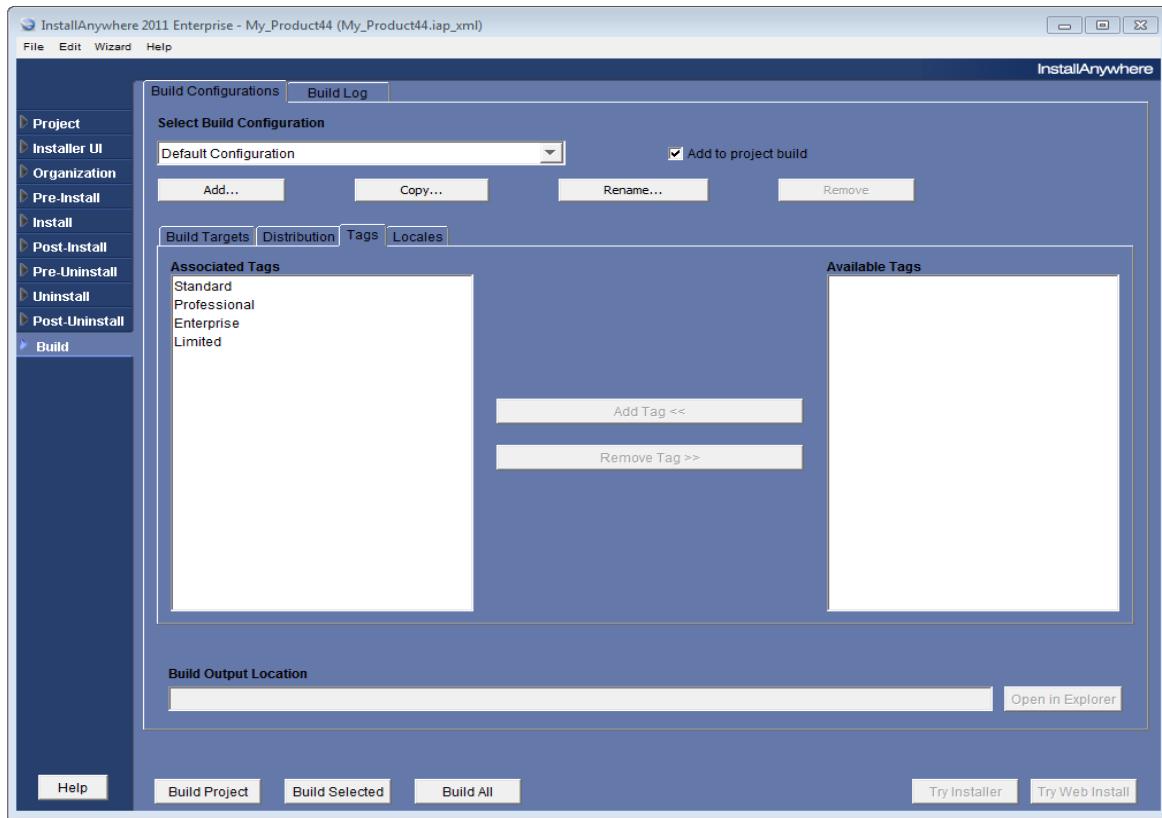


Figure 7-51: Build Configurations Tab / Tags Subtab

The **Tags** subtab of the **Build Configurations** tab includes the following controls:

Table 7-73 • Tags Subtab of the Build Configurations Tab

Control	Description
Associated Tags	List of Tags associated with the selected Build Configuration. When this Build Configuration is built, all untagged project elements plus those project elements associated with the Tags listed in this column will be included.
Available Tags	List of Tags defined in this project that are not associated with the selected Build Configuration. When this Build Configuration is built, project elements associated with the Tags listed in this column will not be included.
Add Tag	Click to move the selected Tag from the Available Tags list to the Associated Tags list, adding it to the Build Configuration.
Remove Tag	Click to move the selected Tag from the Associated Tags list to the Available Tags list, removing it from the Build Configuration.

Locales Subtab

Use the **Locales** subtab to define the languages to include in each Build Configuration. A locale is enabled when it is checked.

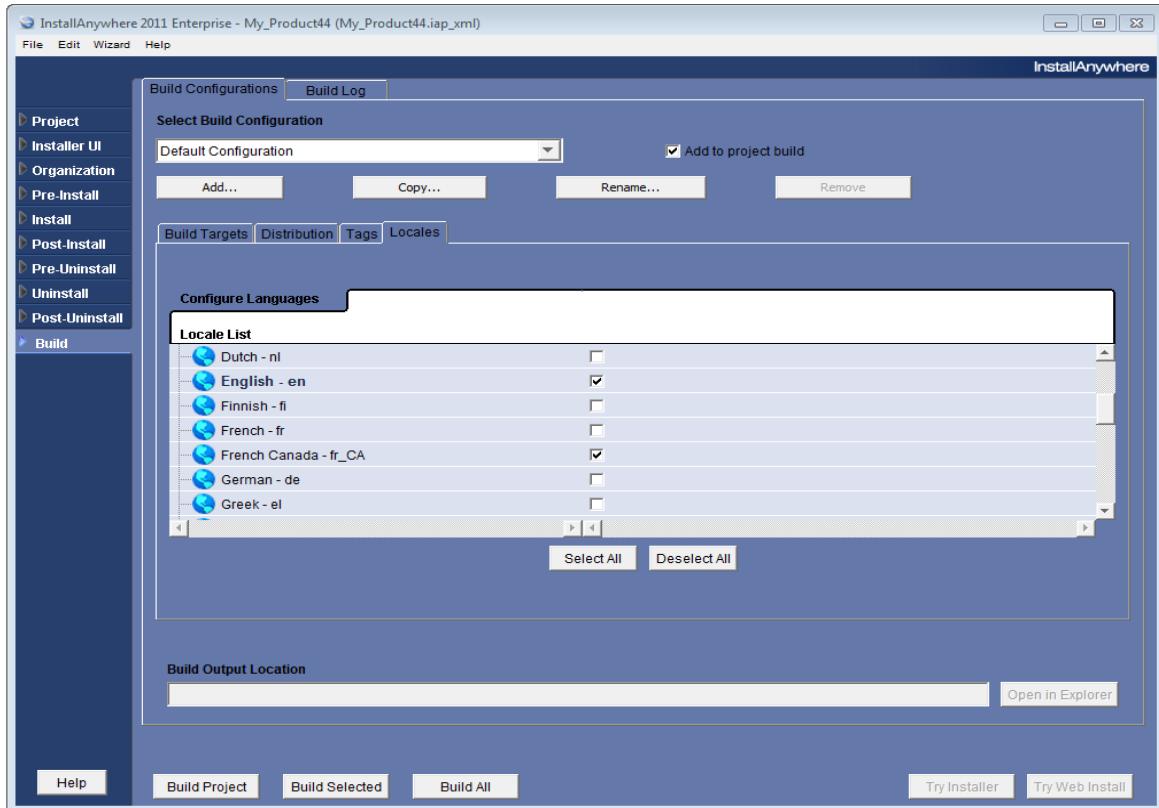


Figure 7-52: Build Configurations Tab / Locales Subtab

The Locales subtab includes the following controls:

Table 7-74 • Build Configurations Tab / Locales Subtab Controls

Control	Description
Locale List	Lists all locale options available to your edition of InstallAnywhere. Click the check box for a locale to enable that locale in your installers.
Select All	Checks all locale options in the Locale List .
Deselect All	Unchecks all locale options in the Locale List except for English .



Note • Avoid customizing the locale files of an InstallAnywhere project while that project is open in InstallAnywhere. InstallAnywhere may overwrite your manual changes to the locale file when you later save the open project.

Build Log Tab

The **Build Log** tab displays an XML log of the build once an installer project is successfully built. Click **Refresh Log** to display the current log. Click **Delete Log** to remove the log.

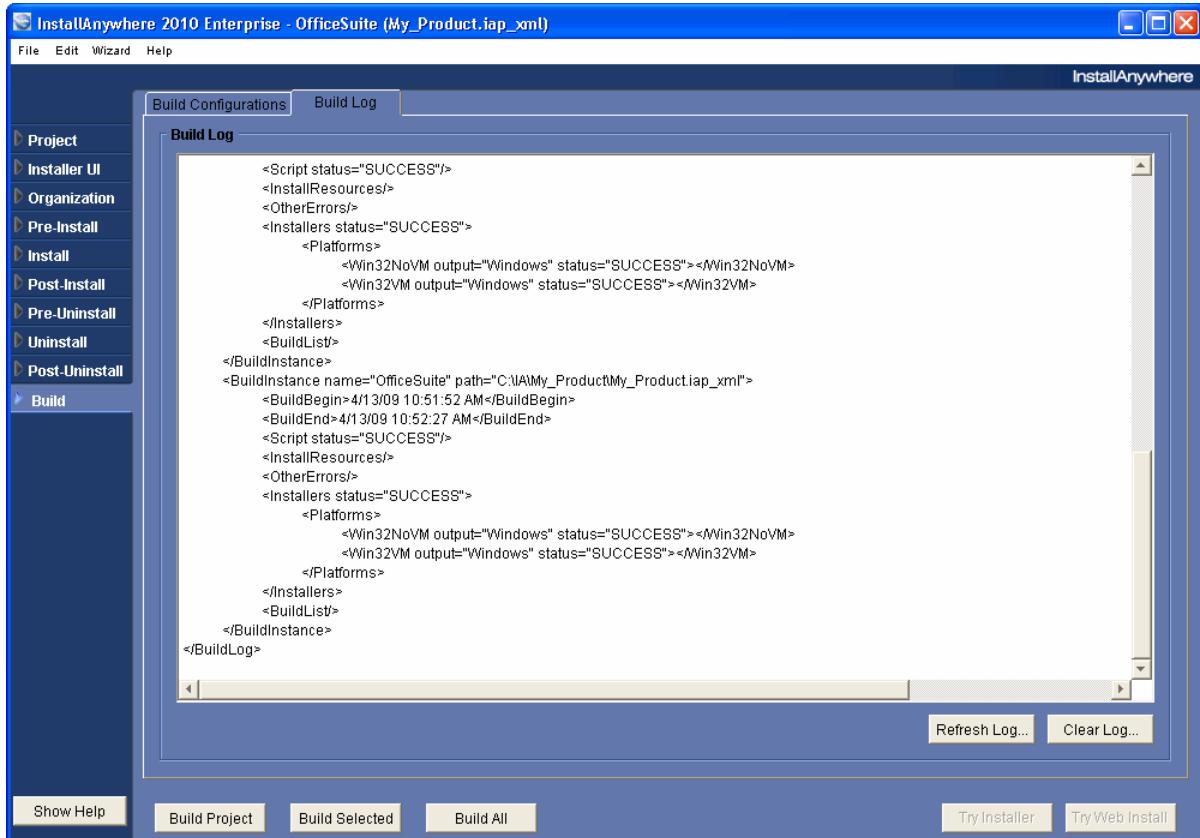


Figure 7-53: Build Log Tab of Build Task

Project Wizard Interface

The Project Wizard takes developers through the simple steps to build an installer. The Project Wizard will build a Web installer by default. From any frame of the Project Wizard, developers may switch to the Advanced Designer.



Note • By default, *InstallAnywhere* shows the *New Project* dialog box when it is launched, and then proceeds to run the Project Wizard interface. You can change the startup behavior to use the Advanced Designer instead by changing the **InstallAnywhere Startup** setting in **Edit > Preferences**.

The Project Wizard simplifies the installer creation process by guiding users through six frames.

Table 7-75 • Frames in the Project Wizard interface

Frame	Description
New Project	Use to create a new project.
Open Project	Use to select an existing project to open.
Project Info	Accepts general project information such as Product Name, Installer Name, Install Folder Name, and Application Shortcut Name.
Add Files	Allows you to add or remove files from the File/Folder Hierarchy.
Choose Main	Provides controls to help you identify the main class or classes for your application and set the application icon.
Classpath	Provides controls to help you specify which items to add to the classpath.
Build Installer	Allows you to select the target platforms for your project and choose to include or exclude a VM. This frame is also where you build installers.
Try Installer	Shows the location of your built installers and your saved project. From this frame, you can also try running the installer.

New Project

When you launch InstallAnywhere, the **New Project** dialog box opens, and prompts you to choose whether you want to create a new project or open an existing project.



Note • The **New Project** dialog box serves as the first panel of the Project Wizard interface. However, after you identify the project that you want to create, you can click the **Advanced Designer** button on this dialog box to switch to the Advanced Designer interface.

If you want to create a new project, select the **Create New Project** option on the **New Project** dialog box.

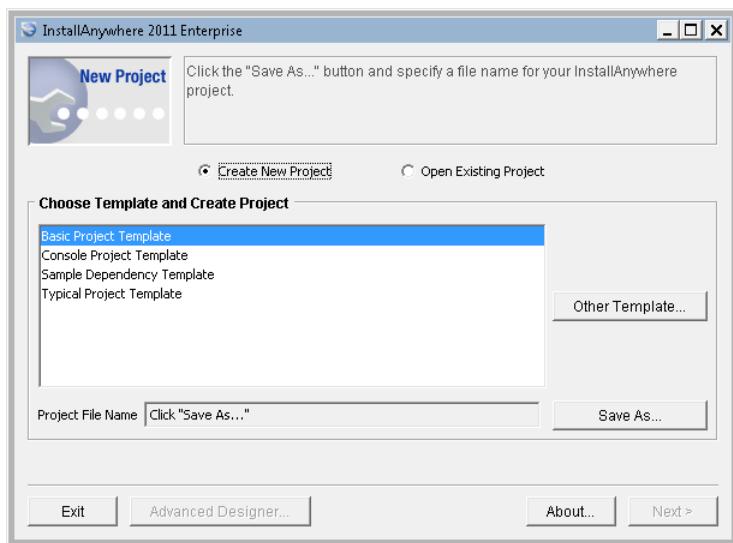


Figure 7-54: New Project Dialog Box

When the **Create New Project** option is selected, the **New Project** dialog box includes the following options:

Table 7-76 • New Project Dialog Box

Option	Description
Choose Template and Create Project	<p>Lists the available templates. Select the template that you want to use to create the new project.</p> <p>A template is simply a Merge Module that has been placed within the <code>iatemplates</code> folder inside the InstallAnywhere installation folder.</p> <p>Templates are generally used as starting points for installer projects. Installer items that remain unchanged such as the license agreement panel would be saved in an InstallAnywhere template. You may also want to create a template to maintain the look and feel for your installer projects.</p>

Table 7-76 • New Project Dialog Box (cont.)

Option	Description
Other Template	Opens the Choose Template dialog box. There, you can select a custom template (*.iam.zip).
Project File Name	After you have clicked Save As to open the Save New Project As dialog box and entered a name for this new project, the project name is listed in this field.
Save As	Opens the Save New Project As dialog box, where you are prompted to enter a name for the new project.
Exit	Click to exit InstallAnywhere.
Advanced Designer	Click to switch to the Advanced Designer interface.
About	Click to open the InstallAnywhere About dialog box.
Next	Click to save the new project using the settings you provided and proceed to the Project Info panel of the Project Wizard.

Open Project

When you launch InstallAnywhere, the **New Project** dialog box opens, and prompts you to choose whether you want to create a new project or open an existing project. If you choose the **Open Existing Project** option, the name of this dialog box changes to **Open Project**.



Note • The **Open Project** dialog box serves as the first panel of the Project Wizard interface. However, after you identify the project that you want to open, you can click the **Advanced Designer** button on this dialog box to switch to the Advanced Designer interface.

If you want to open an existing project, select the **Open Existing Project** option.

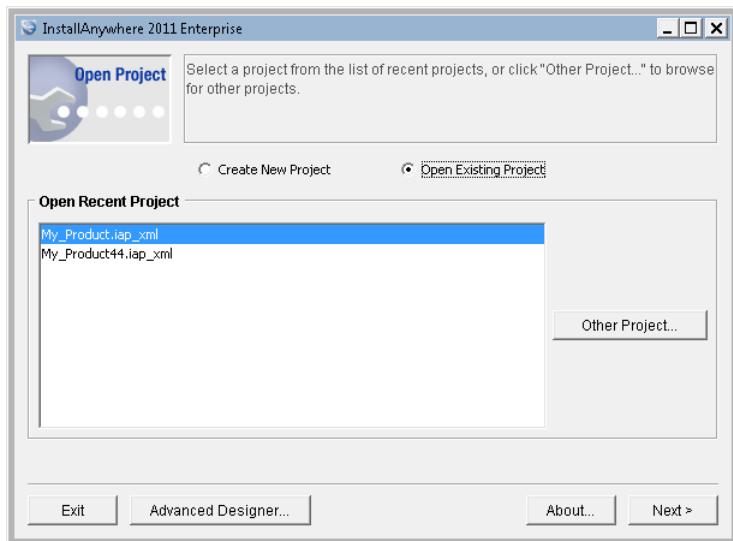


Figure 7-55: Open Project Dialog Box

When the **Open Existing Project** option is selected, the **Open Project** dialog box includes the following options:

Table 7-77 • Open Project Dialog Box

Option	Description
Open Recent Project	List of InstallAnywhere projects that you have recently opened.
Other Project	Click to open the Open Project File dialog box, which prompts you to browse to an existing InstallAnywhere project file.
Exit	Click to exit InstallAnywhere.
Advanced Designer	Click to switch to the Advanced Designer interface.
About	Click to open the InstallAnywhere About dialog box.

Table 7-77 • Open Project Dialog Box (cont.)

Option	Description
Next	Click to proceed to the Project Info panel of the Project Wizard.

Project Info

On the **Project Info** panel of the Project Wizard, you define basic information for the installer.

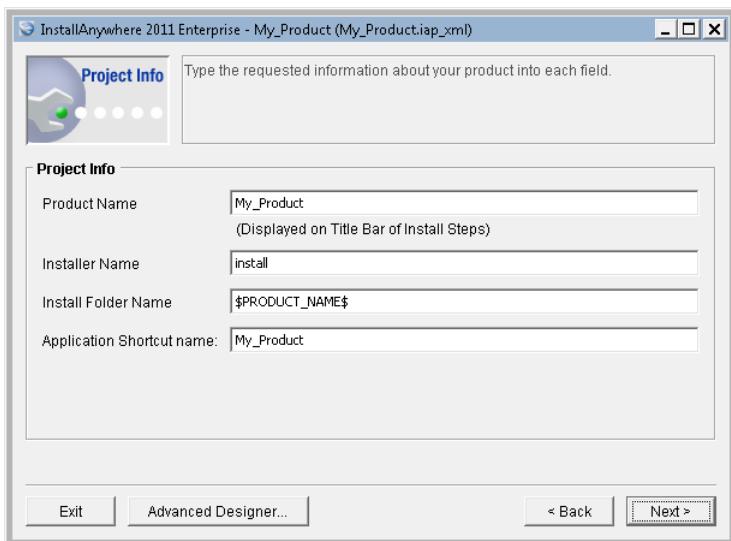


Figure 7-56: Project Info Panel of the Project Wizard

The following information is specified:

Table 7-78 • Project Info Panel of the Project Wizard

Option	Description
Product Name	Enter a name to identify the product.
Installer Name	Enter a name to identify the file name of the actual installer executable file.
Install Folder Name	Enter a location or an InstallAnywhere variable to identify the location where this product will be installed. For Windows, the name entered in this field will identify the directory in the C:\Program Files directory where this product will be installed.

Table 7-78 • Project Info Panel of the Project Wizard

Option	Description
Application Shortcut name	<p>Enter a location on the Start menu where a shortcut to launch this product will be created.</p> <p>On Windows platforms, if OfficeSuite is entered for the Product Name and OfficeSuiteAdvanced is entered for the Application Shortcut name, then you will be able to launch the application by selecting the OfficeSuite > OfficeSuiteAdvanced shortcut on the Start menu.</p>

Add Files

The Add Files frame of the Project Wizard allows developers to add or removes files or directory of files from the installer project.

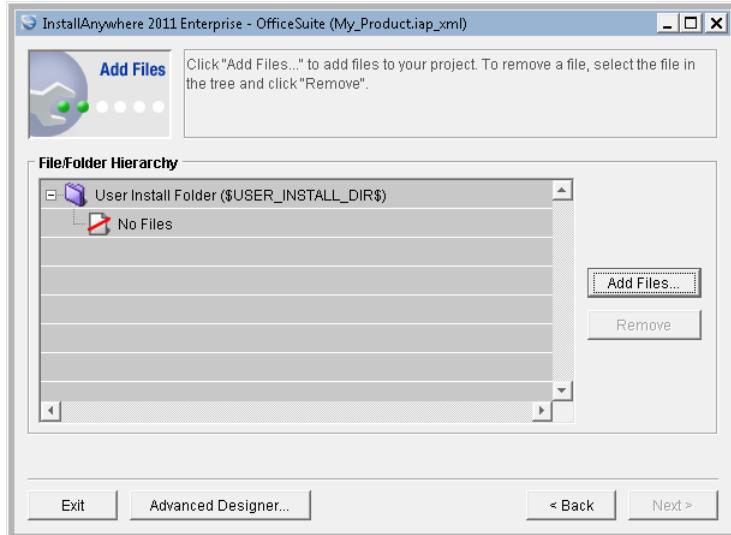


Figure 7-57: Add Files Frame of the Project Wizard

Clicking the Add Files button opens the Add Files To Project dialog box. Add All will select all files or folder showing in the top selection pane. Items that have been selected will be displayed in the bottom pane.

The Add Files To Project dialog uses the standard InstallAnywhere navigation bar with options for up directory, back, forward, down directory, home, and favorites.

Choose Main

The Choose Main frame of the Project Wizard selects the Main Class for the installer. In Java the Main-Class is the starting point when executing a JAR File.

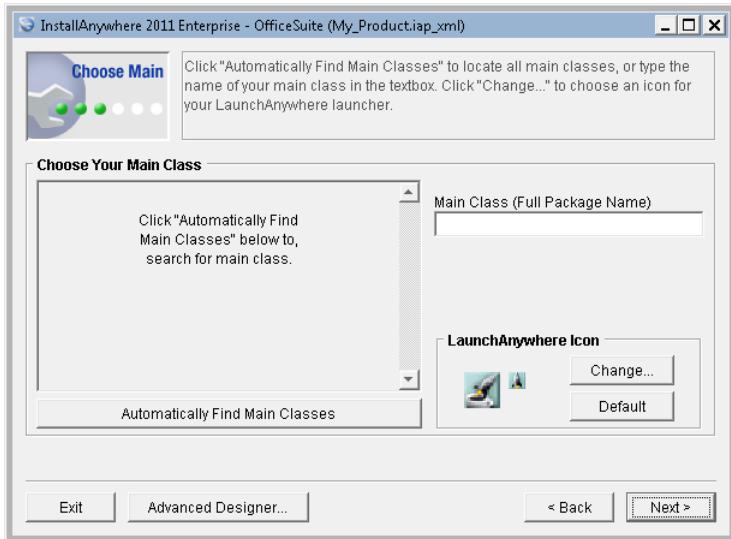


Figure 7-58: Choose Main Frame of the Project Wizard

The Project Wizard will automatically locate main classes to select from when clicking the Automatically Find Main Classes button.

Classpath

The Classpath frame of the Project Wizard selects the classpath. Classpath is how Java applications locate the Java classes the application needs to run.

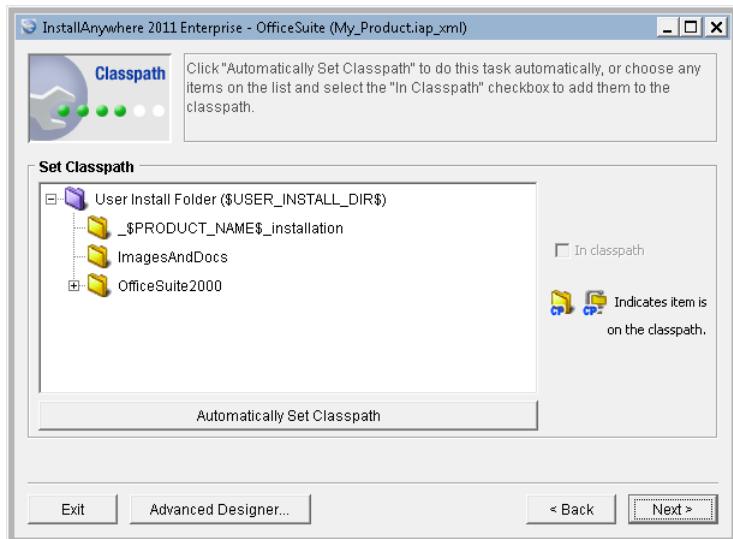


Figure 7-59: Classpath Frame of the Project Wizard

Clicking the Automatically Set Classpath button will accomplish this task for the installer project.

Build Installer

The Build Installer frame selects the target installation platforms for the creation of installers.

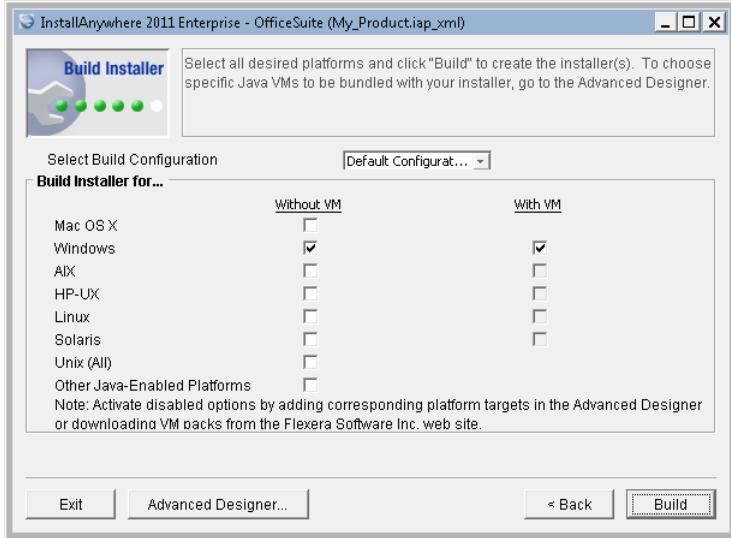


Figure 7-60: Build Installer Frame of the Project Wizard

Java Virtual Machines (VMs) may also be selected to be bundled with the installer. Once the platforms and the VMs have been selected, click Build to build the installers.

The Build Installer frame of the InstallAnywhere Wizard includes the following options:

Table 7-79 • Build Installer Frame

Option	Description
Select Build Configuration	From this list, select the Build Configuration that you want to build.
Build Installer for ...	The following platforms are listed: <ul style="list-style-type: none">• Mac OS X• Windows• AIX• HP-UX• Linux• Solaris• Unix (All)• Other Java-Enabled Platforms

Table 7-79 • Build Installer Frame

Option	Description
Without VM	Select the check box in this column for the platforms that you want to build without bundling a Java virtual machine with the build target.  Note • <i>Installers that are built without VMs are smaller and download faster than installers bundled with one.</i>
With VM	Select the check box in this column for the platforms that you want to build that will have a Java virtual machine of that platform bundled with the build target.
Exit	Click to exit InstallAnywhere.
Advanced Designer	Click to switch to the Advanced Designer interface.
Back	Click to return to the Classpath panel of the Project Wizard.
Build	Click to build the selected Build Configuration.

Try Installer

The Try Installer frame lets developers test the Web installer that the Project Wizard builds.

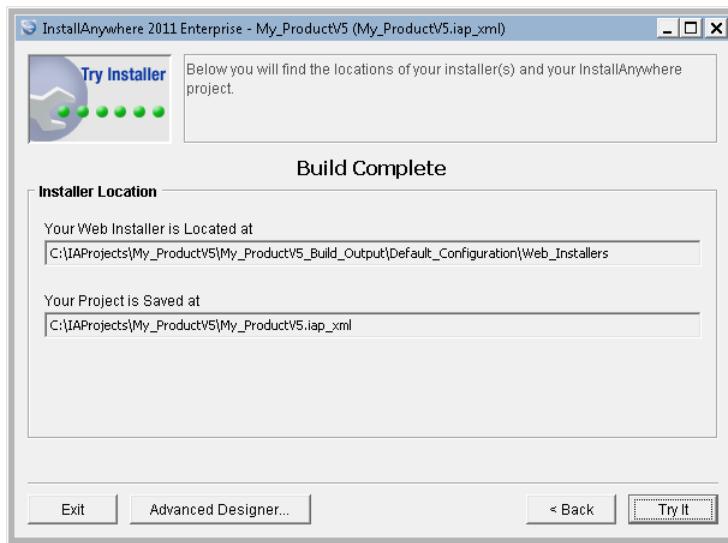


Figure 7-61: Try Installer Frame of the Project Wizard

Clicking the Try It button launches the systems preferred browser and actually installs the product on the development machine.

Dialog Boxes

InstallAnywhere includes several common dialog boxes that users commonly see when starting a new project, opening an existing project, and so on.

Table 7-80 • InstallAnywhere Dialog Box Reference

Dialog Box	Description
About Dialog Box	Shows information about your InstallAnywhere application: version number, product code, and copyright. The About dialog box also includes a button to open the InstallAnywhere Preferences dialog box.
Add Files to Project Dialog Box	Use to add files or directories of files to your project.
Advertise Variables Dialog Box	When building a Merge Module installer, use this dialog to specify which variables in the Merge Module can be set by the parent installer and which variables in the parent installer can be set by the Merge Module.
Configure Input Items Dialog Box	Provides controls to fully describe the contents of the Get User Input panel.
Change Disk Space and Name Dialog Box	Specify how the contents of a CD-ROM/DVD installer is split into multiple media volumes (CDs, DVDs).
Choose an Action Dialog Box	<p>Lists available actions in the following tabs:</p> <ul style="list-style-type: none"> • General • Panels • Consoles • System i (i5/OS) • Install • Uninstall • Plug-Ins  <p>Note • This dialog box is customized to display those actions appropriate to the Advanced Designer task that you are performing (Pre-Install, Install, Post-Install, Pre-Uninstall, Uninstall, Post-Uninstall); therefore, the number of tabs and actions displayed will vary.</p>
Choose Icon Dialog Box	On this dialog box, which is opened by clicking the Change button on the Create LaunchAnywhere for Java Application customizer, you can choose a different icon for the launcher.

Table 7-80 • InstallAnywhere Dialog Box Reference (cont.)

Dialog Box	Description
Choose Variable Dialog Box	Instead of having to manually type in the name of variables in text boxes or text areas in your installation project, you can open the Choose Variable dialog box and select a variable from a list.
Configure Search and Replace Strings Dialog Box	When a Modify Text File action runs, you can instruct the installer to search for specific text strings in the text file(s) and replace them with replacement strings. You specify the search and replacement text strings on this dialog box.
Create New Project Dialog Box	Available from the Advanced Designer interface, Create New Project is the dialog box you see when you click the New Project command in the File menu.
Create VM Pack Dialog Box	You can use this dialog box, which is opened when you select Create VM Pack on the File menu of the Advanced Designer, to create a VM pack for any platform.
Custom Rules Dependencies Dialog Box	On this dialog box, which is opened by clicking Configure Dependencies on the Evaluate Custom Rule customizer, you can select a .JAR or .ZIP file which contains classes referenced by your custom rule so that those classes are included in the archive and are available to the rule at runtime.
Edit Advertised Variables Dialog Box	When adding an Install Merge Module action to your installation project, you use this dialog box to specify which variables in the Merge Module can be set by the parent installer and which variables in the parent installer can be set by the Merge Module.
Filter File Dialog Box	Use to define conditions that the installer will use to select files for modification when performing a Modify Text File - Multiple Files action.
InstallAnywhere Preferences Dialog Box	Provides access to InstallAnywhere preference settings in six tabs: General Settings, Resources, Source Paths, Updates, and Designer Colors.
LaunchAnywhere Properties Dialog Box	Use to edit the Name , Value , and Comment of any of the selected LaunchAnywhere launcher's properties.
Manage Instances Dialog Box	This dialog box opens when an end-user launches the installer for a product which has already been installed using a Instance Management-enabled installation. The user is prompted to choose to either Install a New Instance or Modify an Existing Instance by choosing an instance from the list.
New Project Dialog Box	The first dialog box of the Project Wizard interface, New Project allows you to specify settings for a new project and choose whether to continue in the Project Wizard or open the new project in the Advanced Designer. The New Project dialog box includes an Open Existing Project option. With this option selected, the New Project dialog box becomes the Open Project dialog box.

Table 7-80 • InstallAnywhere Dialog Box Reference (cont.)

Dialog Box	Description
Open Project Dialog Box	Available from the New Project dialog box, Open Project appears when you click the Open Existing Project option on the New Project dialog box. Open Project provides a list of recent projects and controls for selecting other existing projects.
Open Project File Dialog Box	Available from the Advanced Designer, this dialog appears when you click Open Project from the File menu.
Read/Modify XML File Dialog Box	Available from the Read/Modify XML File action customizer, select the XML file that will be installed on the target system that you want to read or modify.
RPM Specification Settings Dialog Box	On supported Linux systems, RPM registration creates a virtual package and uses it to make entries to the RPM database. Define RPM registration settings on this dialog box.
Save New Project As Dialog Box	Available from the New Project and Create New Project dialog boxes, this dialog box allows you to specify basic settings for the creation of the new project (file name, folder name, and so on).
Search Results Dialog Box	You can use this dialog box, which is opened by selecting Search on the File menu, to search your entire installation project for a specific variable or Build Configuration Tag. You can also perform global replacements of variables.
SWVPD Registry Settings Dialog Box	On this dialog box, you can choose to include AIX registry (Software Vital Product Data) support in your project's installers.
Uninstaller Properties Dialog Box	Use to edit the Name , Value , and Comment of any of the selected Uninstaller launcher's properties.

About Dialog Box

The About dialog box shows the InstallAnywhere application version number, product code, and copyright information. This dialog also provides a **Preferences** button to open the **InstallAnywhere Preferences** dialog box.

- The About dialog box can be displayed from the Project Wizard by clicking **About** on the New Project or Open Project dialog boxes.
- The About dialog box can be displayed from the Advanced Designer by clicking **Help > About InstallAnywhere**.

Click **OK** to close the About dialog box.

Add Files to Project Dialog Box

The **Add Files to Project** dialog box opens when you click **Add Files** on the **Install** task of the Advanced Designer or on the **Add Files** panel of the Project Wizard. In this dialog box, you can add files or directories to your project.

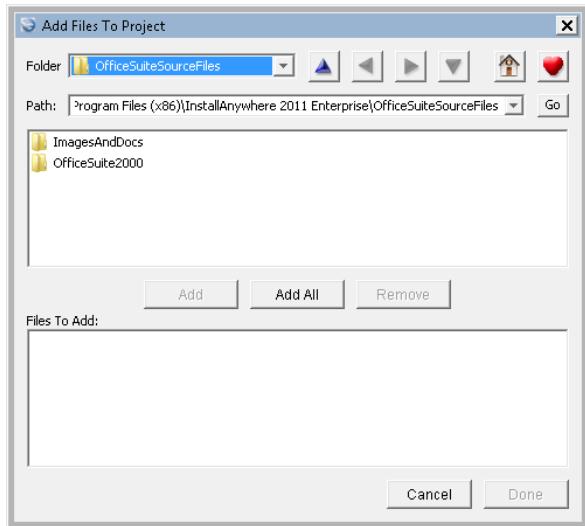


Figure 7-62: Add Files to Project Dialog Box

The Add Files To Project dialog box includes the following options:

Table 7-81 • Add Files to Project Dialog Box Options

Option	Description
Folder	The directory that is selected in this list controls which subdirectories are listed below the Path field. You can click the arrow and select a different directory from the list.
Path	Lists the full path of the directory selected in the Folder list. You can click the arrow and select a different directory from the list. You can also enter a directory path in the Path field and click Go to open that directory.
	Use to jump to different directory in the Folder list. <ul style="list-style-type: none"> • Up arrow—Go up one directory level. • Left arrow—Go back to the previous directory. • Right arrow—Go forward one directory. • Down arrow—Open the selected directory.
	Click to open the user-specific Home directory which, in Windows is: C:\Documents and Settings\User_Name

Chapter 7: Reference

Dialog Boxes

Table 7-81 • Add Files to Project Dialog Box Options (cont.)

Option	Description
	Click to open the Favorites menu, where you can choose to jump to a previously identified favorite directory (by selecting it from the list) or to create a new favorite directory (by selecting Add Folder to Favorites).
Go	Enter a directory path in the Path field and click Go to open that directory.
Add	To populate the Files to Add list, select a directory or directories and click the Add button. The selected directories will then be listed in the Files to Add list.
Add All	To populate the Files to Add list with all of the directories listed below the Path field, click Add All .
Remove	Click to remove the selected directories from the Files to Add list.
Files to Add	Files and directories that you have selected are listed in this list.  Note • This type of file linking adds a static list of files to your project: any files you add later to this source directory will not automatically be added to your project. To specify a directory from which InstallAnywhere should regenerate a dynamic list of source files during each build, you can use the Install SpeedFolder action.
Cancel	Click to cancel the operation without adding any files to the project.
Done	Click to add the selected files and directories to the project.

Advertise Variables Dialog Box

When building a Merge Module installer, you need to specify which variables in the Merge Module can be set by the parent installer and which variables in the parent installer can be set by the Merge Module. You do this on the **Advertise Variables** dialog box, which is opened by clicking **Advertise Variables** on the **Project > Variables** subtask.

Advertised variables are used to inform the parent installer of settings required for a Merge Module's configuration, and to return information—such as status or return codes—back to the parent installer. Advertised variables must be set before a Merge Module can be installed.

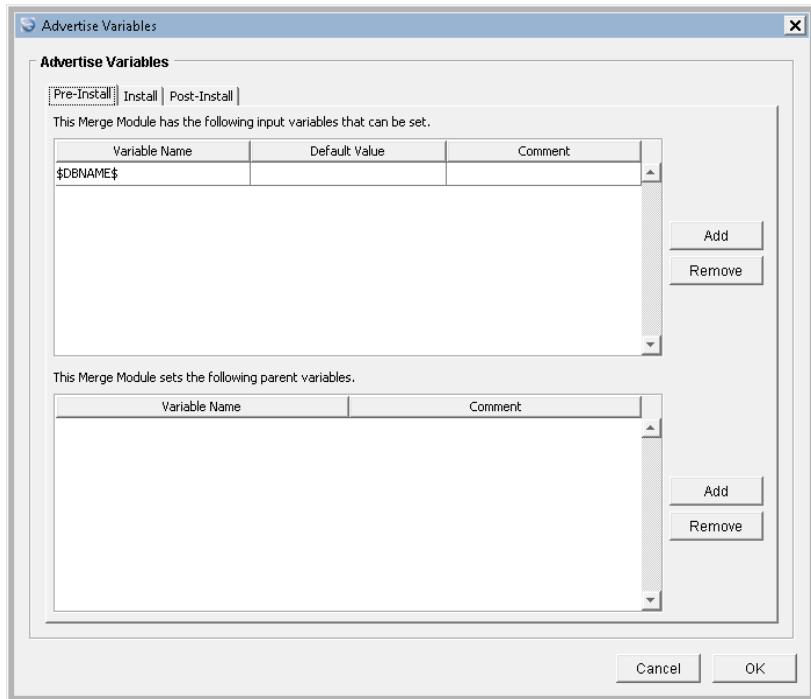


Figure 7-63: Advertise Variables Dialog Box

Chapter 7: Reference

Dialog Boxes

The Advertise Variables dialog box includes the following options:

Table 7-82 • Advertise Variables Dialog Box

Option	Description
Pre-Install, Install, Post-Install Tabs	Select the appropriate tab to indicate the task that you want to advertise variables for: Pre-Install , Install , or Post-Install .  Note • Advertised variables are defined by task: <ul style="list-style-type: none">• Variables set as advertised as Pre-Install will only affect the Dynamic Merge Module in Pre-Install.• The Install Merge Module action only pays attention to Advertised Variables set in the Install task.
Input Variables List	To advertise an input variable so that it will be visible in the parent project (the project that is going to install this Merge Module), click Add next to the input variables list to add an entry to the list, and then enter the following information: <ul style="list-style-type: none">• Variable Name—Enter the name of a variable that has been defined in one of the panels of this project, using the syntax \$VARIABLE_NAME\$.• Default Value—If you want to hard code a value for this variable, enter text here. If you want the value of this variable to be entered by the end user during installation, leave this field blank.• Comment—Enter text to describe the purpose of this variable. Click Remove to remove a selected variable from the list.
Parent Variables List	To advertise a variable in the parent project (the project that is going to install this Merge Module) so that it will be visible in this project, click Add next to the parent variables list to add an entry to the list, and then enter the following information: <ul style="list-style-type: none">• Variable Name—Enter the name of a variable that you want to pass to the parent project using the syntax \$VARIABLE_NAME\$.• Comment—Enter text to describe the purpose of this variable. Click Remove to remove a selected variable from the list.



Note • For each Build Configuration, you can specify if you want to create a Merge Module installer as part of the build output by opening the **Build** task and selecting the **Build Merge Module/Template** option on the **Distribution** subtab of the **Build Configurations** tab.

Configure Input Items Dialog Box

The **Configure Input Items** dialog box, which is opened by clicking **Configure** on the customizer of the **Get User Input - Simple** panel action, provides controls to fully describe the contents of the **Get User Input** panel.

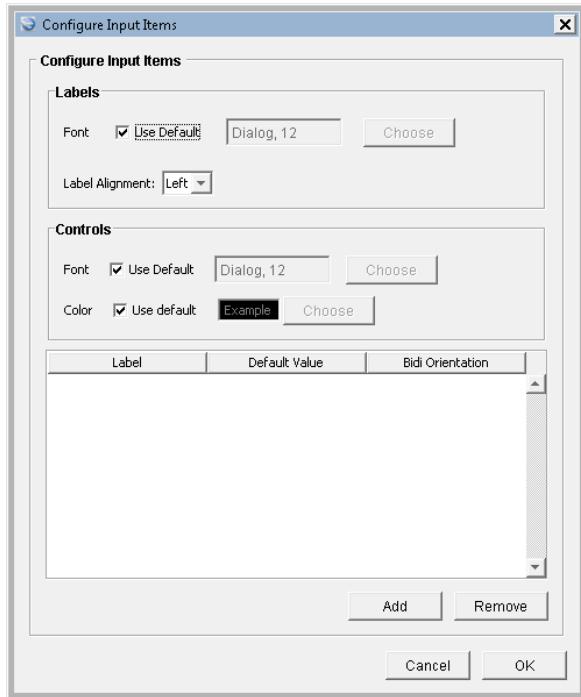


Figure 7-64: Configure Input Items Dialog Box

You can use the **Configure Input Items** dialog box to:

- Add or remove controls of the type set in **Input Method**.
- Define labels and default values for each control.
- Specify the text orientation and text reading direction for each control.
- Set, for all panel controls, the typeface, size, text color, and background color.



Note • The specific settings available on the **Configure Input Items** dialog box depend on the **Input Method** that is selected on the Get User Input panel customizer.



Note • Labels may be literal values or values represented by InstallAnywhere variables (which will resolve prior to the panel being displayed).

Change Disk Space and Name Dialog Box

On the **Change Disk Space and Name** dialog box, you specify how the contents of a CD-ROM/DVD installer is split into multiple media volumes (CDs, DVDs).

You open this dialog box from the Advanced Designer **Build** task by opening the **Distribution** subtab of the **Build Configurations** tab and clicking **Change Disk Space and Name**.

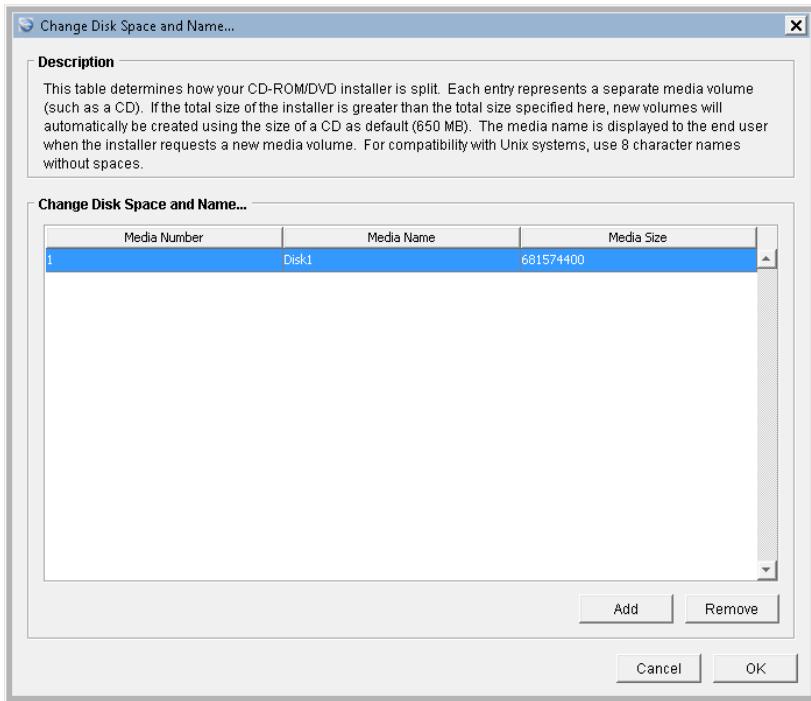


Figure 7-65: Change Disk Space and Name Dialog Box

The **Change Disk Space and Name** dialog box includes the following controls and options:

Table 7-83 • Change Disk Space and Name Dialog Box

Option	Description
Media Number	Number that uniquely identifies the media volume. (Read only)
Media Name	<p>Enter a name that describes the contents of the media volume. This name is displayed to the end user when the installer requests a new media volume.</p> <p></p> <p>Note • For compatibility with Unix systems, enter an 8-character name that does not include any spaces.</p>

Table 7-83 • Change Disk Space and Name Dialog Box

Option	Description
Media Size	Enter the maximum size of the media volume. The default value is approximately 650 MB (681574400 bytes). 
	Note • If the total size of the installer is greater than the total size specified on the combined media volumes, new volumes will automatically be created
Add	Click to add a new media volume to the list.
Remove	Click to remove the selected media volume from the list.

Choose an Action Dialog Box

The **Choose an Action** dialog box only presents actions which are available for the task within which the developer is working. For example:

- If you click **Add Action** from within the **Install** task, the **Choose an Action** dialog box does not provide tabs for panels or consoles.
- For the **Install** task, the **Choose an Action** dialog box may show just two tabs: a tab for **Install** actions and a tab for **General** actions.

Actions under the **General** tab are available from all Pre- and Post-Install (and Uninstall) tasks as well. The **Pre-** and **Post-Install/Uninstall** tasks have actions listed under the **Panels** and **Consoles** tabs.

Table 7-84 • Choose an Action Dialog Box Controls

Tab	Description	Advanced Designer Tasks
General	Lists general actions.	<ul style="list-style-type: none"> • Pre-Install • Install • Post-Install • Pre-Uninstall • Uninstall, • Post-Uninstall
Install	Lists actions restricted to the Install task.	<ul style="list-style-type: none"> • Install
Panels	Lists available panel actions. Panels actions accept user input and provide feedback during these tasks.	<ul style="list-style-type: none"> • Pre-Install • Post-Install • Pre-Uninstall • Post-Uninstall

Table 7-84 • Choose an Action Dialog Box Controls (cont.)

Tab	Description	Advanced Designer Tasks
Consoles	Console actions serve the same purpose as panel actions for console (non-GUI, non-silent) installers.	<ul style="list-style-type: none"> • Pre-Install • Post-Install • Pre-Uninstall • Post-Uninstall
System i (i5/OS)	Lists actions specific to the System i (i5/OS) platform. (System i actions are only available in InstallAnywhere Enterprise edition.)	<ul style="list-style-type: none"> • Pre-Install • Install • Post-Install • Pre-Uninstall • Post-Uninstall
Plug-Ins	<p>Shows plug-in actions, if any are installed.</p>  <p>Note • The Plug-ins tab appears when a custom code plug-in has been added.</p>	<ul style="list-style-type: none"> • Pre-Install • Install • Post-Install • Pre-Uninstall • Post-Uninstall



Note • For information on the individual actions listed on the **Choose an Action** dialog box, see [Actions](#).

Choose Icon Dialog Box

On the **Choose Icon** dialog box, which is opened by clicking the **Change** button on the **Create LaunchAnywhere for Java Application** customizer, you can choose a different icon for the launcher.

- Windows Tab
- Mac OS X Tab

Windows Tab

On the **Windows** tab, you are prompted to select an icon file (.ico), a 32x32 GIF file, and a 16x16 GIF file.

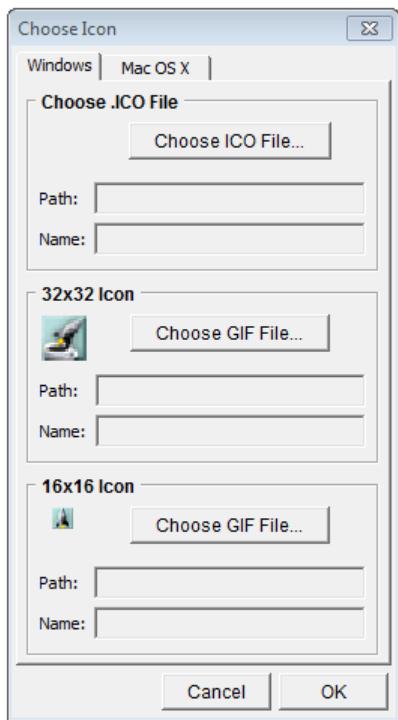


Figure 7-66: Choose Icon Dialog Box

The **Windows** tab of the **Choose Icon** dialog box includes the following controls:

Table 7-85 • Windows Tab of Choose Icon Dialog Box

Control	Description
Choose .ICO File	<p>Click Choose ICO File and select a Windows icon file (.ico). After a file is selected, the Path and Name fields will be populated.</p> <p></p> <p>Note • The Choose Icon dialog box does not display a preview of the selected .ico file.</p>
32x32 Icon	<p>Click Choose GIF File and select a GIF (.gif) file that is 32x32 pixels. After a file is selected, the Path and Name fields will be populated.</p>
16x16 Icon	<p>Click Choose GIF File and select a GIF (.gif) file that is 16x16 pixels. After a file is selected, the Path and Name fields will be populated.</p>

Mac OS X Tab

The Mac OS X tab of the **Choose Icon** dialog box includes the following controls:

Table 7-86 • Mac OS X Tab of Choose Icon Dialog Box

Control	Description
Mac OS X Icon File (.icns)	Click Choose ICNS File and select a Mac OS X Icon Resource File (.icns). After a file is selected, the Path and Name fields will be populated.

Choose Variable Dialog Box

Instead of having to manually type in the name of variables in text boxes or text areas in your installation project, you can open the **Choose Variable** dialog box and select a variable from a list. While you are editing the contents of a text box or a text area, you can open the **Choose Variable** dialog box by pressing Alt + V.

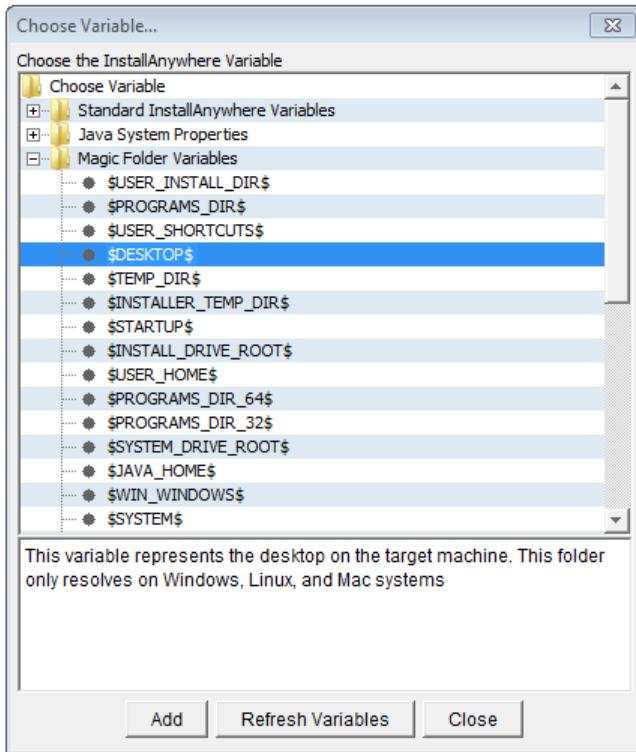


Figure 7-67: Choose Variable Dialog Box

The **Choose Variable** dialog box lists all of the variables available in your project, grouped by category, with embedded help for the selected variable. Whenever you add a new variable to your installation project, the **Choose Variable** dialog box is refreshed to include it. Variables are listed in the following categories:

- Standard InstallAnywhere Variables

- Java System Properties
- Magic Folder Variables
- User Defined Magic Folders
- User Defined Variables

Click **Add** to add the selected variable to the currently active text box. Click **Refresh Variables** to manually refresh the variable list.

The **Choose Variable** dialog box remains open until you click the **Close** button.

Configure Search and Replace Strings Dialog Box

When a Modify Text File action runs, you can instruct the installer to search for specific text strings in the text file(s) and replace them with replacement strings.

On the **Configure Search and Replace Strings** dialog box, which is opened by clicking **Configure** on a **Modify Text File** action customizer, you specify the search and replacement text strings.

Click **Add** to add an entry to the **Configure Search and Replace Strings** list, and enter the **Search For** and **Replace With** text. When the installer runs, it will search for all instances of the **Search For** text and replace them with their corresponding **Replace With** text.

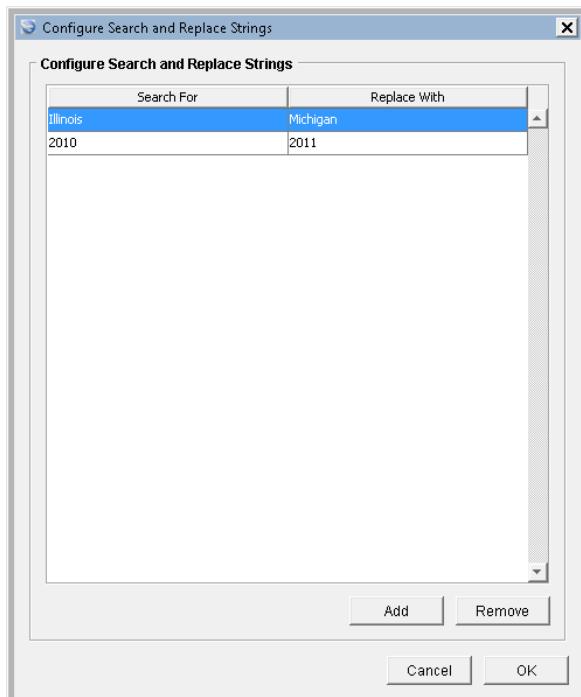


Figure 7-68: Configure Search and Replace Strings Dialog Box

Create New Project Dialog Box

Available from the Advanced Designer interface, the Create New Project dialog box allows you to create a new project based on a standard or custom template.



Note • In InstallAnywhere Enterprise Edition, a template must be selected for the new project. For a first project, use the Basic Project template.

Table 7-87 • Dialog Box Controls

Controls	Description
Template list	Lists the standard InstallAnywhere project templates:
Other Template	Opens the Choose Template dialog box. There, you can select a custom template (iam.zip).
Save As	Opens the Save New Project As dialog box. This dialog box is where you name new projects.
Cancel	Dismisses the Create New Project dialog box without creating a new project.
OK	Saves the new project using the settings you provide and closes the Create New Project dialog box.

Create VM Pack Dialog Box

You can use the **Create VM Pack** dialog box, which is opened when you select **Create VM Pack** on the **File** menu of the Advanced Designer, to create a VM pack for any platform.

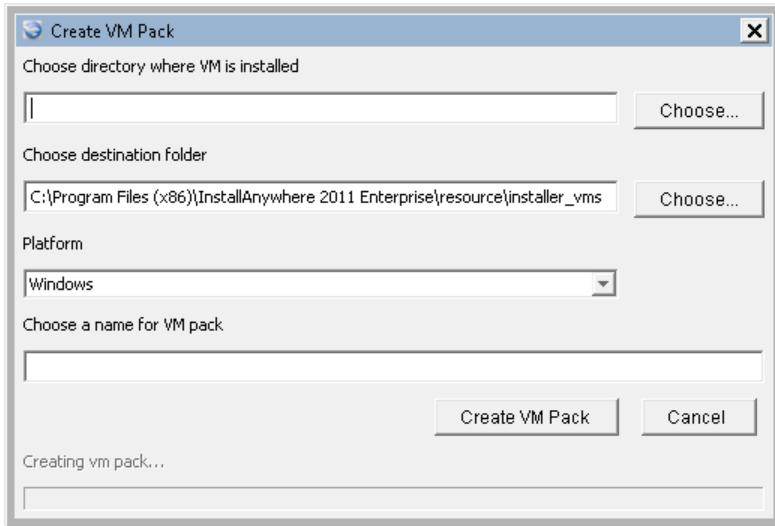


Figure 7-69: Create VM Pack Dialog Box



Note • You can also use the VM Pack Utility as a standalone tool on a machine where InstallAnywhere is not installed. See [Running the VM Pack Utility as a Standalone Tool](#).

Create VM Pack Dialog Box Controls

The **Create VM Pack** dialog box includes the following controls:

Table 7-88 • Create VM Pack Dialog Box

Control	Description
Choose directory where VM is installed	Click Choose and select the directory where the VM is installed. Important • The VM must be located in a directory named <i>jre</i> . See Preparing a VM Pack for Use: "jre" Directory Name Requirement for more information.
Choose destination folder	Click Choose and select the directory where you want to store the new VM pack.

Table 7-88 • Create VM Pack Dialog Box

Control	Description
Platform	Choose the platform of this new VM pack from the list. Available options are: <ul style="list-style-type: none"> • Windows • Solaris • HP-UX • AIX • Linux
Choose a name for VM pack	Enter a name to uniquely identify the new VM pack.
Create VM Pack	Click to create the VM pack using the specified information.

Preparing a VM Pack for Use: “jre” Directory Name Requirement

In order to create a VM Pack using the **Create VM Pack** utility, the VM that you choose must be located in a directory named **jre**. If the VM that you want to use is in a directory that is named **jre**, you do not need to take any further steps.

However, if the VM that you want to use is in directory named something other than **jre**—such as in the following example, where the directory is named **jre1.6.0_07**—you need to copy the entire contents of the directory to a directory named **jre** and then select that location for the **Choose directory where VM is installed** field on the **Create VM Pack** dialog box.

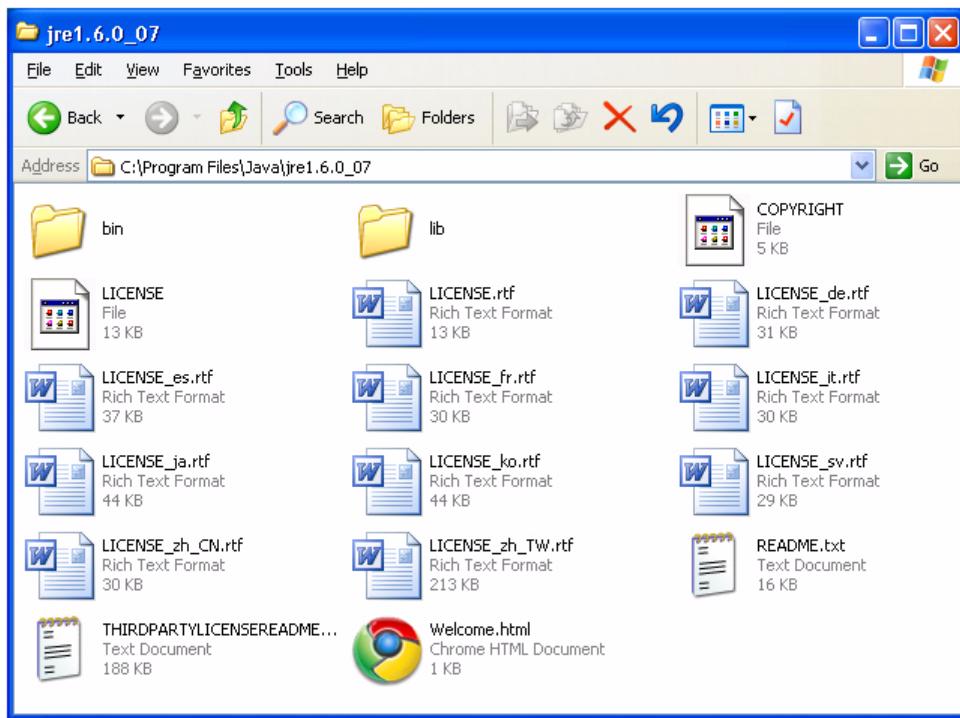


Figure 7-70: VM in Directory Named jre1.6.0_07



Important • It is recommended that you copy of the contents of the directory to a new directory named *jre* instead of renaming the existing directory.

Running the VM Pack Utility as a Standalone Tool

You can also use the VM Pack utility as a standalone tool on a machine where InstallAnywhere is not installed.



Task:

To run the Create VM Pack utility as a standalone tool:

1. Copy the VMPackUtility directory, which is found in the InstallAnywhere installation directory, to the machine on which the JDK/JRE installation is present:
2. Execute the utility using the following command line:

```
$ > java -jar Utility.jar
```

A wizard which automates the process of creating a VM pack opens.

Custom Rules Dependencies Dialog Box

On the **Custom Rules Dependencies** dialog box, which is opened by clicking **Configure Dependencies** on the **Evaluate Custom Rule** customizer, you can select a .JAR or .ZIP file which contains classes referenced by your custom rule so that those classes are included in the archive and are available to the rule at runtime.

Edit Advertised Variables Dialog Box

When adding an **Install Merge Module** action to your installation project, you need to specify which variables in the Merge Module can be set by the parent installer and which variables in the parent installer can be set by the Merge Module. You do this on the **Edit Advertised Variables** dialog box, which is opened by clicking **Edit Variables** on the customizer of an **Install Merge Module** action on the **Install** task.

Advertised variables are used to inform the parent installer of settings required for a Merge Module's configuration, and to return information—such as status or return codes—back to the parent installer. Advertised variables must be set before a Merge Module can be installed.

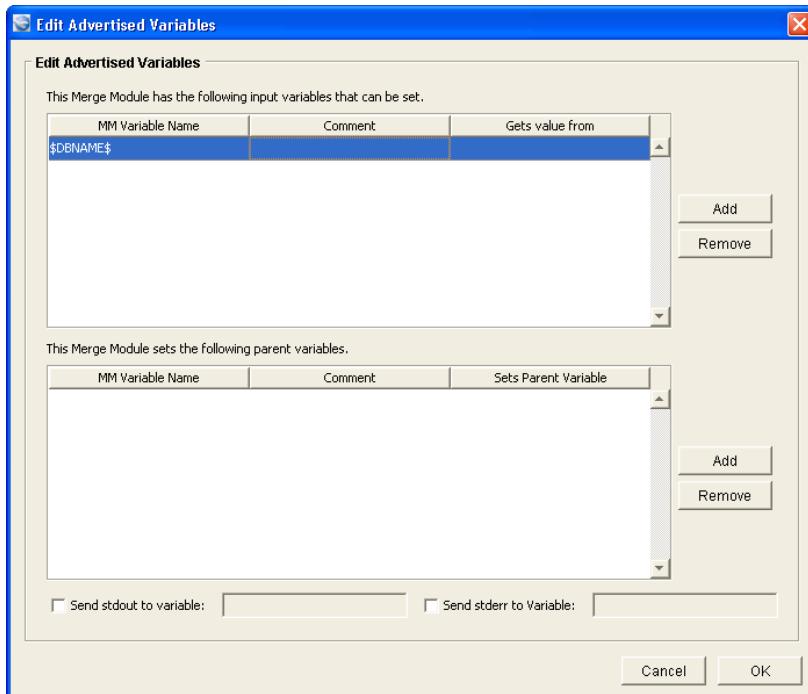


Figure 7-71: Edit Advertised Variables Dialog Box

The **Edit Advertised Variables** dialog box includes the following options:

Table 7-89 • Edit Advertised Variables Dialog Box

Option	Description
Input Variables List	<p>To advertise an input variable in the Merge Module so that it will be visible in the parent project (the project that is going to install this Merge Module), click Add next to the input variables list to add an entry to the list, and then enter the following information:</p> <ul style="list-style-type: none"> • MM Variable Name—Enter the name of a variable that has been defined in the selected Merge Module, using the syntax \$VARIABLE_NAME\$. • Comment—Enter text to describe the purpose of this variable. • Gets value from—Enter the name of a variable that has been defined in one of the panels in this project using the syntax \$VARIABLE_NAME\$. <p>Click Remove to remove a selected variable from the list.</p>
Parent Variables List	<p>To advertise a variable in the parent project (the project that is going to install this Merge Module) so that it will be visible in this Merge Module, click Add next to the parent variables list to add an entry to the list, and then enter the following information:</p> <ul style="list-style-type: none"> • MM Variable Name—Enter the name of a variable that has been defined in the selected Merge Module using the syntax \$VARIABLE_NAME\$. • Comment—Enter text to describe the purpose of this variable. • Sets Parent Variable—Enter the name of a variable that has been defined in one of the panels of this project using the syntax \$VARIABLE_NAME\$. <p>Click Remove to remove a selected variable from the list.</p>
Send stdout to variable Send stderr to Variable	<p>To set the stderr and stdout of the selected Merge Module to InstallAnywhere variables, select these options and enter a variable name in the box. This is useful if you want to debug a Merge Module.</p>

Filter File Dialog Box

On the **Filter File** dialog box, you can define conditions that the installer will use to select files for modification when performing a **Modify Text File - Multiple Files** action.

The **Filter File** dialog box is opened by clicking **Configure Filter** on the **Modify Text File - Multiple Files** action customizer.

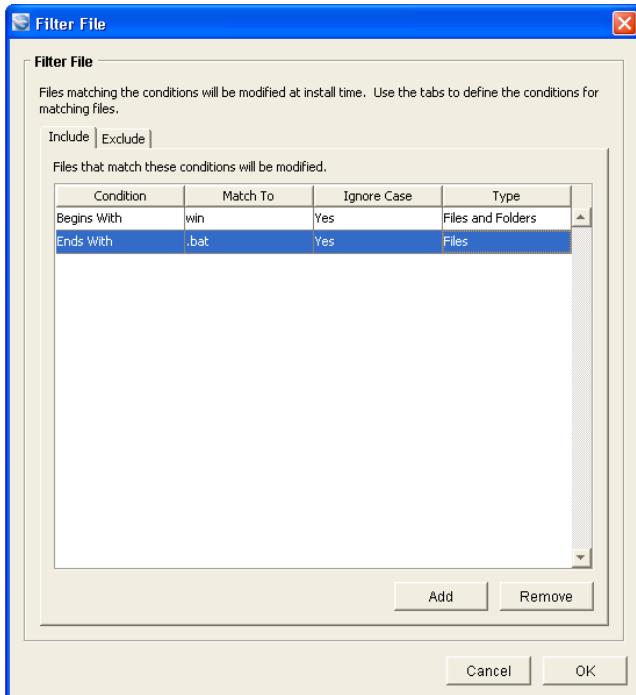


Figure 7-72: Filter File Dialog Box

You can create conditions to both include or exclude specific files. Click **Add** to add a condition to the list on either the **Include** or the **Exclude** tab.

- **Include tab**—Files matching the defined conditions will be modified at install time.
- **Exclude tab**—Files matching the defined conditions will not be modified at install time. Exclude conditions override all include conditions

When building conditions, you have the following options:

Table 7-90 • Filter Options

Option	Description
Condition	Select an option to specify how the text entered in the Match To box should be used to filter the list of files to change: <ul style="list-style-type: none">• Begins With• Ends With• Contains• Regular Expression
Match To	Enter a text string to search for in the file name.
Ignore Case	Set to either Yes or No to indicate whether case should be considered when searching for the Match To text.
Type	Set to either Files , Folders , or Files and Folders .

InstallAnywhere Merge Module Import Assistant Dialog Box

On the **InstallAnywhere Merge Module Import Assistant** dialog box, which is opened when you click **Import Merge Module** on the **Organization > Modules** subtask, you can customize what aspects of the selected Merge Module will be directly imported and integrated into the currently open project.

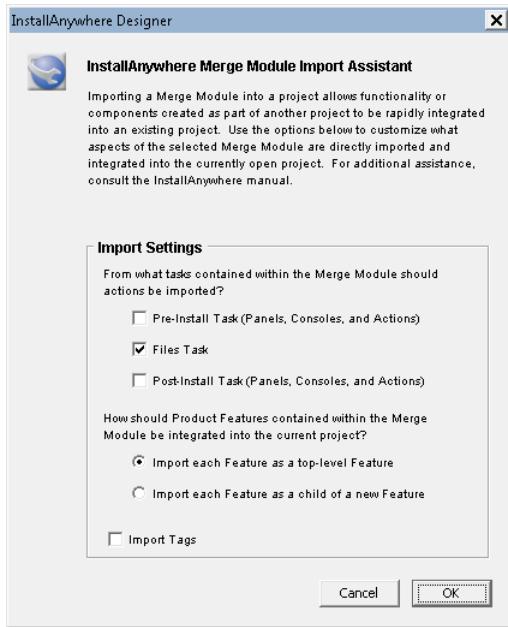


Figure 7-73: InstallAnywhere Merge Module Import Assistant

The InstallAnywhere Merge Module Import Assistant dialog box includes the following controls:

Table 7-91 • InstallAnywhere Merge Module Import Assistant Dialog Box

Control	Description
From what tasks contained within the Merge Module should actions be imported?	Select one or more of the following options to specify the tasks in the merge module from which actions should be imported: <ul style="list-style-type: none">• Pre-Install Task (Panels, Consoles, and Actions)• Files Task• Post-Install Task (Panels, Consoles, and Actions)
How should Product Features contained within the Merge Module be integrated into the current project?	Select one of the following options to specify how product features in the merge module should be integrated into the current project: <ul style="list-style-type: none">• Import each Feature as a top-level Feature• Import each Feature as a child of a new Feature

Table 7-91 • InstallAnywhere Merge Module Import Assistant Dialog Box

Control	Description
Import Tags	Select this option to import the Build Tags defined in the imported merge module into this project.

InstallAnywhere Preferences Dialog Box

Use the **InstallAnywhere Preferences** dialog box to customize the InstallAnywhere development environment. The following tabs provide specific options. (This dialog box is available from the Edit menu and from the About dialog box.)

Table 7-92 • Dialog Box Controls

Tab	Description
General Settings	Sets preferences for InstallAnywhere default interface (Advanced Designer or Project Wizard), project loading, command-line builds, manifest files, and the default Install task view (Components or Product Features).
Resources	Sets resource paths for plug-ins and VM packs.
Source Paths	Enables or disables source paths, allows you to add and remove source paths, enables or disables individual default source paths: \$IA_PROJECT_DIR\$, \$IA_HOME\$, \$USER_HOME\$.
Updates	Defines communication settings for automated updates and participation in the Customer Experience Improvement Program.
Designer Colors	Sets preferences for the color scheme InstallAnywhere will use.

General Settings

Use the **General Settings** tab to select default settings for InstallAnywhere.

Table 7-93 • General Settings Tab of the InstallAnywhere Preferences Dialog Box

Option	Description
InstallAnywhere Startup	This setting determines whether or not the InstallAnywhere Project Wizard opens when you start InstallAnywhere. <ul style="list-style-type: none"> • Project Wizard—Choose if you want the Project Wizard to appear when you start InstallAnywhere. • Advanced Designer—Choose if you want the Advanced Designer to appear when you start InstallAnywhere.

Table 7-93 • General Settings Tab of the InstallAnywhere Preferences Dialog Box

Option	Description
Project Loading	<p>This setting determines whether InstallAnywhere should verify the location of files referenced in the Install task when a project opens.</p> <ul style="list-style-type: none"> • Always (recommended)—Choose this option if you want InstallAnywhere to verify the location of files referenced in the Install task when a project opens. • Never—Choose this option if you want to allow a project to open without InstallAnywhere verifying the location of files referenced in the Install task.
Command Line Builds	<p>This setting determines whether a project built from the command line can successfully build if referenced files are missing.</p> <ul style="list-style-type: none"> • Fail (recommended)—Choose if you want a project built from the command line to fail if referenced files are missing. • Continue without including the missing files—Choose to allow a project built from the command line to continue building with referenced files that are missing.
Multiple Build Configuration settings	<p>Specify the desired action to occur if an error is encountered when building multiple Build Configurations:</p> <ul style="list-style-type: none"> • Stop Building—Stop the entire project build. • Continue Building—Continue building the remaining Build Configurations.
Manifest Files	<p>InstallAnywhere uses manifest files to identify files that are included in the installer. Manifest files are useful when groups of developers are working on a project in separate locations.</p> <p>This setting determines whether a project that includes a manifest file can successfully build if files referenced in the manifest file are missing.</p> <ul style="list-style-type: none"> • Fail (recommended)—Choose if you want a project that includes a manifest file to fail if files referenced in the manifest file are missing. • Continue without including the missing files—Choose to allow a project that includes a manifest file to continue building with files referenced in the manifest file that are missing.
Default Install Task View	<p>This setting determines whether the default view for the Install task displays Actions assigned to Components or Product Features. Choose one of the following options:</p> <ul style="list-style-type: none"> • Components—Choose if you want the Install task to display Actions assigned to Components by default. • Product Features—Choose if you want the Install task to display Actions assigned to Product Features by default.

Resources

Use the Resources tab to provide semi-colon delimited resource paths for plug-ins, VM packs, and JVM spec files. When you specify additional paths, you can use both full path references and pre-defined source path variables.



Note • Remember to use proper notation when employing source path variables. For example:

`IA_HOME/my_vms or $TEAM_SHARE$/plugins`

The **Resource** tab includes the following properties:

Table 7-94 • Resources Tab of the InstallAnywhere Preferences Dialog Box

Property	Description
Plugin Resource Paths	<p>InstallAnywhere scans the paths listed here and populates the Plug-Ins tab with the plug-ins it finds on those paths as well as those stored at the standard plug-in location:</p> <p><code><InstallAnywhere>\plugins\</code></p> <p>Note • The plug-in list in the Choose an Action dialog box is updated every time the dialog box opens.</p>
VM Pack Resource Paths	<p>InstallAnywhere scans the paths listed here and populates the VM to Bundle with Installer List with VM packs it finds on those paths as well as those stored at the standard VM pack location:</p> <p><code><InstallAnywhere>\resource\installer_vms\</code></p> <p>Note • The VM packs listed in the VM list on the Build Targets tab (VM to Bundle with Installer) are refreshed each time you edit preferences or start InstallAnywhere. Therefore, you cannot simply copy a VM pack into a valid VM pack path. You must first open and close the Preferences dialog box or restart InstallAnywhere.</p>

Chapter 7: Reference

Dialog Boxes

Table 7-94 • Resources Tab of the InstallAnywhere Preferences Dialog Box

Property	Description
JVM Specs Resource Paths	<p>By default, all JVM spec files are installed in the <IA_HOME>\resource\jvms directory. However, JVM spec files can be present at any location on the machine where InstallAnywhere is installed.</p> <p>In cases where a JVM spec file is located in a directory other than <IA_HOME>\resource\jvms, you need to identify the location of those JVM spec files in the JVM Specs Resource Paths field. In this field, enter a semicolon-separated path list where JVM spec files are located.</p> <p>InstallAnywhere scans the paths listed here and populates the Choose JVM Spec dialog box with JVM spec files it finds on these paths as well as those stored at the default JVM spec file location (<IA_HOME>\resource\jvms).</p>  <p>Note • <i>The organization of the JVM spec files must be similar to that of \$IA_HOME\$/resource/jvms directory; each spec file must be placed in its own platform folder, such as:</i></p> <ul style="list-style-type: none"> aix generic-unix hpux linux solaris unix windows

Source Paths

Source paths are special variables that represent common paths to your project's files. When you add a file, if the path to the file matches one of the values listed here, the source path variable name is substituted in place of the matching path. If you move your project and source files to a different folder or computer, InstallAnywhere can find all of the files by simply updating these source paths.

Use the **Source Paths** tab to define source paths as variables for installer projects.

Table 7-95 • Source Paths Tab of the InstallAnywhere Preferences Dialog Box

Option	Description
Enable Source Paths	Select this option to enable you to add or remove source paths.
Source Path List	<p>Listing of all defined source paths. Information includes:</p> <ul style="list-style-type: none">• Access Path Name—Variable name to use in the source path.• Folder—Location of the source path.

Table 7-95 • Source Paths Tab of the InstallAnywhere Preferences Dialog Box

Option	Description
Add	<p>To add an entry to the Source Paths list, perform the following steps:</p>  <p>To add a new Source Path:</p> <ol style="list-style-type: none"> 1. Click Add. A blank box appears under Access Path Name. 2. Double-click in the blank box and enter a variable name for the source path, such as RESOURCE. 3. Click the blank box under Folder. The Choose Folder dialog box opens. 4. Browse to the source path (or click New to create a new folder) and then click Select. The full path of the selected directory is now listed in the Folder column. 5. Click OK. <p>To refer to this Source Path elsewhere in your InstallAnywhere project, use the following notation:</p> <p><code>\$<access_path_name>\$</code></p> <p>For example, if the Access Path Name in this Source Path is RESOURCE, refer to it elsewhere in the project as \$RESOURCE\$.</p>
Remove	Click to delete the selected Source Path.
Default Source Paths	Listing of all defined Source Paths. Select those Source Paths that you



Note • For more information about source paths, see [Team Development](#).

Updates

Use the **Updates** tab to enable FlexNet Connect to automatically check for updates available InstallAnywhere. For more information on configuring InstallAnywhere to automatically check for updates and how to install updates, see [Installing InstallAnywhere Updates](#).

Use the **Updates** tab to also manage your participation in Flexera Software's Customer Experience Improvement Program. To participate in the Customer Experience Improvement Program, select the **I want to help improve InstallAnywhere by sending usage data to Flexera Software** check box.

This check box is selected by default if you agreed to join the program when you opened InstallAnywhere. If you no longer wish to participate in this program, clear the check box.

Designer Colors

Use the **Designer Colors** tab to alter the color scheme used within the InstallAnywhere development environment.

To change the color scheme, click the name of the color scheme and click **OK**. You must close and re-open InstallAnywhere for this change to take effect.

LaunchAnywhere Properties Dialog Box

The **LaunchAnywhere Properties** dialog box is opened by clicking the **Edit Properties** button on the **General Settings** tab of the **Create LaunchAnywhere for Java Application** customizer. On this dialog box, you can edit the **Name**, **Value**, and **Comment** of any of the selected LaunchAnywhere launcher's properties.

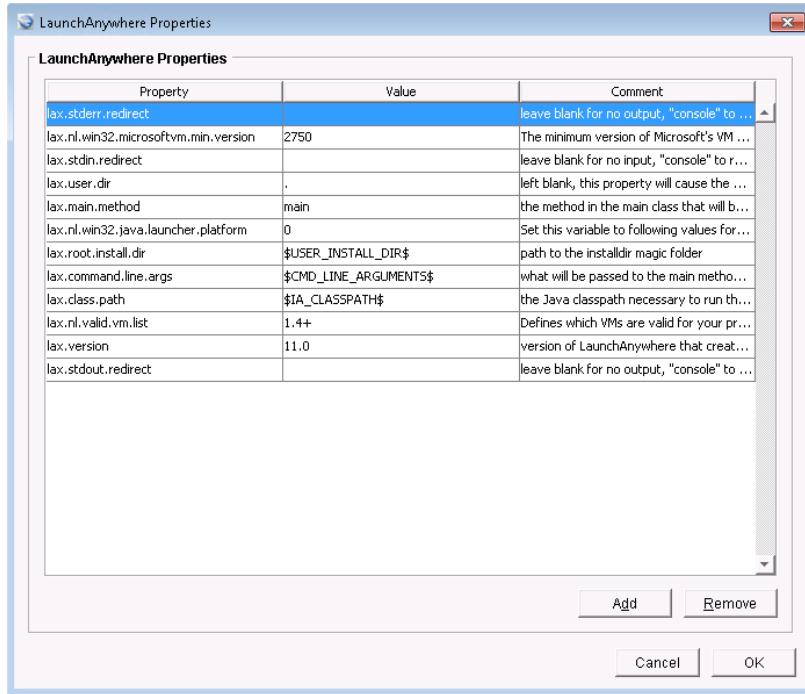


Figure 7-74: LaunchAnywhere Properties Dialog Box

Manage Instances Dialog Box



Edition • This feature is included with InstallAnywhere Enterprise Edition.

The **Manage Instances** dialog box opens when an end-user launches the installer for a product which has already been installed using a Instance Management-enabled installation. The user is prompted to choose to either **Install a New Instance** or **Modify an Existing Instance** by choosing an instance from the list.

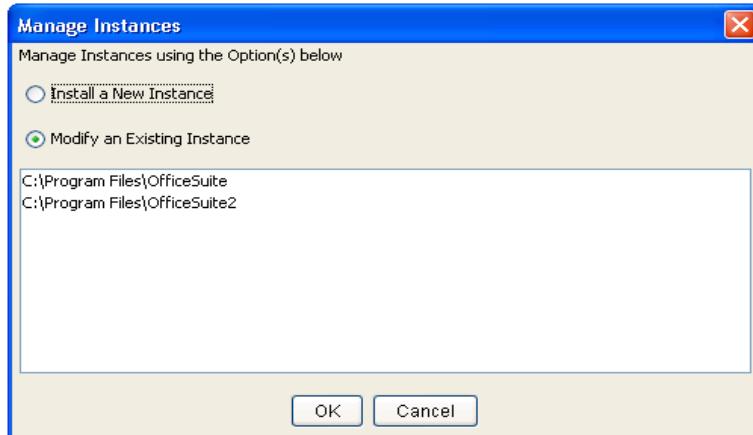


Figure 7-75: Manage Instances Dialog Box

The action that takes place depends upon what the user selects on this dialog box:

- **If the user selects Install a New Instance**, the installation continues and prompts the user to select a location for the new instance of the product.
- **If the user selects Modify an Existing Instance** and selects an instance from the list, Maintenance Mode is launched, and the user is prompted to choose whether to **Add Features**, **Remove Features**, **Repair Product**, or **Uninstall Product**.

New Project Dialog Box

The **New Project** dialog box is provided to name and create InstallAnywhere projects. This dialog box is the first dialog you see when you open InstallAnywhere. It is also the first panel of the Project Wizard.

You can toggle between the **New Project** dialog box and the [Open Project Dialog Box](#) by selecting the **Create New Project** or **Open Existing Project** option.



Note • You can set the InstallAnywhere default user interface mode by selecting **Project Wizard** or **Advanced Designer** under **InstallAnywhere Startup** on the **General Settings** tab of the [InstallAnywhere Preferences Dialog Box](#).



Note • In InstallAnywhere Enterprise Edition, a template must be selected for the new project. For a first project, use the Basic Project template.

The **New Project** dialog box includes the following options:

Table 7-96 • New Project Dialog Box

Option	Description
Create New Project	Select this option to show the New Project dialog box controls. This is the default selection.
Open Existing Project	Select this option to display the Open Project Dialog Box and show the controls for choosing an existing InstallAnywhere project.
Choose Template and Create Project	<p>Lists the available templates. Select the template that you want to use to create the new project.</p> <p>A template is simply a Merge Module that has been placed within the <code>iatemplates</code> folder inside the InstallAnywhere installation folder.</p> <p>Templates are generally used as starting points for installer projects. Installer items that remain unchanged such as the license agreement panel would be saved in an InstallAnywhere template. You may also want to create a template to maintain the look and feel for your installer projects.</p>
Other Template	Click to open the Choose Template dialog box, where you can select a custom template (<code>iam.zip</code>).
Project File Name	After you have clicked Save As to open the Save New Project As dialog box and entered a name for this new project, the project name is listed in this field.
Save As	Click to open the Save New Project As dialog box, where you are prompted to enter a name for the new project.
Exit	Click to exit InstallAnywhere.
Advanced Designer	Click to switch to the Advanced Designer interface.
About	Click to open the InstallAnywhere About dialog box.
Next	Click to save the new project using the settings you provided and proceed to the Project Info panel of the Project Wizard.

Open Project Dialog Box

The Open Project dialog box is available from the Project Wizard's New Project dialog box when you click Open Existing Project.

Table 7-97 • Dialog Box Controls

Control	Description
Create New Project	The default setting for the New Project dialog box, Create New Project shows the New Project dialog box controls. See New Project Dialog Box .
Open Existing Project	Changes the New Project dialog box to the Open Project dialog box and show the controls for choosing an existing InstallAnywhere project.
Project List	Lists recent InstallAnywhere projects
Other Project	Opens the Open Project File dialog box.
Exit	Dismisses the Open Project dialog box without creating a new project.
Next	Shows the first frame of the Project Wizard (Project Info).
Advanced Designer	Opens the project in the Advanced Designer interface.
About	Opens the About dialog box.

Open Project File Dialog Box

The **Open Project File** dialog box is available from the Advanced Designer as well as the Project Wizard:

- In the Advanced Designer, click **File > Open**.
- In the Project Wizard, navigate to the **Open Project** dialog box and click **Other Project**.

Table 7-98 • Dialog Box Controls

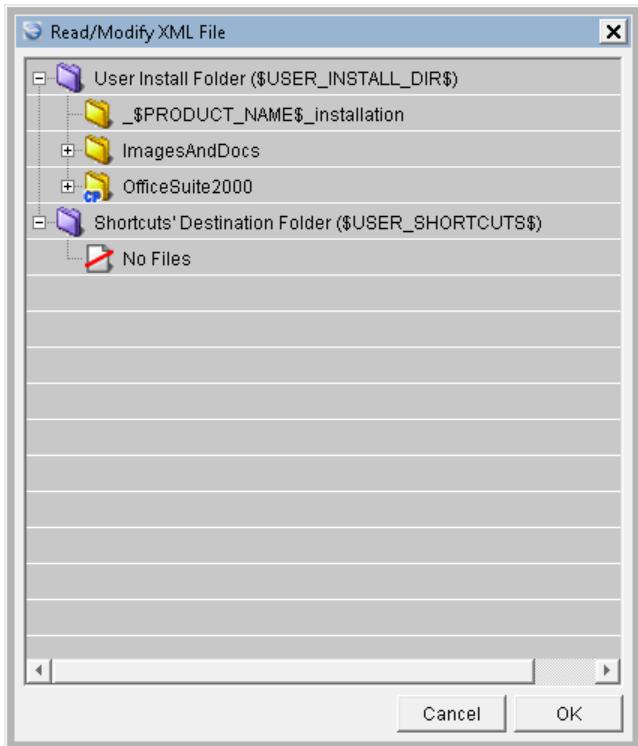
Control	Description
Folder	Shows the current working directory and provides a listing of the parent directory and its adjacent directories.
Path	Shows the full path to the current working directory. This control also lists paths to many previously used directories and allows you to type a path as well. (To move to a manually typed directory, either press the Enter key or click Go .)
Directory List	Lists the contents of the currently selected directory.

Table 7-98 • Dialog Box Controls

Control	Description
Files of Type	Filters the files that appear in the Directory List. Choose from <ul style="list-style-type: none"> • InstallAnywhere Project Files (*.iap_xml, *.iap) • Project Manifest (*.imf_xml)
New	Creates a new directory in the currently selected directory using a folder name you provide.
Cancel	Closes this dialog box without making any changes.
Open	Opens the selected project file. In the Advanced Designer, the project opens to the Project task. In the Project Wizard you return to the Open Project dialog when you click Open. The project you chose now appears, selected, in the Projects list.

Read/Modify XML File Dialog Box

On the **Read/Modify XML File** dialog box, which opens when you click **Choose Target** on a **Read/Modify XML File** action customizer, select the XML file that will be installed on the target system that you want to read or modify.

**Figure 7-76:** Read/Modify XML File Dialog Box

RPM Specification Settings Dialog Box

On supported Linux systems, RPM registration creates a virtual package and uses it to make entries to the RPM database.

When you select the **Enable RPM Registration** option on the **UNIX** platform of the **Project > Platforms** subtask of the Advanced Designer, the **Configure** button is then enabled. Click **Configure** to open the **RPM Specification Settings** dialog box, which includes the following controls:

Table 7-99 • RPM Specification Settings Dialog Box Controls

Control	Description
Name	The Name string from the RPM file name you plan to use.
Version	This version number is included in the package name.
Release	The release number that distinguishes two different packages with the same version number.
Description	A brief description of the functionality or application provided.
Summary	A one-line description of the package functionality.
Copyright	The copyright for the package. On target systems that are running RPM version 4 or later, the Copyright field is not used.
License	The license for the package to be installed (GPL, LGPL, commercial). On target systems that are running versions of RPM prior to 4, the License field is ignored.
URL	The URL you want the package to reference.
Distribution	The name of the distribution this package belongs to, if any.
Vendor	The name of the package vendor.
Group	Specify the group to which the package belongs. This value tells high-level installation programs (such as Red Hat's gnorpm) where to place this particular program in its hierarchical structure. (Find group descriptions in /usr/doc/rpm*/GROUPS.) Or specify groups based on your installed product family. Default: Applications/System.
Packager	The name of the team that created the package. This value is typically an email address or contact information for the installed product.

Save New Project As Dialog Box

The **Save New Project As** dialog box is available from the **Create New Project** dialog box and from the **New Project** dialog box.

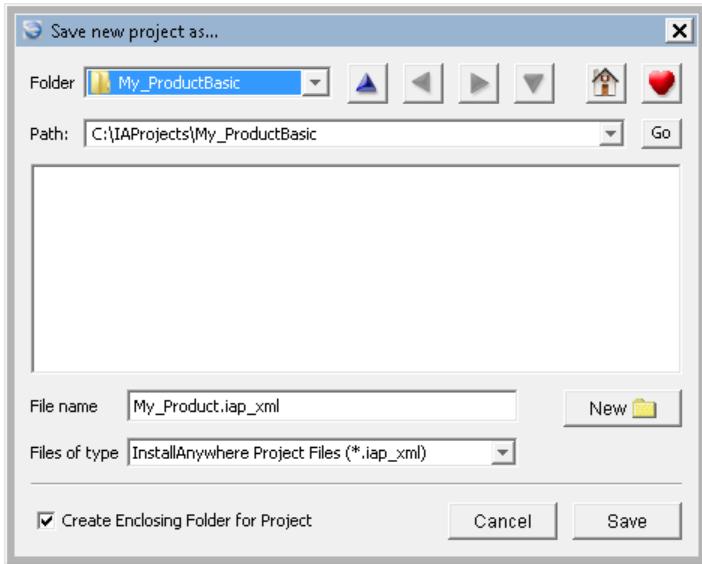


Figure 7-77: Save New Project As Dialog Box



Note • The **Create New Project** command is available from the Advanced Designer's **File** menu while the **New Project** dialog box is the first dialog box users see when they have InstallAnywhere configured to start in Project Wizard mode.

Table 7-100 • Save New Project As Dialog Box

Options	Description
Folder	Shows the current working directory and provides a listing of the parent directory and its adjacent directories.
	Use to jump to different directory in the Folder list. <ul style="list-style-type: none"> • Up arrow—Go up one directory level. • Left arrow—Go back to the previous directory. • Right arrow—Go forward one directory. • Down arrow—Open the selected directory.
	Click to open the user-specific Home directory which, in Windows is: C:\Documents and Settings\User_Name

Table 7-100 • Save New Project As Dialog Box

Options	Description
	Click to open the Favorites menu, where you can choose to jump to a previously identified favorite directory (by selecting it from the list) or to create a new favorite directory (by selecting Add Folder to Favorites).
Path	Shows the full path to the current working directory. This control also lists paths to many previously used directories and allows users to type a path as well. (To move to a manually typed directory, either press the Enter key or click Go .)
Directory List	Lists the contents of the currently selected directory.
File Name	Shows the currently selected project file name or, if no project is selected, the default project file name (My_Product.iap_xml). Type a new file name to specify a name for the new project.
Files of Type	Filters the files that appear in the Directory List. Choose from <ul style="list-style-type: none"> • Show All • InstallAnywhere Project Files (*.iap_xml)
New	Creates a new directory in the currently selected directory using a folder name you provide.
Create Enclosing Folder for Project	Determines whether or not InstallAnywhere creates a new folder for the new project file. If enabled, this control creates a folder named for the current file name. If disabled, InstallAnywhere saves the new project file in the current working directory.
Cancel	Closes this dialog box without making any changes.
Save	Stores the file name and location of the new project file. (New folders are created, but the project file is not created until the new project opens in the Advanced Designer or Project Wizard interface.)

Search Results Dialog Box



Edition • This feature is included with InstallAnywhere Enterprise Edition.

Using the **Search Results** dialog box, which is opened by selecting **Search** on the **File** menu or pressing Ctrl + F, you can search your entire installation project to locate all references to a specific variable or Build Configuration Tag. You can choose to search for an item in any of the installation tasks/phases (**Pre-Install**, **Install**, **Post-Install**, **Pre-Uninstall**, **Uninstall**, **Post-Uninstall**) and can choose to search **Features** and/or **Components**. You can also perform global replacements of variables.

Chapter 7: Reference

Dialog Boxes

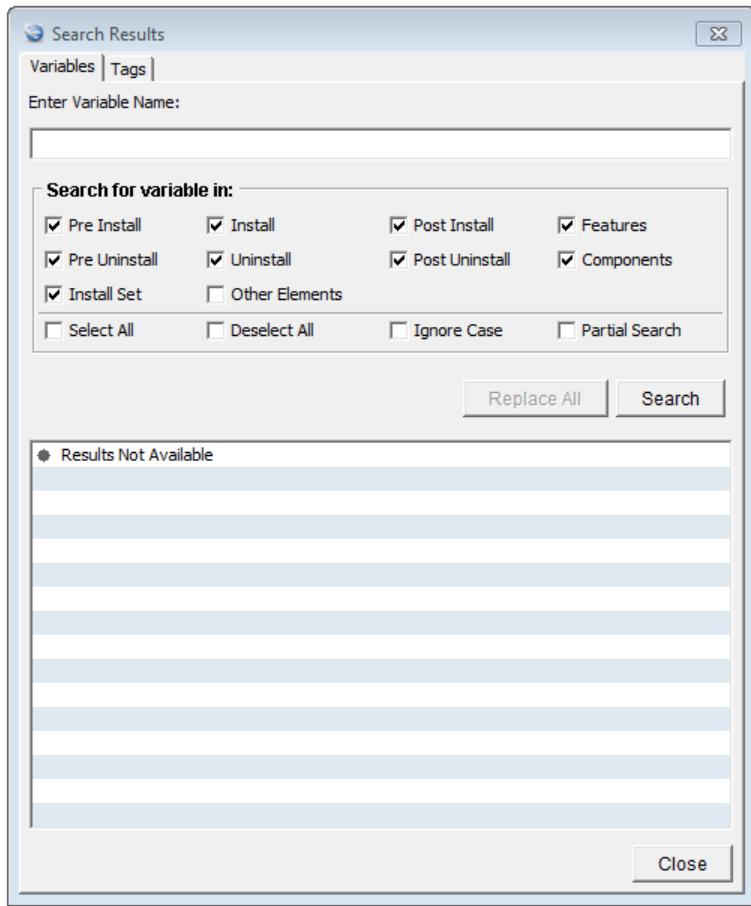


Figure 7-78: Search Results Dialog Box

Variables Tab

The **Variables** tab of the **Search Results** dialog box includes the following controls.

Table 7-101 • Variables Tab of Search Results Dialog Box

Control	Description
Enter Variable Name	<p>Enter the name of the variable to search for.</p> <p>If you want to perform a partial search (by selecting the Partial Search option), enter the text string in the variable that you want to search for.</p>  <p>Note • It is always recommended that you enter the \$ symbols before and after the variable name.</p>

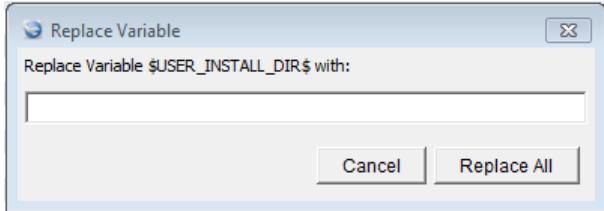
Table 7-101 • Variables Tab of Search Results Dialog Box

Control	Description
Search for variable in	Select the installation tasks/phases (Pre-Install , Install , Post-Install , Pre-Uninstall , Uninstall , and/or Post-Uninstall) and product elements (Features and/or Components) that you want to search.
Select All	Select to automatically select all of the options in the Search for variable in section.
Deselect All	Select to automatically clear the selection of all of the options in the Search for variable in section.
Ignore Case	Select to perform a case-insensitive search.
Partial Search	Select if you want to perform a search for a text string in a variable name.  Note • After performing a partial search, the Replace feature is not available.
Results List	Before a search is performed or if no search results were found, Results Not Available is displayed. After a search is performed, results are listed in a tree structure with the total number of variables found listed at the top, and items associated with the selected variable listed below, grouped by category: 

Chapter 7: Reference

Dialog Boxes

Table 7-101 • Variables Tab of Search Results Dialog Box

Control	Description
Replace All	After a search has been performed, and search results are listed, click Replace All to replace all instances of the found variable with a replacement value. The Replace Variable dialog box opens, prompting you to enter the replacement value.  
Note	<i>• It is always recommended that you enter the \$ symbols before and after the replacement variable name. For example instead of searching for \$NAME\$ and replacing it with NEWNAME, you should replace it with \$NEWNAME\$.</i>
Search	Click to initiate a search for the specified variable (or text string in variable) in the selected installation phases.

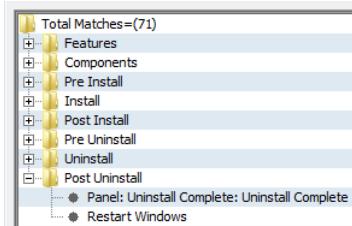
Tags Tab

The **Tags** tab of the **Search Results** dialog box includes the following controls.

Table 7-102 • Tags Tab of Search Results Dialog Box

Control	Description
Select Tag	Select the Build Configuration Tag that you want to search for from the list.
Search for Selected Tag in	Select the installation tasks/phases (Pre-Install , Install , Post-Install , Pre-Uninstall , Uninstall , and/or Post-Uninstall) and product elements (Features and/or Components) that you want to search.
Search	Click to initiate a search for the selected Tag.

Table 7-102 • Tags Tab of Search Results Dialog Box

Control	Description
Results List	<p>Before a search is performed or if no search results were found, Results Not Available is displayed.</p> <p>After a search is performed, results are listed in a tree structure with the total number of Tags found listed at the top, and the found Tags grouped by category:</p> 

SWVPD Registry Settings Dialog Box

On the **SWVPD Registry Settings** dialog box, which is opened by clicking the **Configure** button in the **SWVPD Registry Information** area of the **UNIX** platform of the **Project > Platforms** subtask, you can choose to include AIX registry (Software Vital Product Data) support in your project's installers. On AIX systems, this option ensures that products are properly added to and removed from the SWVPD registry.

The SWVPD Registry Settings dialog box includes the following controls:

Table 7-103 • SWVPD Registry Settings Dialog Box

Control	Description
Product Name	The name under which you want this product logged in SWVPD.
Product Version	The version under which you want this product logged in SWVPD.
Product Description	The description to be entered for this product in SWVPD.



Note • See the [UNIX](#) platform for more information.

Uninstaller Properties Dialog Box

The **Uninstaller Properties** dialog box is opened by clicking the **Edit Properties** button on the **General Settings** tab of the **Create Uninstaller** customizer. On this dialog box, you can edit the **Name**, **Value**, and **Comment** of any of the selected Uninstaller launcher's properties.

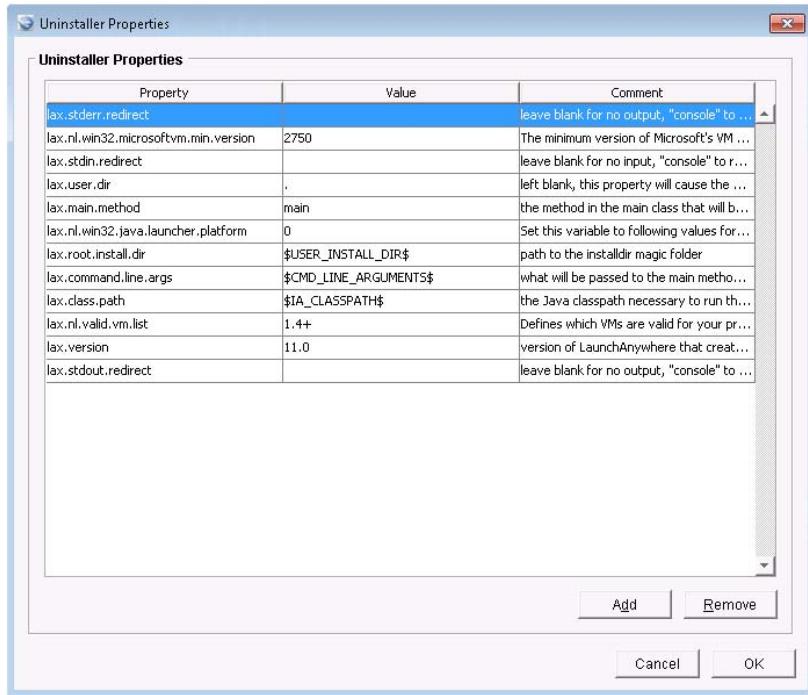


Figure 7-79: Uninstaller Properties Dialog Box

Menus

InstallAnywhere provides a small set of commands available from Menus. While File and Edit are fairly standard, some usage options are suggested in this section.

Table 7-104 • InstallAnywhere Menus

Menu	Description
File	Contains the standard open, save, and exit commands, plus commands to save a copy of the project or revert the current project to its last saved state.
Edit	Includes several commands for Actions and Rules clipboards as well as the Preferences command.
Wizard	Contains the Start Wizard command that launches the Project Wizard interface when you are working in the Advanced Designer.

Table 7-104 • InstallAnywhere Menus

Menu	Description
Help	Includes the About InstallAnywhere command plus commands to open the help library, check for updates, and register InstallAnywhere.

File

The **File** menu contains the standard **Open** and **Save** commands with a few more options.

New

Select **New** to create a new InstallAnywhere development project. To open and name a new project, click **Save As** within the **Create New Project** dialog box. On the **Create New Project** dialog box, you can also open templates by selecting **Open Other**.

Open

Select **Open** to open the **Open Project File** dialog box.

Open Recent

Select **Open Recent** to open a pop-up list of recently opened InstallAnywhere project files.

Save

Select **Save** to save the project. Since the project is named upon creation, this task will never request a name.

Save As

Select **Save As** to rename the project. The original project still exists, but remains in the state it was at its last save. The renamed project contains any changes and will become the currently open project.

Save a Copy As

Select **Save a Copy As** to save the project using a new name. However, unlike selecting **Save As**, selecting **Save a Copy As** will only make a copy, leaving the original project as the currently open project.

Revert to Saved

Select **Revert to Saved** to revert to the last saved state of the project. If you want to save a version of the project before reverting, select **Save a Copy As**, and then select **Revert to Saved**.

Search

Select **Search** to search for InstallAnywhere variables present in any location of a project. When you select **Search**, the **Search Results** dialog box opens, prompting you to enter the variable name that you are searching for and the project phases to search.

Chapter 7: Reference

Menus

For instructions on how to perform a variable search, see [Searching for InstallAnywhere Variables](#).

Create VM Pack

Select **Create VM Pack** to create a new VM pack. The **Create VM Pack** dialog box opens, prompting you specify information about the VM pack. For instructions on how to create a VM Pack, see [Creating VM Packs](#).

Exit

Select **Exit** to close the current project and exit the InstallAnywhere development environment. If any changes have been made since launching or the last save, you will be prompted to save those changes.

Edit

The **Edit** menu includes several commands for Actions and Rules clipboards. You can **Cut Rules**, then **Copy Action**, **Paste Action**, and **Paste Rules** with no interference between the Action and Rules operations. The Edit menu also includes a Preferences command that opens the InstallAnywhere Preferences dialog box.

Cut, Copy, Paste Actions

Cut Action removes an action from the task making a copy on the clipboard while Copy Action leaves the action and then makes the copy to the clipboard. Actions can then be pasted in other area of the current task, or may be pasted in other tasks by using the Paste Action command.

Clear Action

Clear Action removes the action in the same way as the Remove button.

Cut, Copy, Paste Rules

Cut Rules, Copy Rules, and Clear Rules allow developers to choose between the rule that is selected and all of the rules that are associated with an action. Developers choose between these selections by means of a pop-up menu that appears when they select the option. Cut Rules removes the rule or rules and makes a copy on the clipboard. Copy Rules leaves the rule or rules and makes a copy on the clipboard.

Paste Rules allows developers to add the rule to the existing rule or list of rules or replace the existing rule or rules.

Clear Rules

Clear Rules removes the selected rule or all of the rules association with an Action.

Preferences

Use the **InstallAnywhere Preferences** dialog box to customize the InstallAnywhere development environment. See [InstallAnywhere Preferences Dialog Box](#) for more information.

Wizard

The **Start Wizard** command in the Wizard menu opens the InstallAnywhere Project Wizard. The Project Wizard opens displaying the Project Info frame. The project you have open when you click Start Wizard remains open.

Help

The following commands are available on the **Help** menu:

About InstallAnywhere

The About InstallAnywhere command opens the InstallAnywhere splash screen, which contains version and copyright information and the product serial number.

- Click **Preferences** to open the [Preferences](#) dialog box.
- Click **OK** to close the splash screen.

Specify license information

The **Specify license information** command opens the **InstallAnywhere Licensing Wizard**. You can choose to continue to evaluate InstallAnywhere or to specify license information.



Note • Unregistered copies of InstallAnywhere create installers that will expire after three days.

InstallAnywhere Help

The InstallAnywhere Help command opens the InstallAnywhere online help in the default Internet browser.

Check for Updates

The **Check for Updates** command checks if there are any update notifications available for InstallAnywhere.

If aFileSync update is available, the **Update Available** dialog box appears. AFileSync update represents a critical update to the product.

- Click **Now** to immediately close InstallAnywhere and install the available update. A dialog appears displaying the progress of the update installation.
- Click **On Exit** to install the available update the next time you close InstallAnywhere.
- Click **Later** to dismiss the dialog box. The update is not installed, but this update is still available next time you check for updates.
- Select the **Check for updates automatically** check box if you want InstallAnywhere to check for updates automatically.

If a non-critical update or a message is available, a **FlexNet Connect** window appears. Select the available update in the list and click **Install**.

- See [Installing InstallAnywhere Updates](#) for more details on installing updates to InstallAnywhere.
- See [Updates](#) for details on setting InstallAnywhere update preferences.

Download Additional VM Packs

Select **Download Additional VM Packs** to open the **InstallAnywhere: Files & Utilities** page of the Flexera Software Web site. On the **VM Pack** tab of this page, you can download Java Virtual Machines that can be bundled into installers built using InstallAnywhere.

Actions

Actions represent operations for the installer to perform. InstallAnywhere supports an extensible action architecture that provides the ability to perform operations before, during, and after the installation. Actions can install files and folders, create shortcuts, execute custom code during the installation process, extract contents from a compressed file, and more.

Table 7-105 • InstallAnywhere Actions By Type

Action Type	Description
Install Actions	Available only in the Install task, these actions deploy the installer payload to the target machine.
Uninstall Actions	Available in the Uninstall task only, you can use these actions to customize the flow of the InstallAnywhere Uninstaller.
General Actions	Available in the Install task, as well as the pre- and post-install/uninstall tasks, these actions are typically transparent to the end user and require no end user interaction.
Panel Actions	Available in the Pre-Install, Post-Install, Pre-Uninstall, and Post-Uninstall tasks, these actions add panels to your installer for the purposes of communicating with the end user and acquiring end user input.
Console Actions	Similar to Panel actions, these actions communicate with end users and acquire end user input for installers that use a console interface.
System i (i5/OS) Actions	Consolidated in a single tab, these actions include general, install, panel, and console actions specifically for i5/OS installers.
Plug-In Actions	Developers can register custom code as plug-ins with the InstallAnywhere Advanced Designer. Plug-ins are stored within the <code><InstallAnywhere>/plugins</code> folder.



Tip • Action customizers share many properties and controls. See [Common Action Properties](#) for descriptions of the common properties for action customizers, panel action settings, and Get User Input panels.



Note • Plug-ins also appear in the Choose an Action dialog box when one or more InstallAnywhere plug-ins are installed on your system. For more information, see [About Plug-ins](#) and [Packaging Custom Code as a Plug-in](#).

Install Actions

Available only in the **Install** task, Install actions deploy the installer payload to the target machine. The following table lists the **Install** actions available in InstallAnywhere.

Table 7-106 • Install Actions

Action	Editions	Description
Create Alias, Link, Shortcut	E S	Create an alias (Mac OS X), symbolic link (Unix and Linux), or shortcut (Windows).
Create Alias, Link, Shortcut to DIM File	E	Create an alias (Mac OS X), symbolic link (Unix and Linux), or shortcut (Windows) to a file within a Developer Installation Manifest (DIM).
Create DIM Reference	E	Create a reference to a DIM.
Create Folder	E S	Create a new folder on the end user's system.
Create LaunchAnywhere for Java Application	E S	Create a launcher to start the installed Java application.
Create Uninstaller	E S	Create the uninstaller and several additional files needed by the uninstaller.
Deploy WAR/EAR Archive	E	Deploy a WAR or EAR archive to an application server.
Enable Update Notifications	E S	Include the FlexNet Connect Java agent in installers built from this project.
Expand Archive	E S	Expand a ZIP file (.zip, .jar, .war, .ear) or decode a Mac Binary file (.bin) on the end user's system.
Expand Archive (7-zip)		Expand a 7-ZIP archive file (*.7z or *.xz) on the end user's system.
Expand Archive (TAR)		Expand a TAR file (.tar) on the end user's system.
Install Archive	E S	Install a ZIP file (.zip, .jar) on the target system.

Table 7-106 • Install Actions (cont.)

Action	Editions	Description
Install File	E S	Install a file from the installer onto the end user's system.
Install from Manifest	E	Install all of the files and folders specified in the manifest file on the end user's system. See Manifest Files for more information.
Install HP-UX Depot	E	InstallAnywhere can install and uninstall HP-UX depot files through the Install HP-UX Depot action. You must specify the package name within the depot file, as it may contain multiple packages. In order to install multiple packages in the same depot, add one action for every package you want to install. (This method does not increase the size of the installer.)
Install Linux RPM	E	<p>InstallAnywhere can install and uninstall Linux RPMs through the Install Linux RPM action. These RPMs can either be bundled with the installer or pre-existing on the system.</p> <p>If the RPM is relocatable, and the Relocatable check box is set in the action customizer, the RPM will be installed to its location in the file tree.</p> <p>Additionally, the RPM can be set to ignore dependencies (similar to the --nodeps option for the command-line RPM tool) and to force the installation (--force).</p>
Install Merge Module	E	Install a Merge Module as if the Merge Module were run as a separate silent installer.
Install Solaris Package	E	InstallAnywhere can install and uninstall Solaris package files through the Install Solaris Package action. These packages can either be bundled with the installer or pre-existing on the system. You must enter the name of the package. Additionally, InstallAnywhere supports bundling admin and response files for the package with the installer. For more information on these files, consult the man pages for pkgadd(1), pkgask(1), and admin(4).
Install SpeedFolder	E S	<p>Dynamically pickup files at build-time from a folder. All files are installed in one fast operation.</p> <p>SpeedFolders are especially useful for large installations with many files or builds that occur automatically.</p>
Move File	E	Move or rename a file from one location to another location on the end user's system.
Move Folder	E	Move or rename a folder from one location to another location on the end user's system.

Table 7-106 • Install Actions (cont.)

Action	Editions	Description
Set System Environment Variable		Set environment variables on the end user's system. Compatible with Windows and Unix only. Unix Bash, sh, ksh, zsh, csh, and tcsh shells are supported.
Run SQL Script	E	Runs an SQL script on a database server.
Trigger Rollback		Add to the Install task to specifically initiate a rollback if a certain rule or condition is met.

Create Alias, Link, Shortcut

You can use the **Create Alias, Link, Shortcut** action to create an alias (Mac OS X), a symbolic link (Unix and Linux), or a shortcut (Windows).

The **Properties** tab of the **Create Alias, Link, Shortcut** action customizer includes the following properties:

Table 7-107 • Create Alias, Link, Shortcut Action Customizer

Property	Description
Destination	Specify the following information to identify the destination information: <ul style="list-style-type: none"> Path—Specify the location on the target machine where the alias, link, or shortcut will be created. First choose a platform-specific directory or location from the list, and then enter subdirectory information in the box. Name—Enter a name to identify the alias, link, or shortcut.
Original	Choose one of the following to specify whether the file is being installed, or already exists on the end user's system. <ul style="list-style-type: none"> Installed file—Select this option to create an alias, link, or shortcut to a file that will be installed during installation. Click Choose Target to open the Choose an Alias, Link, Shortcut, Target dialog box and select that file. Existing file—Select this option to create an alias, link, or shortcut to a file that already exists on the target system, and enter the path (fixed or relative) and file name for that file in the box.
Alias, Shortcut Icon	Click Change to choose a different icon for the shortcut. Click Same as Target to revert the icon to the InstallAnywhere default. When you click Change , the Choose Icon dialog box opens, prompting you to select an icon file (.ico), a 32x32 GIF file (.gif), and a 16x16 GIF (.gif) file.
Arguments	Enter any arguments that this shortcut should pass to the executable it represents as a target.

Table 7-107 • Create Alias, Link, Shortcut Action Customizer

Property	Description
Start In	Enter the directory to start in this command.

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Create Alias, Link, Shortcut to DIM File

You can use the **Create Alias, Link, Shortcut to DIM File** action to create an alias (Mac OS X), a symbolic link (Unix and Linux), or a shortcut (Windows) to a file within a Developer Installation Manifest (DIM).

Properties Tab

The **Properties** tab of the **Create Alias, Link, Shortcut to DIM File** action customizer includes the following properties:

Table 7-108 • Create Alias, Link, Shortcut to DIM File Action Customizer | Properties Tab

Property	Description
Destination	Specify the following information to identify the destination information: <ul style="list-style-type: none"> • Path—Specify the location on the target machine where the alias, link, or shortcut will be created. First choose a platform-specific directory or location from the list, and then enter subdirectory information in the box. • Name—Enter a name to identify the alias, link, or shortcut.
Original	Click Choose Target to open the Choose an Alias, Link, Shortcut, Target dialog box and select the DIM file that you want to create a shortcut to.
Alias, Shortcut Icon	Click Change to choose a different icon for the shortcut. Click Same as Target to revert the icon to the InstallAnywhere default.
Arguments	Enter any arguments that this shortcut should pass to the executable it represents as a target.
Start In	Enter the directory to start in this command.

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Create DIM Reference

You can use the **Create DIM Reference** action to create a reference to a Developer Installation Manifest (DIM) file.

DIM references can be reassigned to Features and Components like all other actions. By default, DIM references are associated with the first available feature and are added at the same level as Magic Folders. Use the **Assign to** list and the navigational arrows to reassign DIM references and change their order within the Visual Tree.



Note • Any changes you make in the DIM Reference customizer are reflected in both the **Install** task and in the **Organization > DIM References** subtask.

Properties Tab

The **Properties** tab of the **DIM Reference** action customizer has three subtabs:

- [General Tab](#)
- [Build Variables Tab](#)
- [Runtime Variables Tab](#)

General Tab

The **Properties > General** subtab of the **DIM Reference** action customizer includes the following properties

Table 7-109 • Create Alias, Link, Shortcut to DIM File Action Customizer | Properties > General Tab

Property	Description
Name	(Read only) Name of the selected DIM file.
Source Path	Click Choose DIM Reference to associate the DIM reference with a .dim file. The Add DIM Reference to Project dialog box opens. Browse to the .dim file and click Open . All of the General tab properties are updated to reflect the DIM that you selected. 
	Note • All of the General tab properties are read-only except for Destination Path . This property represents the location that the predefined variable \$[INSTALLDIR] is associated with according to the DIM.
Unique ID	(Read only) ID of the selected DIM file.
Destination Path	If you want to change the location associated with \$[INSTALLDIR], enter the desired location in the Destination Path box.
Meta Information	(Read only) Meta information of the selected DIM file.

Build Variables Tab

The **Properties > Build Variables** subtab of the **DIM Reference** action customizer lists the build variables associated with the selected DIM file.

Table 7-110 • Create Alias, Link, Shortcut to DIM File Action Customizer | Properties > Build Variables Tab

Property	Description
Name	(Read only) Name of associated variable.
Description	(Read only) Description of associated variable.
Value	Value of associated variable. To change the value associated with the variable, click the Value box and either enter the desired value or click the ellipses button to browse to a directory location.

Runtime Variables Tab

The **Properties > Runtime Variables** subtab of the **DIM Reference** action customizer lists the runtime variables associated with the selected DIM file.

Table 7-111 • Create Alias, Link, Shortcut to DIM File Action Customizer | Properties > Runtime Variables Tab

Property	Description
Name	(Read only) Name of associated variable.
Description	(Read only) Description of associated variable.
Value	Value of associated variable. To change the value associated with the variable, click the Value box and either enter the desired value or click the ellipses button to browse to a directory location.

Rules, Tags & Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Create Folder

You can use the **Create Folder** action to create a new folder on the end user's system. If the folder already exists, the existing folder will not be deleted.

Properties Tab

The **Properties** tab of the **Create Folder** action customizer includes the following properties:

Table 7-112 • Create Folder Action Customizer | Properties Tab

Property	Description
Destination	Specify the following information to identify the destination information: <ul style="list-style-type: none"> • Path—Specify the location on the target machine where the folder will be created. First choose a platform-specific directory or location from the list, and then enter subdirectory information in the box. • Name—Enter a name to identify the folder.

Table 7-112 • Create Folder Action Customizer | Properties Tab

Property	Description
Do not uninstall	Select this option to prevent the uninstaller from removing the folder from the target system.
In classpath	Select this option to put the folder on the classpath for all LaunchAnywhere executables installed.  Note • If a folder is on the classpath, Java will look in that folder for classes and things it is trying to find. However, a more standard behavior is to put everything in a JAR file and then to put the JAR file in the classpath.
Recursively delete all contents of this folder during uninstall	Select this option if you want the contents of this folder and all subfolders deleted during uninstallation.
Override default UNIX / Mac OS X permissions	Select this option to set the file permissions to a specific value for this folder. When you select this option, the Permissions text box is enabled. In the text box, enter the numeric permissions value (such as the octal number mode, 755).

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Create LaunchAnywhere for Java Application

You can use the **Create LaunchAnywhere for Java Application** action to create a launcher to start the installed Java application.

Properties Tab

The Properties tab of the **Create LaunchAnywhere for Java Application** customizer has two subtabs:

- [General Settings Tab](#)
- [Advanced Settings Tab](#)

General Settings Tab

The **General Settings** tab includes the following options:

Table 7-113 • Create LaunchAnywhere for Java Application | Properties > General Settings Tab

Option	Description
Path	Identifies the location of the LaunchAnywhere executable to be created on the target system.
Name	Sets the name for the launcher.
Main Class	Specifies the main class in your Java application. This is the class the launcher invokes. Click Automatically Find Main Classes to open the Choose a main class dialog box, which lists the available classes in the project, and select a class from the list.
Arguments	Specifies the list of arguments to pass to the Java application's main method. The default value is \$CMD_LINE_ARGUMENTS\$.
Launcher Type	Choose Graphical Launcher if your application has a graphical interface. Otherwise, choose Console Launcher .
Icon	Click Change to open the Choose Icon dialog box where you can choose a different icon for the launcher. Click Default to revert the launcher icon to the InstallAnywhere default.
Launcher Properties	Click Edit Properties to open the LaunchAnywhere Properties Dialog Box . This dialog box allows you to edit the properties specific to the launcher.
Windows Execution Level	Choose from As Invoker , Highest Available , or Administrator to set the privileges required by your application.
Override Default UNIX/Mac OS X Permission	Select this option to set the file permissions to a specific value for this launcher. When you select this option, the Permissions text box is enabled. In the text box, enter the numeric permissions value (such as the octal number mode, 755).

Advanced Settings Tab

The **Advanced Settings** tab controls the VM selection for this launcher and includes the following options:

Table 7-114 • Create LaunchAnywhere for Java Application | Properties > Advanced Settings Tab

Option	Description
Configure LaunchAnywhere with	<p>Select one of the following options:</p> <ul style="list-style-type: none">• VM Selected by the Installer or by the End User via Choose VM Panel• VM Used by the Installer• The First VM Found in the System Matching the VM Search Settings Defined Under Project >Java• No Specific VM (Allow LaunchAnywhere to Search for a VM Based on its lax.nl.valid.vm.list Property)  <p>Note • See Java Virtual Machines for a details about VM selection. See Creating Launchers for Java Applications for instructions on using this action.</p>

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Create Uninstaller

InstallAnywhere automatically adds this **Create Uninstaller** action to create an uninstaller. The **Create Uninstaller** action is basically a special LaunchAnywhere executable that creates the uninstaller and several additional files needed by the uninstaller. You can use the **Create Uninstaller** customizer to edit uninstaller settings.

Properties Tab

The **Properties** tab of the **Create Uninstaller** customizer has two subtabs:

- [General Settings Tab](#)
- [Advanced Settings Tab](#)

General Settings Tab

The **General Settings** tab includes the following options:

Table 7-115 • Create Uninstaller | Properties > General Settings Tab

Option	Description
Path	Identifies the location of the uninstaller executable to be created on the target system.  <i>Tip • Install the uninstaller into its own folder.</i>
Name	Sets the name for the uninstaller. By default, the name is: Change \$PRODUCT_NAME\$ Installation
Uninstaller Title	Enter the text that will be displayed on the title bar of the Uninstaller panels. By default, the title is: Change \$PRODUCT_NAME\$ Installation
Uninstaller Type	Choose Graphical Launcher if your application has a graphical interface. Otherwise, choose Console Launcher .
Icon	Click Change to choose a different icon for the uninstaller. Click Default to revert the icon to the InstallAnywhere default.
Uninstaller Properties	Click Edit Properties to open the Uninstaller Properties Dialog Box . This dialog box allows you to edit the properties specific to the uninstaller.
Windows Execution Level	Choose from As Invoker , Highest Available , or Administrator to set the privileges required by your application.

Table 7-115 • Create Uninstaller | Properties > General Settings Tab

Option	Description
Uninstall In-use files After Restart (Windows Only)	For Windows only, select this option to specify how the uninstaller manages changes to files that are locked (in-use) at uninstall time.
Override default UNIX/Mac OS X permissions	Establishes permissions specifically for this launcher. Enter the permissions value in the text box.

Advanced Settings Tab

The **Advanced Settings** tab controls the VM selection for this uninstaller and includes the following options:

Table 7-116 • Create Uninstaller | Properties > Advanced Settings Tab

Option	Description
Configure LaunchAnywhere with	<p>Select one of the following options:</p> <ul style="list-style-type: none"> VM Selected by the Installer or by the End User via Choose VM Panel VM Used by the Installer The First VM Found in the System Matching the VM Search Settings Defined Under Project->Java No Specific VM (Allow LaunchAnywhere to Search for a VM Based on its lax.nl.valid.vm.list Property)  <p>Note • See Java Virtual Machines for details about VM selection. See Creating Launchers for Java Applications for instructions on using this action.</p>

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Deploy WAR/EAR Archive

You can use the **Deploy WAR/EAR Archive** action to deploy a WAR or EAR archive to an application server.



Note • This action is available only when an **Application Server** host has been defined on the **Organization > Hosts** subtask and you have selected that Application Server in the Visual Tree.

Properties Tab

The **Properties** tab of the **Deploy WAR/EAR Archive** customizer has two subtabs:

- [General Settings](#)
- [Optional Settings](#)

General Settings

The **Properties > General Settings** tab of the **Deploy WAR/EAR Archive** customizer includes the following options:

Table 7-117 • Deploy WAR/EAR Archive Customizer | Properties > General Settings

Option	Description
Application Name	Enter a name to identify the archive file.
Source	Specify the EAR or WAR file you want to deploy by choosing one of the following options: <ul style="list-style-type: none">• Selected Archive—Select this option and click Choose Archive to select the archive you want to install.• Existing Archive—Select this option and enter a path and file name for the archive on the target system.
Undeploy During Uninstall	Select this option to ensure that the archive is undeployed when the uninstaller runs. This option is selected by default.

Optional Settings

The **Properties > Optional Settings** tab of the **Deploy WAR/EAR Archive** customizer includes the following options.

Table 7-118 • Deploy WAR/EAR Archive Customizer | Properties > Optional Settings

Option	Description
Deployment Plan	<p>For servers that support the JSR-88 standard (Geronimo, WebSphere, WebLogic, and Sun Application Server), you can choose an archive for the Deployment Plan. A deployment plan is an XML file that contains server-specific information not included in the EAR or WAR file's deployment descriptor.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> • Selected Archive—To choose an archive to bundle with the installer, click Selected Archive; then click Choose Archive and locate the deployment plan file. • Existing Archive—To choose an archive that exists on the target system, click Existing Archive and type a path and file name for the deployment plan.

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Enable Update Notifications

You can use the **Enable Update Notifications** action to include the FlexNet Connect Java agent in installers built from this project. This action deploys the FlexNet Connect Java agent along with your product. The Java agent periodically checks for notifications about your product and allows your users to view messages as well as to download and install any updates it finds. By default, the Enable Update Notifications action also creates a launcher for the Java agent so your end users have the option of checking for updates manually.



Note • The use of the **Enable Update Notifications** action requires a FlexNet Connect account in either a Flexera Software-hosted or self-hosted environment. You must create a product on the Publisher site with a product code that matches the **Product ID** in your installer (**Project > Description** task).

Properties Tab

The **Properties** tab of the **Enable Update Notifications** customizer has two subtabs:

- General Settings
- Advanced Settings

General Settings

The **Properties > General Settings** tab of the **Enable Update Notifications** customizer includes the following options:

Table 7-119 • Enable Update Notifications Customizer | Properties > General Settings

Option	Description
Have you registered your product with FlexNet Connect?	<p>Opens the login page for the Flexera Software-hosted Publisher site. You must register your product with FlexNet Connect to take advantage of the Enable Update Notifications action.</p> <p>If you have not yet registered your product, click Click Here and register your product at the Publisher site. If you use a self-hosted version of FlexNet Connect, you can manually open your Publisher site to make sure your product is registered.</p>
Destination for FlexNet Connect files	<p>Defines the install location for FlexNet Connect files. This shows where your installer will place the us.jar file (source: <InstallAnywhere>\isus\us.jar).</p> <p>Click the drop-down list to specify the location you want the FlexNet Connect files to be installed.</p>  <p>Note • The location of the Enable Update Notifications action in the Visual Tree changes to reflect the new destination.</p>  <p>Note • The text box associated with the FlexNet Connect file destination is read-only; however, you can place the Enable Update Notifications action in any folder in the Visual Tree and that change is automatically reflected in the customizer.</p>
Create LaunchAnywhere to execute update check	Select this option to enable or disable the creation of a LaunchAnywhere executable. By default, this option is selected. If you do not want to include a LaunchAnywhere executable for this action, clear this check box. Clearing this check box choice disables all subsequent settings on the General Settings tab.
Launcher Name	Enter a name for the LaunchAnywhere executable that initiates a check for notifications.

Table 7-119 • Enable Update Notifications Customizer | Properties > General Settings

Option	Description
Windows Execution Level	If your project builds installers for Windows, select one of the following options to identify the execution level you want to request for your Windows launchers. Choose from <ul style="list-style-type: none"> • As Invoker—The launcher acquires the same execution level as its parent process. • Highest Available—The launcher requests the highest execution level (Windows privileges and user rights) available to the current user. • Administrator—The launcher requires local admin privileges to run. Depending on the privileges of the current user account and the configuration of the target system, this setting may result in a launcher that will not start.
Create Alias, Link, Shortcut	Click this button to generate a link, alias, or shortcut for the Update Notifications launcher.
Launcher Icon	Shows the icon associated with the Enable Update Notifications launcher. Click Change to select a custom icon for this launcher. To revert to the default LaunchAnywhere icon, click Default .

Advanced Settings

The **Properties > Advanced Settings** tab of the **Enable Update Notifications** customizer includes the following controls:

Table 7-120 • Enable Update Notifications Customizer | Properties > Advanced Settings

Control	Description
Language of FlexNet Connect agent	Defines the language setting options for the FlexNet Connect agent you deploy. Click Use installer's runtime language to use the language in which the installer is run. Click Custom to pre-define a single language for the FlexNet Connect agent; then choose the language from the drop-down list.
Version of product used by FlexNet Connect	Define the version FlexNet Connect uses to associate your product with updates and messages. Click Use product's default to make the FlexNet Connect version setting match the version you specify in the Project > Description task. Click Custom to activate the Version text box and then type the version you want FlexNet Connect to use.
Update check interval (days)	Specify the number of days since the last scheduled check for notifications FlexNet Connect waits before issuing another check.

Table 7-120 • Enable Update Notifications Customizer | Properties > Advanced Settings

Control	Description
FlexNet Connect Server Settings	Define the server on which you have a FlexNet Connect account. Choose Use Flexera Software hosted FlexNet Connect server if you use FlexNet Connect via a Flexera Software-hosted account. Choose Self-hosted URL to activate the URL text box and then type the URL to your FlexNet Connect server.
FlexNet Connect libraries	Define the FlexNet Connect libraries you want to deploy with your installer. Click Use FlexNet Connect library supplied by InstallAnywhere to install the agent and software manager included with InstallAnywhere. These FlexNet Connect libraries are contained in the us.jar file, which is installed along with InstallAnywhere at <InstallAnywhere>\isus\us.jar. Click Custom location to activate the Location text box. To specify different libraries, click Choose and navigate to the JAR file you want to use and click Select . This adds the path to the custom FlexNet Connect libraries you chose. (You can also simply type the path and filename to the custom JAR file in the Location text box.)

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).



Note • For more information, see [Deploying Your Product with FlexNet Connect](#).

Expand Archive

You can use the **Expand Archive** action to expand a ZIP file (.zip, .jar, .war, .ear) or decode a Mac Binary file (.bin) on the end user's system.

Properties Tab

The **Properties** tab of the **Expand Archive** action customizer includes the following properties:

Table 7-121 • Expand Archive Action Customizer

Property	Description
Source	<p>Specify the archive file you want to expand by choosing one of the following options:</p> <ul style="list-style-type: none"> • Selected Archive—Select this option and click Choose Archive to select the archive you want to expand. • Existing Archive—Select this option and enter a path and file name for the archive on the target system. <p>The supported archive file formats are:</p> <ul style="list-style-type: none"> • ZIP files (.zip) • Java archives (.jar) • WAR files (.war) • Enterprise archives (.ear)
Destination	<p>Specify Path information to identify the destination information for the archive, the location on the target machine where the archive will be expanded.</p> <p>Choose a platform-specific directory or location from the list, and then enter subdirectory information in the box.</p>

Expand Archive Actions and Extracted File Permissions

InstallAnywhere includes three expand archive actions: **Expand Archive**, **Expand Archive (7-zip)**, and **Expand Archive (TAR)**. The permissions assigned to files that are extracted from an archive file during installation vary between the three types of expand archive actions. In some cases, the original permissions of the file are retained. In others, the settings made in the **Default Permissions** area of the **Project > Platforms > UNIX** or **Mac OS X** subtask are applied to the extracted files. The following table lists how permissions are applied for these three expand archive actions.

Table 7-122 • Expand Archive Actions and Extracted File Permissions Comparison

Action	Original File Permissions Preserved?	Apply Permissions Set on Project > Platforms Subtask?
Expand Archive	Not guaranteed to preserve	Yes
Expand Archive (7-zip)	No	Not guaranteed to apply

Table 7-122 • Expand Archive Actions and Extracted File Permissions Comparison

Action	Original File Permissions Preserved?	Apply Permissions Set on Project > Platforms Subtask?
Expand Archive (TAR)	Yes	No

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Expand Archive (7-zip)

You can use the **Expand Archive (7-zip)** action to expand a 7-zip or LZMA compressed archive file.

The **Expand Archive (7-zip)** action is available in the **Install** phase only.

Properties Tab

The **Properties** tab of the **Expand Archive (7-zip)** action customizer includes the following properties:

Table 7-123 • Expand Archive (7-zip) Action Customizer

Property	Description
Source	<p>Specify the 7-zip archive file you want to expand by choosing one of the following options:</p> <ul style="list-style-type: none"> • Selected Archive—Select this option and click Choose Archive to select the 7-zip archive you want to expand. • Existing Archive—Select this option and enter a path and file name for the 7-zip archive on the target system. <p>The supported archive file format is 7-zip archive (*.7z or *.xz) with LZMA compression.</p>
Destination	<p>Specify Path information to identify the destination information for the 7-zip archive, the location on the target machine where the 7-zip archive will be expanded.</p> <p>Choose a platform-specific directory or location from the list, and then enter subdirectory information in the box.</p>



Note • For information on how permissions are applied to files extracted from a *Expand Archive* (7-zip) action during installation, see [Expand Archive Actions and Extracted File Permissions](#).

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Expand Archive (TAR)

You can use the **Expand Archive (TAR)** action to expand a TAR archive file on the target system. This action has the advantage of preserving the permissions of all files that are part of the TAR archive.

Properties Tab

The **Properties** tab of the **Expand Archive (TAR)** action customizer includes the following properties:

Table 7-124 • Expand Archive (TAR) Action Customizer

Property	Description
Source	<p>Specify the TAR archive file you want to expand by choosing one of the following options:</p> <ul style="list-style-type: none"> • Selected Archive—Select this option and click Choose Archive to select the TAR archive you want to expand. • Existing Archive—Select this option and enter a path and file name for the TAR archive on the target system. <p>The supported archive file format is a TAR file (.tar).</p>
Destination	<p>Specify Path information to identify the destination information for the TAR archive, the location on the target machine where the TAR archive will be expanded.</p> <p>Choose a platform-specific directory or location from the list, and then enter subdirectory information in the box.</p>



Note • For information on how permissions are applied to files extracted from a *Expand Archive (TAR)* action during installation, see [Expand Archive Actions and Extracted File Permissions](#).

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Install Archive

Install a ZIP file (.zip, .jar) on the target system. This action is not available from the **Choose an Action** dialog box, but its customizer appears automatically when a ZIP file is added.

Install File

Install a file from the installer onto the end user's system. The **Install File** customizer includes the following options:

Table 7-125 • Install File Customizer

Option	Description
Source Path	Location of this file on the current system or in relationship to the InstallAnywhere project.
Destination Path	Specify the directory on the target machine where the file will be installed.
Destination Name	Specify the name that you want the selected file to be named on the target system.
Do not uninstall	Select this option to prevent the uninstaller from removing the file from the target system.
Override default UNIX / Mac OS X permissions	Select this option to provide pre-determined permissions for a launcher. Enter the numeric permissions value in the text box. For example, for the octal number mode, you could enter 755 .

Table 7-125 • Install File Customizer

Option	Description
If the file already exists on the end user's system	<p>Select one of the following options to define installer behavior when it attempts to install a file that already exists on the target system:</p> <ul style="list-style-type: none"> • Use project default • Always overwrite • Never overwrite • Overwrite if older, do not install if newer • Overwrite if older, prompt if newer • Prompt if older, do not install if newer • Always prompt user  <p>Note • If you select Always prompt user and you also choose to generate a response file, the overwrite choice that the user makes is recorded in the response file using the <code>-fileOverwrite</code> command. See Recording User Response to File Overwrite Prompts in Response File for more information.</p>



Note • This action is not available from the **Choose an Action** dialog box, but its customizer appears automatically when files are added.

Install Merge Module

Use this action to install a Merge Module as if the Merge Module were run as a separate silent installer. Specify the following options in the **Install Merge Module** customizer:

Table 7-126 • Install Merge Module Customizer

Option	Description
Bundle Merge Module at Build Time	Select this option if the Merge Module will be available when building the installer. Click Choose Merge Module and select the Merge Module. These Merge Modules will be included in the actual generated installer.
Locate Merge Module at Install Time	Select this option to have the installer install a Merge Module that is available at install time but is external to the installer. In the Location box, enter the location of the install time Merge Module. The Merge Module can be located either on the end user's system or stored on a CD. If you set the Location to a folder that contains several Merge Modules, all will be installed.
Execute Pre-Install Actions	Select this option to run the Merge Module's Pre-Install actions.

Table 7-126 • Install Merge Module Customizer

Option	Description
Execute Post-Install Actions	Select this option to run the Merge Module's Post-Install actions.
Inherit Parent Install Folder	Select this option to set the Merge Module to use the same value for the install folder that the parent installer is using.
Show Progress Dialog	<p>Select this option to display a separate progress bar, on a pop-up dialog box, when the Merge Module is being installed.</p> <p>If this option is not selected, then the Merge Module installation progress is updated on the main installation project's progress bar.</p>  <p>Note • See About Merge Module Progress for more information.</p>
Edit Variables	Click to open the Edit Advertised Variables Dialog Box where you can specify which variables in the Merge Module can be set by the parent installer and which variables in the parent installer can be set by the Merge Module.
Uninstall Merge Module when parent is uninstalled	<p>Select this option to uninstall this merge module when the main project gets uninstalled.</p>  <p>Tip • The point at which merge modules are uninstalled can also be configured using the Uninstall Merge Modules action in the Uninstall task.</p>  <p>Note • This option is selected by default for new InstallAnywhere 2011 projects. For projects made with previous releases of InstallAnywhere and opened in InstallAnywhere 2011, this option will not be selected.</p>
Add Merge Module log to parent log	<p>Select this option to append the Merge Module installation log to the main project log. By default, this option will not be selected.</p>  <p>Note • For this option to work properly, it is mandatory that logging is enabled for both the parent and the merge module. The same applies for appending the <code>stderr</code> and <code>stdout</code> entries to the log.</p>  <p>Note • You can only append merge module logs to the parent log if Plain text format is selected under Log Format on the Project > Log Settings task. XML log format is not supported.</p>

Install SpeedFolder

You can use an **Install SpeedFolder** action to dynamically pick up files at build-time from a folder. All files are installed in one fast operation. SpeedFolders are especially useful for large installations with many files or builds that occur automatically.

Specify the following options in the **Install SpeedFolder** customizer:

Table 7-127 • Install SpeedFolder Customizer

Option	Description
Source Path	Click Choose Folder and specify the source location of this SpeedFolder.
Destination Path	Specify the location on the target machine where the SpeedFolder will be created: <ul style="list-style-type: none"> Choose a platform-specific directory or location from the list. Enter subdirectory information in the box.
Options	Select one of the following options to specify whether to install the folder and its contents or only the folder's contents. <ul style="list-style-type: none"> Install the folder and the folder's contents Install only the folder's contents
If file already exists on end user's system	Use this option to define the installer behavior when it attempts to install a file that already exists on the target system. Select one of the following options: <ul style="list-style-type: none"> Use project default Always overwrite Never overwrite Overwrite if older, do not install if newer Overwrite if older, prompt if newer Prompt if older, do not install if newer Always prompt user
Filter Files	Filters are conditions that the installer uses to select files for modification. Click Configure Filter to open the Filter File dialog box. See Filter File Dialog Box for more information.
In classpath	Puts the item on the classpath for all LaunchAnywhere executables installed.
Do not uninstall	Tells an action to not attempt to undo the results of the action at uninstall time.
Recursively uninstall	Select this option if you want the contents of this folder and all subfolders deleted during uninstallation.

Table 7-127 • Install SpeedFolder Customizer

Option	Description
Override default UNIX/Mac OS X permissions	Sets the file permissions to a specific value for this action.

Run SQL Script

Runs an SQL script on a database server.



Note • This action requires an existing **Database Server** host on the **Organization > Hosts** subtask.

The **Run SQL Script** action allows you to specify separate SQL scripts (**Install Script** and **Uninstall Script**) to run during installation and uninstallation. You can specify an Install Script, an Uninstall Script, or both.

In either case, choose from:

- **Selected Script**—to specify a script file to bundle with your installer. Click **Choose File** to open the Choose a File dialog box. Then, browse to the script file you want your installer to run and click **Open**.
- **Existing Script**—to specify a script file that exists on the target system. In the **Existing Script** text box, type the path to the script file you want your installer to run.

SQL Statement Delimiter

Identifies the string that separates SQL statements. To change the default delimiter (;) type the new character or characters (for example, \n or GO). The Run SQL Script action parses the SQL script and then sends each statement to the database driver for execution.



Note • For both **Install Scripts** and **Uninstall Scripts** that are bundled with your installer (**Selected Script**), you can view the currently selected script by clicking the **View Script** button. (The SQL script cannot be altered in the **SQL Script Preview** window.)

Trigger Rollback

You can add a **Trigger Rollback** action to the **Install** task to specifically initiate a rollback if a certain rule or condition is met. For example, suppose that you are trying to install an application that interacts with a database. You could add a Trigger Rollback action to the installation so that if the installer detects that the database is not accessible from the machine on which the installation is in progress, the installation could be rolled back.

To add a Trigger Rollback action to the Install task, perform the following steps:

**Task:** *To add a Trigger Rollback action to the Install task:*

1. Open a project in the Advanced Designer.
2. Open the **Install** task.
3. Click **Add Action**. The **Choose an action** dialog box opens.
4. Select **Trigger Rollback Action** and click **Add**.
5. Click **Close** to close the **Choose an action** dialog box.
6. Select the **Trigger Rollback Action** in the **Visual Tree**.
7. In the **Trigger Rollback Action** customizer, select the **Rules** subtab.
8. Click **Add Rule**.
9. Add the rule or rules that you want to use to define the conditions for rollback. See [Rules Reference](#) for a description of all available rules.

If the conditions defined in the rules are not met, the installation will be rolled back.

Uninstall Actions

The following table lists the Uninstall actions available in InstallAnywhere. You can use them to customize the flow of the InstallAnywhere Uninstaller. These actions are only available in the **Uninstall** task.

Table 7-128 • Uninstall Actions

Action	Editions	Description
Uninstall Category	E	Groups sets of uninstall actions.
Uninstall AIX Entries	E	Uninstalls all entries made in the SWVPD registry of the AIX machine.
Uninstall DB Scripts	E	Runs the Uninstall scripts mentioned in the Run SQL action against the databases.
Uninstall Files	E	Uninstalls all of the files installed or created by the installer during installation.
Uninstall Folders	E	Uninstalls all of the folders installed or created by the installer during installation.
Uninstall JEE Archives	E	Undeploys WAR/EAR files deployed during installation.
Uninstall LaunchAnywheres	E	Uninstalls all of the LaunchAnywheres installed or created by the installer during installation.

Table 7-128 • Uninstall Actions (cont.)

Action	Editions	Description
Uninstall Merge Modules	E	<p>Uninstalls all associated merge modules installed by a parent installer during installation.</p> <p>For more information, see Uninstaller for Merge Modules and Multiple Products.</p>
Uninstall RAIR Entries	E	Uninstalls the RAIR component entries in i5/OS machines.
Uninstall Registry Entries	E	Uninstalls all of the registry entries installed or created by the installer during installation.
Uninstall RPM Packages	E	Uninstalls all RPM entries made by the installer.
Uninstall Shortcuts/Links/Aliases	E	Uninstalls all of the shortcuts, links, and aliases installed or created by the installer during installation.

General Actions

The following table lists the general actions available in InstallAnywhere.

Table 7-129 • General Actions

Action	Editions	Description
Action Group	E	Adds a folder to the Action List in which you can group sets of InstallAnywhere actions. (Available in the Install task as well as all pre- and post-install and uninstall tasks.)
Add Jump Label	E	<p>Use this action to branch off the installation conditionally. By applying InstallAnywhere rules, developers may jump the end user to a later or earlier part of the installation, depending on the specifics of their system or install.</p> <p>Use this action in conjunction with the Jump to Target action. (Available during the Pre-Install and Post-Install tasks.)</p>
Comment	E S	The Comment action is designed to allow developers to add a simple comment to the installer.
Copy File	E	Copy a file from one location to another location on the end user's system.

Table 7-129 • General Actions (cont.)

Action	Editions	Description
Copy Folder	E	Copy a folder from one location to another location on the end user's system.
Delete File	E	<p>Delete a file from the end user's system.</p> <ul style="list-style-type: none"> • Installed file—Select this option to delete a file to be deployed by your installer. Then click Choose Target to choose the file you want to delete. • Existing file—Select this option to delete a file from the target system, and enter the path (fixed or relative) and file name for the file you want to delete.
Delete Folder	E	Delete a folder from the end user's system.
Evaluate Dependencies	E	<p>Evaluate dependencies on which the installer is based. This action sets the following InstallAnywhere variables:</p> <p>\$DEPENDENCY_SUCCESSES\$ \$DEPENDENCY_FAILURES\$ \$DEPENDENCY_REPORT\$ \$DEPENDENCY_STATUS\$</p>
Execute Ant Script	E	Execute Ant Script allows developers to execute scripts designed for the Apache Jakarta Project's Ant application. If this action is selected, InstallAnywhere bundles Ant with the application.
Execute Command	E	Execute Command allows developers to run executables as they would from the target system's command line.
Execute Custom Code	E S	This action allows developers to extend the functionality of InstallAnywhere. The InstallAnywhere API is purely Java based and allows developers to do nearly anything that is possible in Java. The Execute Custom Code action represents the non-interactive interface for this API.
Execute Script/Batch File	E	Execute Script/Batch File allows developers to enter the text of a script or batch file which the installer executes on the target system. When this action runs, it first resolves any InstallAnywhere variables in the script. It then saves the script to the InstallAnywhere temp directory and executes the script. The Execute Script/Batch File action finishes by cleaning up and deleting the script from the temp directory.

Table 7-129 • General Actions (cont.)

Action	Editions	Description
Execute Target File	E S	<p>Launches any executable or opens a document that is included in the installer. If the target is a document that has the appropriate application associations set up, then the document is opened in the correct application.</p>  <p>Note • This action is available only during the installation of files and after files have been installed.</p>  <p>Note • To avoid the substitution of unknown variables for this action by instructing InstallAnywhere to only resolve InstallAnywhere variables which are listed in the project under Project > Variables (the known variables), select the Do not substitute unknown variables option. For more information, see Preventing the Substitution of Unknown Variables.</p>
Execute Uninstaller	E	<p>Runs the specified uninstaller with the settings provided for stdout, stderr, and exit code logging. This action also allows developers to define the GUI mode for the uninstaller.</p>
Find Component in Registry	E	<p>Determines if a component exists on a system through the cross-platform registry, as well as discover existing component versions, their location, and if there are multiple instances of a particular component on the destination system.</p>
Get Windows Registry Entry	E	<p>InstallAnywhere allows developers access to information stored in the Windows Registry through this action. This action retrieves the value or checks the existence of a key/value and then stores that information in InstallAnywhere variables to be used in the installation.</p>  <p>Tip • If you are targeting 64-bit systems, click the Access Specific Registries View (64-bit systems) check box. Then select whether you want this action to reference the 64-bit or 32-bit portion of the registry.</p>
Jump to Target	E	<p>Related to the Add Jump Label action, this action allows developers to jump over or back to a specific point in an installation. When controlled by InstallAnywhere rules, this action gives developers a conditional method of moving non-linearly through an install. (Available during the Pre-Install and Post-Install tasks.)</p>

Table 7-129 • General Actions (cont.)

Action	Editions	Description
Launch Default Browser	E S	<p>This action launches the user's default web browser with the arguments developers specify. It can open a URL or a file on the system.</p>  <p>Note • This action is available during the Pre-Install, Post-Install, Pre-Uninstall, and Post-Uninstall tasks.</p>
Modify Text File - In Archive	E	This action alters text files within an archive (ZIP or JAR).
Modify Text File - Multiple Files	E	This action alters several text files on the target system. Use this action any time you need to make the same changes to multiple text files within the same directory.
Modify Text File - Single File	E	This action modifies a text file on the target system.

Table 7-129 • General Actions (cont.)

Action	Editions	Description
Output Debug Information	E S	<p>InstallAnywhere has comprehensive debugging built into the installer. This action sends developer-specified output to either the console or a file. Developers can output the entire contents of the InstallAnywhere variable manager, the install tree, Java properties, and other information related to the installation.</p> <p>Output data options include:</p> <ul style="list-style-type: none"> • InstallAnywhere Variables—The values of all InstallAnywhere variables. • Magic Folders—The values for all Magic Folder variables. • Java Properties—The values for Java properties on the target system. • Visual Tree—A hierarchical representation of the project's visual tree. • Component Tree—A hierarchical representation of the project's component tree. • Pre-install Actions—A list of actions executed in the Pre-Install phase. • Post-install Actions—A list of actions executed in the Post-Install phase. <p>Output destination options include</p> <ul style="list-style-type: none"> • Output to the console (stderr)—Check to send selected output data to the console of the target machine. • Output to a file—Check to send selected output data to the file name referenced in the Path text box. <p>For more information, see Using the Output Debug Information Action.</p>
Output Text to Console	E S	<p>This action outputs the text specified to the debug console. This is useful for measuring the progress of a non-interactive installer in silent or console mode, or the progress of a non-interactive portion of the installation.</p>  <p>Note • This action is available in the Pre-Install, Install, and Post-Install tasks.</p>
Perform XSL Transform	E	<p>This action allows developers to specify an XSLT and target for an Extensible Stylesheet Language transform. Predefined XSL Transforms can be found in</p> <pre><InstallAnywhere>\resource\extras\presets</pre>

Table 7-129 • General Actions (cont.)

Action	Editions	Description
Perform XSL Transform - In Archive	E	This action works the same as the Perform XSL Transform, but does so for files in an archive—quite useful for configuring web applications in WAR, EAR, and JAR files.
Query InstallShield Universal Software Information	E	Queries InstallShield Universal Software registries for the UUID specified and returns the following information by default: \$ISMP_COMPONENT_COUNT\$ \$ISMP_COMPONENT_VERSIONS\$ \$ISMP_COMPONENT_LOCATIONS\$
Read/Modify XML File		This action enables developers to read or modify XML files on the target system.
Refresh Windows Environment	E S	This action forces Windows target systems to refresh the environment. Refresh Windows Environment ensures that environment variables you set with the Set System Environment Variable action are immediately available on the target system.
Register Windows Service	E	Registers a Windows Service on the end user's system.  Note • This action is available only in the <i>Install</i> task.
Restart Windows	E S	This action restarts a Windows system. The system reboots as soon as this action is reached. Use this action carefully and only in conjunction with rules.  Note • This action is available in the <i>Install</i> , <i>Post-Install</i> , <i>Pre-Uninstall</i> , and <i>Post-Uninstall</i> tasks.
Set InstallAnywhere Variable - Multiple Variables	E S	The root of any InstallAnywhere installation, these actions allow developers to create, and specify values for, InstallAnywhere variables. Variables defined with this action can be set to use the Evaluate at Assignment option. This action can be used to control nearly any aspect of the installation.  Note • See Evaluation of InstallAnywhere Variables and Setting Variables in the Advanced Designer .

Table 7-129 • General Actions (cont.)

Action	Editions	Description
Set InstallAnywhere Variable - Single Variable	E S	<p>Set the value of a single InstallAnywhere variable.</p> <p>This action can also be used to set a variable to be evaluated at assignment.</p>  <p>Note • See Evaluation of InstallAnywhere Variables and Setting Variables in the Advanced Designer.</p>
Set Windows Registry - Multiple Entries	E	<p>Set multiple Windows registry keys, data, and values on the end user's system.</p>  <p>Tip • If you are targeting 64-bit systems, click the Access Specific Registries View (64-bit systems) check box. Then select whether you want this action to reference the 64-bit or 32-bit portion of the registry.</p>
Set Windows Registry - Single Entry	E S	<p>Set an individual Windows registry key, data, and value on the end user's system.</p>  <p>Tip • If you are targeting 64-bit systems, click the Access Specific Registries View (64-bit systems) check box. Then select whether you want this action to reference the 64-bit or 32-bit portion of the registry.</p>
Show Message Dialog	E	<p>This action creates a modal dialog that requests end user input. The message dialog box appears over the currently displayed panel.</p> <p>For more information, see Show Message Dialog.</p>
Start, Stop, Pause Windows Service	E	<p>If the application is interacting with a Windows Service, the installer may need to manage that service. This action, when the installer is run with sufficient privileges, allows the installer to stop, start, or pause registered windows services.</p>
Uninstall InstallShield Universal Software	E	<p>Uninstalls software previously installed by InstallShield Universal or InstallShield MultiPlatform installers.</p>

Delete Folder

You can use the **Delete Folder** action to delete a folder from the end user's system.

Properties Tab

The **Properties** tab of the **Delete Folder** action customizer includes the following properties:

Table 7-130 • Delete Folder Action Customizer

Property	Description
Original	Choose one of the following to specify whether the folder to delete is being installed, or already exists on the end user's system. <ul style="list-style-type: none">• Installed file—Select this option to delete a folder to be deployed by your installer. Then click Choose Target to open the Choose a Folder dialog box and select the folder you want to delete.• Existing file—Select this option to delete a folder that already exists on the target system, and enter the path (fixed or relative) and file name for that folder in the box.

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Execute Ant Script

Execute Ant Script allows developers to execute scripts designed for the Apache Jakarta Project's Ant application. If this action is selected, InstallAnywhere bundles Ant with the application.

To set up an Execute Ant Script action, set the following options on the customizer:

Table 7-131 • Execute Ant Script Customizer

Option	Description
Ant Build Script	<p>Identify the Ant build script you want to execute by choosing one of the following options:</p> <ul style="list-style-type: none"> • Specify Build Script—Click Choose File to identify the script file you want to use. Navigate to the build script file and click Open. To examine the build script you selected, click View Script. • Use Existing/Installed Build Script—Type the path and filename for an Ant build script file that either exists already or will be installed on the target system.
Specify Build Properties	<p>To optionally identify a build properties file to use with the build script, select this option and then click Choose File to identify the properties file to use.</p>
Substitute IA Variables in Build Properties	<p>Select this option to resolve any InstallAnywhere variables in the build properties file you selected.</p>
Ant Target	<p>To specify an Ant target, choose one of the following options:</p> <ul style="list-style-type: none"> • Use Default Target—Select this option to use the target defined in the build script as the default target. • Use Specified Target—Select this option to use the target you specify in the text box. This option allows you to execute the build script with a target other than the default target.
Show Indeterminate Dialog	<p>Choose this option to show a message dialog while the Ant script is running. In the text box, enter the message that you want the indeterminate dialog box to show when the installer runs this action.</p>
Dependencies	<p>Ant supports a number of additional tasks. These tasks typically require an external library separate from the core Ant tasks. To include an external library, click Add jar or zip and select the external library that you want to include in this action. Click Remove to remove an archive listed in the text box.</p> <p></p> <p>Note • <i>InstallAnywhere automatically adds dependencies required by the Execute Ant Script action.</i></p>



Note • The Execute Ant Script action is only for developers familiar with Ant. For more information, go to <http://ant.apache.org/>.

Execute Command

You can use the **Execute Command** action to run executables as you would from the target system's command line. The Execute Command action customizer has the following options:

Table 7-132 • Execute Command Customizer

Option	Description
Command Line	<p>Enter the syntax to run the executable. Type the name of the executable (command) or the full path to the executable.</p> <ul style="list-style-type: none"> Any text following the command is passed as arguments to the executable. Single arguments that contain spaces must be quoted; otherwise, they are passed as multiple arguments. To pass actual quotation marks as part of an argument, escape them with a backslashes character—for example <code>\\"</code>. All elements in Command Line may be represented either as literal values or by InstallAnywhere variables.
Do not substitute unknown variables	<p>Select this option to avoid the substitution of unknown variables for this action by instructing InstallAnywhere to only resolve InstallAnywhere variables which are listed in the project under Project > Variables (the known variables).</p> <p>Note • For more information, see Preventing the Substitution of Unknown Variables.</p>
Suspend installation until process completes	Select this option if you want the installer to halt until the process launched by the command has returned.
Show indeterminate dialog	If you have chosen the Suspend installation until process completes option, select this option if you want the installer to render a dialog during the execution of this command and show the message you enter in the associated text box. The default message is <code>Executing...</code>
Store process's stdout in	Enter the variable to store the stdout from your command. The default value is <code>\$EXECUTE_STDOUT\$</code> .
Store process's stderr in	Enter the variable to store the stderr from your command. The default value is <code>\$EXECUTE_STDERR\$</code> .

Table 7-132 • Execute Command Customizer (cont.)

Option	Description
Store process's exit code in	Enter the variable to store exit codes. The default value is \$EXECUTE_EXITCODE\$.
Suppress first window (Windows only)	Select this option to suppress the first window on Microsoft Windows platforms. This option is particularly useful in suppressing the appearance of the cmd.exe window when executing batch files, or command line executables.



Note • The **Execute Command** action is not the same as a DOS command (on Windows) nor a shell command (on UNIX). Only executable files on the target system can be called using this action. For example, on Windows, you cannot call echo because it represents a special shell command that cmd.exe and command.com understand. On UNIX, echo can be called because it is an executable file (typically located in /usr/bin); however, you cannot use any special shell commands (the setenv or set C-shell commands).



Note • To execute DOS or UNIX-shell commands, use the **Execute Script/Batch File** action or the **Execute Target File** action. These actions allow you to write a batch file or shell script to run during the installation.

Execute Custom Code

The **Execute Custom Code** action allows developers to extend the functionality of InstallAnywhere. InstallAnywhere's API is purely Java based and allows developers to do nearly anything that is possible in Java. The **Execute Custom Code** action represents the non-interactive interface for this API.

The Execute Custom Code customizer includes the following options:

Table 7-133 • Execute Custom Code Customizer

Option	Description
Path	Identifies the archive that contains all classes needed by the custom code action. <ul style="list-style-type: none"> • Choose JAR or ZIP—Click to locate the archive, and then click Open. • Clear—Click to delete the contents of the Path text box.
Class	Specify the fully qualified class name of the class that is called when the custom code is executed—for example, com.acme.MyAction. <p>The Execute Custom Code action can only access classes included in the archive you choose. You must include the builder version of the class if one is going to be used.</p>

Table 7-133 • Execute Custom Code Customizer (cont.)

Option	Description
Show “Please wait...” panel	Select this option to display a message panel to the user while the execution is occurring.  Note • This option is not available for Execute Custom Code actions added to the Install task.
Show indeterminate dialog	Select to show an indeterminate dialog box during the execution of this command that displays the message you enter in the associated text box. The default message is Executing Custom Code...
Invoke <code>uninstall()</code> method	Choose whether to set the <code>uninstall()</code> method for the custom code to run before or after files or folders are uninstalled.  Note • This option is only available in the Install task.
Dependencies	Custom code tasks sometimes are dependent upon other code. To bundle code archives with the installer, click Add jar or zip and select the archives of code on which your custom code depends. Click Remove to remove a listed archive.  Note • For information on the support of signed JARs as dependencies, see Support for Signed JARs as Dependencies .
Open Javadocs	Click to launch the Javadocs for the InstallAnywhere API in your default Web browser.

**Note** • For more information, see [Custom Code APIs](#).**Note** • You can show progress while an action installs using the **Execute Custom Code** customizer. See the sample in the `<InstallAnywhere>/CustomCode/Samples/SampleProgress` folder. See the Javadocs for more information about the classes and APIs in this sample.

Execute Script/Batch File

Execute Script/Batch File allows developers to enter the text of a script or batch file which the installer executes on the target system. When this action runs, it first resolves any InstallAnywhere variables in the script. It then saves the script to the InstallAnywhere temp directory and executes the script. The **Execute Script/Batch File** action finishes by cleaning up and deleting the script from the temp directory.

The Execute Script/Batch File customizer includes the following options:

Table 7-134 • Execute Script/Batch File Customizer

Option	Description
Comment	Enter a comment to describe the function of the script or batch file.
Script	Type the script you want the installer to run. Start your script with a <code>cd <directory></code> command to establish the directory from which the script will run. By default, scripts and batch files are run in a <code>temp</code> directory.
Do not substitute unknown variables	Select this option to avoid the substitution of unknown variables for this action by instructing InstallAnywhere to only resolve InstallAnywhere variables which are listed in the project under Project > Variables (the known variables).
	 <i>Note • For more information, see Preventing the Substitution of Unknown Variables.</i>
Suspend installation until process completes	Select this option if you want the installer to halt until the script or batch file processes complete.
Show indeterminate dialog	If you have chosen the Suspend installation until process completes option, select this option if you want the installer to display a dialog during the execution of this command and show the message you enter in the associated text box. The default message is <code>Executing Installation Script...</code>
Store process's stdout in	Enter the variable to store the <code>stdout</code> from your command. The default value is <code>\$EXECUTE_STDOUT\$</code> .
Store process's stderr in	Enter the variable to store the <code>stderr</code> from your command. The default value is <code>\$EXECUTE_STDERR\$</code> .
Store process's exit code in	Enter the variable to store exit codes. The default value is <code>\$EXECUTE_EXITCODE\$</code> .

Find Component in Registry

If your installer uses a component already installed on your target system, or one that should already be installed on the target system, you can use the **Find Component in Registry** action to locate that component. The **Find Component in Registry** action is used to determine if a component exists on a system by searching the cross-platform registry, as well as to discover existing component versions, their location, and if there are multiple instances of a particular component on the destination system. You can then utilize that component in your installation, or use that installation location as a path within your installation.



Note • This action is referring to the InstallAnywhere registry, not the Windows registry.

The **Find Component in Registry** action customizer includes the following options:

Table 7-135 • Find Component in Registry Action Customizer

Option	Description
Comment	Enter a comment to identify the component that this action is searching for.
Find Component	Enter the UUID of the component that this action is searching for, the unique identifier specified for the component. If you have a project that contains the desired component, the UUID is visible in the component's customizer. Otherwise, if a product has installed the component on a system, the component ID can be found in the InstallAnywhere registry.
Highest version only	Select this option to find only the highest version of this component.
Compare version	Select this option to instruct the installer to compare versions. Select an operator from the list (Lower than , Lower or equal to , Equal to , Higher or equal to , or Higher than) and enter a component version in the box.
Key File location	Select this option to have the installer search for a specified component key file, and enter the key file location in the box. A key file is a single file that identifies the component.
Components found count	Name of variable that contains the count of components that this action finds. By default, the variable is \$REG_COMPONENT_COUNT\$. After the action runs, you can use this variable in subsequent actions and rules. For example, to display a panel only if a component is not found, you can add a Display Message panel with a rule that uses the value of \$REG_COMPONENT_COUNT\$.
Components found versions	Name of variable that contains the versions of the components that this action finds. By default, the variable is \$REG_COMPONENT_VERSION\$. After the action runs, you can use this variable in subsequent actions and rules. For example, to display a panel only if a specific version of a component is found, you can add a Display Message panel with a rule that uses the value of \$REG_COMPONENT_VERSION\$.
Components found locations	Name of the variable that contains the locations of the components that this action finds. By default, the variable is \$REG_COMPONENT_LOCATION\$. After the action runs, you can use this variable in subsequent actions and rules. For example, to display a panel only if a component is found in a specific location, you can add a Display Message panel with a rule that uses the value of \$REG_COMPONENT_LOCATION\$.

Modify Text File - In Archive

You can use the **Modify Text File - In Archive** action to alter text files within an archive (ZIP or JAR). The **Modify Text File - In Archive** customizer includes the following options:

Table 7-136 • Modify Text file - In Archive Customizer

Option	Description
Installed Archive	Choose this option to modify files in an archive your installer deploys on the target system. (Only available in the Install and Post-Install tasks.)
Existing Archive	Choose this option to modify files in an archive that already exists on the target system.  Note • Ensure the name and path to the file you specify matches, including case-sensitivity, to the name of the target file.
Archive Path	Type the path to the file you want to modify in the archive.
Change Line Endings To	<p>Specify the type of line ending to use in the text file:</p> <ul style="list-style-type: none"> • User's Platform (Default) • Mac OS • UNIX • DOS/Windows <p>To ensure that the line endings conform to the native line endings for the target system, choose User's Platform.</p>
Additional Text	<p>Use this option to instruct the installer to add additional text to text file. Choose one of the following options:</p> <ul style="list-style-type: none"> • Prepend—Select this option to add additional text to the beginning of the text file. Then enter the desired additional text in the box. • Append—Select this option to add additional text to the end of the text file. Then enter the desired additional text in the box. • None—Select this option if you do not want the installer to add any additional text.
Substitute InstallAnywhere variables in file	Select to replace any InstallAnywhere variables with their values. When this option is selected, the installer changes any strings in the text file that match InstallAnywhere variables to the values of those variables.

Table 7-136 • Modify Text file - In Archive Customizer

Option	Description
Search and replace strings	If you want the installer to search for specific text strings in the text file and replace them with replacement strings, select this option and then click Configure to open the Configure Search and Replace Strings dialog box. On the Configure Search and Replace Strings dialog box, click Add to define the Search For and Replace With text. When this action runs, the installer locates all instances of the Search For text and replaces them with their corresponding Replace With text.

Modify Text File - Multiple Files

You can use the **Modify Text File - Multiple Files** action to alter several text files on the target system. Use this action any time you need to make the same changes to multiple text files within the same directory.

The **Modify Text File - Multiple Files** customizer includes the following options:

Table 7-137 • Modify Text File - Multiple Files Customizer

Option	Description
Installed Folder	Select this option if you want the installer to modify files in a folder the installer deploys on the target system.  Note • Only available in the Install and Post-Install tasks.
Existing Folder	Select this option if you want the installer to modify files in a folder that already exists on the target system.
Filter Files	Filters are conditions that the installer uses to select files for modification. Click Configure Filter to open the Filter File dialog box. See Filter File Dialog Box for more information.
Change Line Endings To	Specify the type of line ending to use in the text file: <ul style="list-style-type: none">• User's Platform (Default)• Mac OS• UNIX• DOS/Windows To ensure that the line endings conform to the native line endings for the target system, choose User's Platform .

Table 7-137 • Modify Text File - Multiple Files Customizer

Option	Description
Additional Text	<p>Use this option to instruct the installer to add additional text to text file. Choose one of the following options:</p> <ul style="list-style-type: none"> • Prepend—Select this option to add additional text to the beginning of the text file. Then enter the desired additional text in the box. • Append—Select this option to add additional text to the end of the text file. Then enter the desired additional text in the box. • None—Select this option if you do not want the installer to add any additional text.
Substitute InstallAnywhere variables in file	Select to replace any InstallAnywhere variables with their values. When this option is selected, the installer changes any strings in the text file that match InstallAnywhere variables to the values of those variables.
Create backup	Select to create a backup of an existing or installed file prior to modification. <ul style="list-style-type: none"> • The backup file name is determined by adding .backup to the original file name. • Backup files are not removed by the uninstaller.
Search and replace strings	<p>If you want the installer to search for specific text strings in the text file and replace them with replacement strings, select this option and then click Configure to open the Configure Search and Replace Strings dialog box.</p> <p>On the Configure Search and Replace Strings dialog box, click Add to define the Search For and Replace With text. When this action runs, the installer locates all instances of the Search For text and replaces them with their corresponding Replace With text.</p>

Modify Text File - Single File

You can use the **Modify Text File - Single File** action to modify a text file on the target system. The **Modify Text File - Single File** customizer includes the following options:

Table 7-138 • Modify Text File - Single File Customizer

Option	Description
Installed File	<p>Select this option if you want the installer to modify a file the installer deploys on the target system. Click the Choose Target button to open the Choose a Text File dialog box and select the installed file you want the installer to modify.</p>  <p>Note • Only available in the <i>Install</i> and <i>Post-Install</i> tasks.</p>

Table 7-138 • Modify Text File - Single File Customizer

Option	Description
Existing File	Select this option if you want the installer to modify a file that already exists on the target system. Enter the name of the file in the text box.
New File	Select this option if you want the installer to create a new text file on the target system. Enter the name of the file in the text box.
Change Line Endings To	<p>Specify the type of line ending to use in the text file:</p> <ul style="list-style-type: none"> • User's Platform (Default) • Mac OS • UNIX • DOS/Windows <p>To ensure that the line endings conform to the native line endings for the target system, choose User's Platform.</p>
Additional Text	<p>Use this option to instruct the installer to add additional text to text file. Choose one of the following options:</p> <ul style="list-style-type: none"> • Prepend—Select this option to add additional text to the beginning of the text file. Then enter the desired additional text in the box. • Append—Select this option to add additional text to the end of the text file. Then enter the desired additional text in the box. • None—Select this option if you do not want the installer to add any additional text.
Substitute InstallAnywhere variables in file	Select to replace any InstallAnywhere variables with their values. When this option is selected, the installer changes any strings in the text file that match InstallAnywhere variables to the values of those variables.
Create backup	<p>Select to create a backup of an existing or installed file prior to modification.</p> <ul style="list-style-type: none"> • The backup file name is determined by adding .backup to the original file name. • Backup files are not removed by the uninstaller.
Search and replace strings	<p>If you want the installer to search for specific text strings in the text file and replace them with replacement strings, select this option and then click Configure to open the Configure Search and Replace Strings dialog box.</p> <p>On the Configure Search and Replace Strings dialog box, click Add to define the Search For and Replace With text. When this action runs, the installer locates all instances of the Search For text and replaces them with their corresponding Replace With text.</p>

Read/Modify XML File

You can use the **Read/Modify XML File** action to modify an XML file on the target system.



Note • To perform complex modifications of an XML file, you can use multiple **Read/Modify XML File** actions. See [Performing Complex XML File Modifications](#).

Properties Tab

The **Properties** tab of the **Read/Modify XML File** action customizer includes the following options:

Table 7-139 • Read/Modify XML File Customizer

Option	Description
Installed File	Select this option if you want the installer to modify a file the installer deploys on the target system. Click the Choose Target button to open the Read/Modify XML File dialog box and select the installed file you want the installer to modify.
Existing File	Select this option if you want the installer to modify a file that already exists on the target system. Enter the name of the file in the text box.
Read value	Select this option if you want the installer to read a value in the specified XML file. If you choose this option, the following additional options are listed: <ul style="list-style-type: none">• Entire Tag—Select this option to read an entire XML tag in the file and specify the Tag in the text box.• Specific Attribute—Select this option to read a specific attribute of a tag in the XML file and specify that Tag and Attribute in the text boxes.• Variable to store value—Enter the name of an InstallAnywhere variable that you want to store the value that is read.

Table 7-139 • Read/Modify XML File Customizer

Option	Description
Replace value	<p>Select this option if you want the installer to search for a specific value in the XML file and replace it with a replacement value. If you choose this option, the following additional options are listed:</p> <ul style="list-style-type: none"> • Entire Tag—Select this option to search for an entire XML tag in the file and specify that Tag in the text box. • Specific Attribute—Select this option to search for a specific attribute of a tag in the XML file and specify that Tag and Attribute in the text boxes. • Replace by—Enter the replacement value for the specified Tag or Attribute. • Create backup—Select to create a backup of the original XML file.  <p>Note • The name of the Tag can be specified along with its occurrence using the format of <code>tagname{occurrence#}</code>, such as <code>book{2}</code>, where occurrence is a zero based occurrence of the tag book.</p>
Find occurrence of	<p>Select this option if you want the installer to find a specific occurrence of a value of a tag or of a value of an attribute of a tag in the specified XML file. If you choose this option, the following additional options are listed:</p> <ul style="list-style-type: none"> • A tag—Select this option to search the XML file for a specific occurrence of a value of a tag. Then enter the name of that tag and the value in the Tag and with value text boxes. • A tag with specific attribute / Tag / Attribute / with value—Select this option to search the XML file for a specific occurrence of a value of an attribute of a tag. Then enter the name of that tag, attribute, and the value in the Tag, Attribute, and with value text boxes. • Variable to store value—Enter the name of an InstallAnywhere variable that you want to store the value that is found  <p>Note • The Tag specified should be an atomic/leaf node tag in the XML file.</p>

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use to add rules to the selected action. For more information, see [Assigning a Rule to an Action](#) and [Rules Reference](#).
- **Tags**—Use to add Build Configuration Tags to the selected action. See [Assigning Tags to Project Elements](#).
- **Rollback**—Use to specify rollback options for the selected action. See [Using the Rollback Subtab of the Install Task to Fine-Tune a Rollback](#).

Performing Complex XML File Modifications

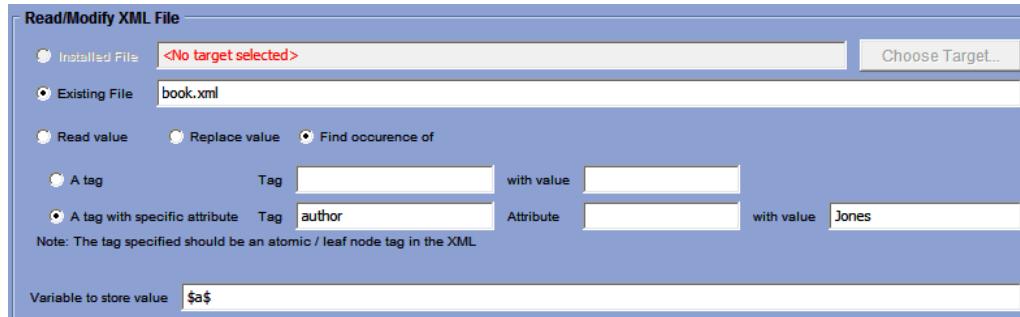
To perform complex modifications, you can use multiple **Read/Modify XML File** actions. For example, suppose you had the following XML file:

```
<?xml version="1.0"?>
<catalog>
  <book id="295">
    <author>Smith</author>
    <title>XML Developer's Guide</title>
    <price>44.95</price>
  </book>
  <book id="296">
    <author>Jones</author>
    <title>Learning XML</title>
    <price>19.95</price>
  </book>
  <book id="297">
    <author>Wilson</author>
    <title>Advanced CSS</title>
    <price>9.95</price>
  </book>
</catalog>
```

Figure 7-80: Sample XML File

In this XML file, if you wanted to replace the value of the `id` attribute with a value of `300` instead of `296` for those `book` elements that have an `author` subelement with a value of `Jones`, you would use the following two **Read/Modify XML File** actions:

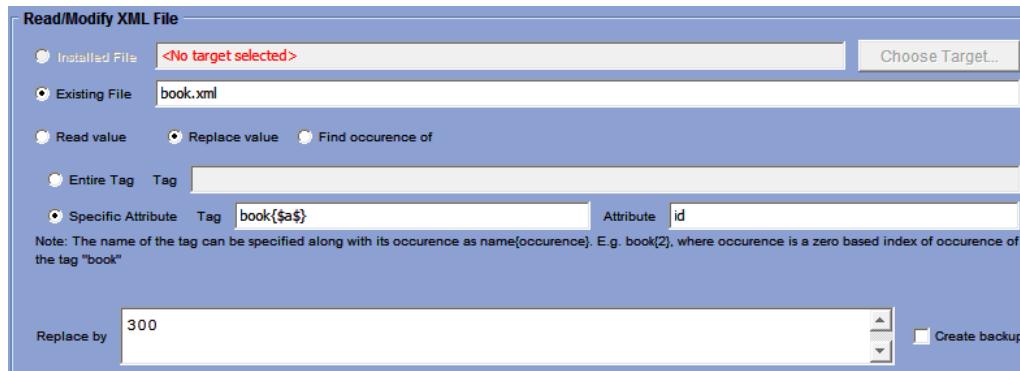
- Use the first **Read/Modify XML File** action with the **Find occurrence of** option selected to find a `book` element with an `author` subelement with a value of `Jones`.



Chapter 7: Reference

Actions

- Use the second **Read/Modify XML File** action with the **Replace value** option selected to do the actual replace.



Note • If there are multiple nodes in the XML file that satisfy a given set of criteria specified in the customizer of the **Read/Modify XML File** action, then only the first node is returned by the result value.

Show Message Dialog

This action creates a modal dialog that requests end user input. The message dialog box appears over the currently displayed panel.

You can use this action to force the end user to return to the previous panel, exit the installer, or input information. When controlled by rules, the message dialog can also serve as a data verification tool.

The following options are included:

Table 7-140 • Show Message Dialog Options

Option	Description
Title	Enter the text that will appear in the title bar of the message dialog.
Label	Enter the title of the message dialog that will appear in bold above the message text.
Message	Enter the text of the message you want to display on the dialog. You may enter as much text as you feel is necessary; the dialog will expand to display all of the text that you enter.
Dialog Type	Select one of the following options to specify which icon you want to display on this message dialog: Warning , Error , Information , or Query .
Button 0, 1	Specify whether you want to include an OK and Cancel button on this message dialog, and set the names of those buttons. An OK button is required. For each of these buttons, specify the action to take when the button is clicked: Return to Previous Panel , Continue to Next , or Cancel and Exit .

Table 7-140 • Show Message Dialog Options

Option	Description
Button 2	<p>Select this option if you want to include a button that will open another dialog containing more detailed information, depending upon the action the user takes on this dialog. By default, the name of this button is Details...</p> <p>To specify which dialog box would be displayed when the user clicks Details, you would add an additional dialog that has a Compare InstallAnywhere Variables rule applied to it which would check for the value of the Results Variable defined on this Show Message Dialog.</p> <p>Also, specify the action to take when the button is clicked: Return to Previous Panel, Continue to Next, or Cancel and Exit.</p>  <p>Note • If you add a Show Message Dialog to the Pre-Install, Post-Install, Pre-Uninstall, or Post-Uninstall phase, then the Return to Previous Panel option will be enabled. If a Show Message Dialog is added to the Install or Uninstall task, Return to Previous Panel will be disabled.</p>
Results Variable	<p>This variable is used to collect the results of the user action on the Show Message Dialog. For example, if the variable is set to \$CHOSEN_DIALOG_BUTTON\$, the value of this variable would vary depending upon which action the user takes:</p> <ul style="list-style-type: none"> • Next—If the user clicks Next, the variable would be set to 0. • Previous—If the user clicks Previous, the variable would be set to 1. • Details—If the user clicks Details, the variable would be set to 2.

Panel Actions

The following table lists the panel actions available in InstallAnywhere.

Table 7-141 • Panel Actions

Action	Editions	Description
Choose Alias, Link, Shortcut	E S	Added as part of the default project, this panel allows the end user to choose an installation location for “shortcuts” (Windows), “aliases” (Mac), and “links” (Unix).
Choose Features to Uninstall	E S	This panel allows the end user to select which features they want to uninstall.
Choose File	E	Choose File allows installers to request that the user select a file on certain criteria and set its result as an InstallAnywhere Variable. The variable can then be used later in the Install.

Table 7-141 • Panel Actions (cont.)

Action	Editions	Description
Choose Folder	E	Choose Folder allows developers to request that the user select a folder on certain criteria and set its result as an InstallAnywhere variable that can be used later in the Install.
Choose Install Folder	E S	Part of the default project, this panel allows the user to choose the primary installation folder. If you omit this panel, the Default Install Folder settings from the Project > Platforms subtasks apply.
Choose Install Sets	E S	This panel allows developers to request that the end user choose an install set or features to install.
Choose Java VM	E S	This panel allows end users to select the Java VM to use for any installed LaunchAnywhere launchers.
Choose Uninstall Type	E S	Allows the end user to select whether to install all or part of the application
Custom Code	E	InstallAnywhere's Custom Code API allows developers to create custom panels where necessary.  <i>Note</i> • For information on the support of signed JARs as dependencies, see Support for Signed JARs as Dependencies .
Disk Space Check	E S	This action performs a disk space check on the installation destination system based on the end user's chosen install location and the end user's chosen features. If there is not enough disk space to perform the install, then the installer prompts the end user to free the required disk space or choose another install location.  <i>Note</i> • This action is automatically added before files are installed. The action does not appear in the list.
Display HTML	E S	The Display HTML panel allows developers to display HTML from an archived file or a specific URL on a panel during the installation.
Display Message	E S	The Display Message panel allows developers to simply display a text message to the end user during the installation. This can be useful for conveying information about installation choices that the end user has made. This panel is also particularly useful in debugging installer issue having to do with InstallAnywhere variables.

Table 7-141 • Panel Actions (cont.)

Action	Editions	Description
Find File/Folder	E	This panel implements a search process that, depending on specifications made by the developer, searches portions of the file system for a specific named file or for a file that matches a certain pattern. The end user can also choose a matching file.
Get Password	E	This panel allows developers to request a password from the end user. Developers can choose to validate the password against a list of specified passwords (enabling the index feature which allows different passwords to effectively unlock different features) or they can simply store the entered password in a variable (as when requesting a password to be used in a configuration routine). InstallAnywhere automatically encrypts password values according to the Security settings on the Project > Variables subtask.
Get Serial Number	E	Implementing InstallAnywhere's built-in serial number verification and creation routines, this action/panel allows developers to add serial number functionality to the installer. Developers can choose to generate any number of serial numbers for any number of products. Serial numbers can represent unique products or sets of products. The result of this action allows developers to create rules that can manage all aspects of the installation based on rights granted by the serial number the end user has entered.
Get User Input - Advanced	E	This action allows developers to get input from the user using multiple input types, and setting multiple variables. This action can have radio buttons, check boxes, text fields, and menus—all on the same panel. (See Get User Input Panels for more information.)
Get User Input - Simple	E	This action allows developers to request input from the end user. (See Get User Input Panels for more information.)
Important Note	E S	The Important Note panel allows developers to display a text or HTML file without the radio buttons found on the license agreement panel. It is particularly useful for displaying Readme or errata type documents.
Install Complete	E S	This action displays information about the installation's status to the end user. It also optionally displays if a restart is needed on Windows system. (Available only after files have been installed.)
Introduction	E S	Part of the default project, this panel offers an introduction to the installation.

Table 7-141 • Panel Actions (cont.)

Action	Editions	Description
License Agreement		<p>This panel displays license information to the user and prompts them to Agree (I accept the terms of the License Agreement) or Disagree (I do NOT accept the terms of the License Agreement).</p> <p>You are prompted to specify a text file (either HTML or plain text) containing the license information.</p> <p>You can select the Force user to scroll through license agreement option to force scroll through before allowing the user to accept the terms and move to the next panel.</p>
Windows Execution Level	E S	<p>This panel allows developers to display a license agreement to the end user. The end user must choose to accept the agreement in order to continue. Developers can set the default state of the radio buttons (Accept or Decline) and choose a file to use for a license agreement. The License Agreement panel can also utilize HTML files, which gives developers a degree of control over the text formatting and allows them to link to external documents.</p>
Pre-Install Summary	E S	<p>This panel summarizes information collected and evaluated prior to the installation of files. It is included in the default InstallAnywhere project.</p>
Scrolling Message	E	<p>This panel allows developers to enter long text in a message panel that includes scroll bars. This is particularly useful for instructions.</p>
Uninstall Complete	E S	<p>Displays information that the uninstaller has completed</p>
Uninstaller Introduction	E S	<p>This panel offers an introduction to the installer</p>

Choose Install Sets

On the **Choose Install Sets** customizer, you can specify the types of install set/feature selection options the end users see. You can specify that the end users can only select an install set, or select an install set followed by specific features in that install set, or just select specific product features.

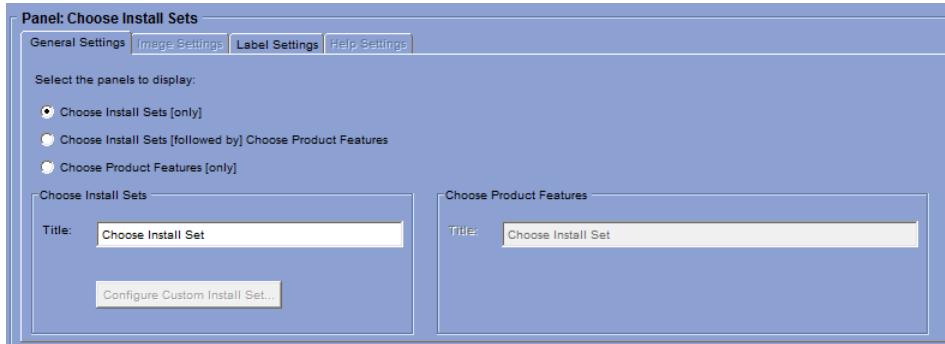


Figure 7-81: Choose Install Sets Customizer



Note • The **Choose Product Features** panel is a component of the **Choose Install Set** panel. You cannot select the **Choose Product Features** panel action from the **Choose an Action** dialog box.

The **Choose Install Sets** customizer includes controls on the following tabs:

- [General Settings](#)
- [Image Settings, Label Settings, and Help Settings](#)

General Settings

The **General Settings** tab of the **Choose Install Sets** customizer includes the following controls:

Table 7-142 • Choose Install Sets Customizer / General Settings Tab

Control	Description
Select the panels to display	Select one of the following options to specify which panels the installer will display: <ul style="list-style-type: none"> • Choose Install Sets [only]—Display only the Choose Install Sets panel, which prompts the user to select which Install Set to install. Install sets are a set of product features which represent high-level, easily selectable, installation options (such as Typical, Minimal, and Custom). • Choose Install Sets [followed by] Choose Product Features—First display the Choose Install Sets panel followed by the Choose Product Features panel, which prompts the user to select which individual features to install. • Choose Product Features [only]—Display only the Choose Product Features panel.
Choose Install Sets / Title	Sets the panel title for the Choose Install Set panel.
Configure Custom Install Set	Click to open a dialog box in which you can customize the name, description, and image associated with the Custom install set.
Choose Product Features / Title	Sets the panel title for the Choose Product Features panel.



Important • If the **Choose Install Sets** panel is in an Add Features or Repair Installation action group, you will only be able to select the third option: **Choose Product Features**. The other two choices will be disabled.

Image Settings, Label Settings, and Help Settings

For information on using the options on the **Image Settings**, **Label Settings**, and **Help Settings** tabs, see [Panel Action Settings](#).

Choose Java VM

You can use the **Choose Java VM** panel to enable end users to select the Java VM to use for any installed LaunchAnywhere launchers.

- Developers can specify whether the installer offers to install a VM bundled with the installer.
- Developers can also permit the user to provide additional search paths or to choose a specific VM available to the target system.

The **Choose Java VM** customizer includes controls on the following tabs:

- General Settings
- Image Settings, Label Settings, and Help Settings

General Settings

The **General Settings** tab of the **Choose Java VM** customizer includes the following controls:

Table 7-143 • General Settings Tab of Choose Java VM Customizer

Control	Description
Title	Enter the title of the Choose Java Virtual Machine panel of the installer. By default, the title is Choose Java Virtual Machine .
Prompt	Enter the instructions that you want to display to the end user on the Choose Java Virtual Machine panel of the installer. By default, the prompt text is Please Choose a Java VM for Use by the Installed Application .
[No checkboxes selected]	If none of the following three options are selected, the Choose Java VM panel only allows the end user to choose from those valid VMs found on the project-defined paths, which are set on the Windows and UNIX subtasks of the Project > Platforms task.
Allow the end user to install the bundled VM	Makes the VM you bundle with your project available in the Choose Java VM panel.
Allow the end user to search for locations other than the paths defined under Project > JVM Settings > Search Panel Settings	Adds a Search Another Location button to the Choose Java VM panel. This button opens the Browse for Folder dialog box, which prompts your users to specify an additional search path.
Allow the end user to choose a specific Java executable	Adds a Choose Java Executable button to this panel. This button shows an Open dialog in which you users can locate the executable for a specific VM and select it as the VM your LaunchAnywhere launchers use.



Note • The behavior of this panel can be affected by the VM Search Settings, specified under the Project > **JVM Settings** subtask, and by the Windows/UNIX JVM Search paths, specified on the **Search Panel Settings Tab** of the **Project > JVM Settings** subtask.



Note • The **Choose Java VM** panel is not shown in Mac OS X installers. To force a Mac OS X installer to use a different VM, use the **Select Java VM for LaunchAnywhere** control on the **Project > Platforms > Mac OS X** subtask.

Image Settings, Label Settings, and Help Settings

For information on using the options on the **Image Settings**, **Label Settings**, and **Help Settings** tabs, see [Panel Action Settings](#).

Display HTML

The **Display HTML** panel allows developers to display HTML from an archived file or a specific URL on a panel during the installation. The **Display HTML** customizer consists of the following tabs:

- [General Settings](#)
- [Image Settings, Label Settings, and Help Settings](#)

General Settings

On the **General Settings** tab of the **Display HTML** customizer, you assign the HTML panel a title and select the source of the HTML.

Table 7-144 • Display HTML Customizer / General Settings Tab

Control	Description
Title	Enter a title for the HTML panel.
Select Source	Specify the source of the HTML file by selecting one of the following options: <ul style="list-style-type: none">• Path—Select this option and click Choose File to locate an archive (.zip or .jar) that includes the HTML you want this panel to display. Then, enter the name of an HTML file from that archive in the Initial Page text box.• Existing URL—Enter the URL for the page you want the Display HTML panel to render. If you are entering an HTTP location, use the following syntax: <code>http://www.mywebsite.com/HTMLFile.html</code> If you want to display a HTML page located on the target system, precede the location with <code>file:///</code>. For example: <code>file:///C:/MyHTMLFiles/HTMLFile.html</code>



Note • When entering a location on the target system, the standard syntax would be precede the file location with `file://`. However, in order for this to render properly, extra java escape characters (forward slashes) are required: `file:///`.

Table 7-144 • Display HTML Customizer / General Settings Tab

Control	Description
Read Form Elements as InstallAnywhere Variables	Select this option to expose any HTML form variables in your installer. This allows you to refer to the form variables in subsequent installer steps by using the following syntax: \$<var_name>\$. For example, to include the value of a form variable named <i>file</i> , simply enclose the variable name in dollar signs: \$ <i>file</i> \$.



Tip • A sample custom HTML panel is available in: <InstallAnywhere>/CustomCode/Samples/HTMLPanelSample/.

Image Settings, Label Settings, and Help Settings

For information on using the options on the **Image Settings**, **Label Settings**, and **Help Settings** tabs, see [Panel Action Settings](#).

Pre-Install Summary

The **Pre-Install Summary** panel summarizes information collected and evaluated prior to the installation of files. It is included in the default InstallAnywhere project. The **Pre-Install Summary** panel also allows you to customize what information is presented.

The **Pre-Install Summary** customizer consists of the following tabs:

- [General Settings](#)
- [Image Settings, Label Settings, and Help Settings](#)

General Settings

The **General Settings** tab of the **Pre-Install Summary** customizer includes the following controls:

Table 7-145 • Pre-Install Summary Customizer / General Settings Tab

Control	Description
Title	Enter a title for the panel.
Prompt	This text is displayed on the Pre-Install Summary panel to explain its purpose. Enter a new prompt or keep the existing value: <i>Please Review the Following Before Continuing:</i>

Table 7-145 • Pre-Install Summary Customizer / General Settings Tab

Control	Description
Include the following information in the Pre-Installation Summary panel	<p>Select the information that you want to display on this Pre-Install Summary panel. Options include the following:</p> <ul style="list-style-type: none"> • Product name • Install folder • Alias, Link, Shortcut folder • Selected install sets • Selected product features • VM install folder/System VM • Disk space information <p>If you select the Disk space information option, also select the increment type to use from the list: Bytes, KiloBytes, MegaBytes, or GigaBytes. The Disk space information will be based up on these selections.</p>  <p>Note • For information on checking disk space, see Checking Disk Space During Installation.</p>
Edit Custom Fields	Click Edit Custom Fields to open the Edit Custom Fields dialog box, where you can provide the Variable Name , Value , and Text Reading Order of variables you wan to display on this panel.
Custom Fields List	List of the custom variable fields that you have added to display on this panel. Information displayed includes: <ul style="list-style-type: none"> • Field Heading—Name of variable. • Field Value—Value of variable. • Text Reading Order—If you are producing installers for Arabic or Hebrew locales, you may also use this field to override the text orientation for each variable. Choose from Based on Locale, Left-to-Right, or Right-to-Left.

Image Settings, Label Settings, and Help Settings

For information on using the options on the **Image Settings**, **Label Settings**, and **Help Settings** tabs, see [Panel Action Settings](#).

Console Actions

Console Actions (commonly called Consoles) are the means for requesting installer end user input when using a command line interface. When the end user selects a console installation, console actions will be used instead of panel actions.

Table 7-146 • Console Actions

Action	Editions	Description
Choose Features to Uninstall	E	This console allows the end user to select which features they want to uninstall.
Choose Install Folder	E	Chooses the primary installation location
Choose Install Sets	E	This console allows developers to request that the end user choose an install set, or features to install.
Choose Java VM	E	This console allows developers to have the end user select the Java VM that will be used for any installed LaunchAnywhere Launchers. Developers can specify the type of VM that the end user should select, and the panel will search the system for an appropriate VM.
Choose Link Folder	E	This console allows the user to determine where to install UNIX links
Choose Uninstall Type	E	Allows the end user to select whether to uninstall all or part of the application
Custom Code	E	InstallAnywhere's Custom Code API allows developers to create custom consoles where necessary.  <i>Note • For information on the support of signed JARs as dependencies, see Support for Signed JARs as Dependencies.</i>
Display Message	E	The Display Message console allows developers to simply display a text message to the end user during the installation.
Get Password	E	This console allows developers to request a password from the end user. InstallAnywhere automatically encrypts password values according to the Security settings on the Project > Variables subtask.
Get Serial Number	E	This console allows end users to generate a list of serial numbers as well as request them from the end user

Table 7-146 • Console Actions (cont.)

Action	Editions	Description
Get User Input	E	This console allows developers to request input from the end user. See Get User Input Panels for more information.
Install Complete	E	This action displays information about the installation's status to the end user. Available only after files have been installed.
Install Failed	E	This console should be displayed when a console installer has generated an error.
License Agreement	E	This console allows developers to display a license agreement to the end user.
Pre-Install Summary	E	This console summarizes information collected and evaluated prior to the installation of files.
Ready to Install	E	This console alerts the end user that the installer is about to install files.
Show Message Console 'Dialog'	E	Displays a message to the end user
Uninstall Complete	E	Displays information that the uninstaller has completed
Uninstaller Introduction	E	This console offers an introduction to the installer

System i (i5/OS) Actions

InstallAnywhere Enterprise Edition includes actions specifically for i5/OS systems on System i. Some of these actions require access to an i5/OS machine at build time.

Table 7-147 • System i (i5/OS) Actions

Action	Description
Choose Remote System i (i5/OS) Install Folder	Allows the user to choose the installation folder on a remote System i (i5/OS) machine.
Get System i (i5/OS) Login Credentials	Requests i5/OS login credentials from the user.
System i (i5/OS) Command	Calls an i5/OS command with the specified parameters.

Table 7-147 • System i (i5/OS) Actions (cont.)

Action	Description
System i (i5/OS) Find Component in RAIR	Queries RAIR to find components in registry by their Product Name, Component Name, Component Version, and Feature Name (the first feature of the component).
System i (i5/OS) Install File	Enables files to be installed in the Integrated File System (IFS) on a remote i5/OS system.
System i (i5/OS) Integrated File System (IFS)	Enables directories to be installed in the Integrated File System (IFS) on a remote i5/OS system.
System i (i5/OS) Library	Enables libraries to be restored on an i5/OS system. At build time, the Save Library (SAVLIB) command is called to save the library from the i5/OS build system.
System i (i5/OS) Licensed Program	Enables licensed programs to be restored on an i5/OS system.
System i (i5/OS) Object	Enables one or more objects in the same library to be restored to an i5/OS system.
System i (i5/OS) Process Remote Install (Merge Modules)	Installs an InstallAnywhere Merge Module on the target system.
System i (i5/OS) Program	Saves an i5/OS program (*PGM) object from the i5/OS build system and then restores and calls the program during the installation.
System i (i5/OS) Program Temporary Fix (PTF)	Saves an i5/OS program temporary fix (PTF) from the i5/OS build system and then loads and applies the PTF during the installation.



Note • If you include these actions in your project, you will be prompted for i5/OS sign-on information when you build your project. Once you have signed on, the same i5/OS system is used in conjunction with your local build system until you restart InstallAnywhere.

Choose Remote System i (i5/OS) Install Folder

This panel allows the user to choose the installation folder on a remote System i (i5/OS) machine. This action includes the following options:

- General Settings
- Image Settings, Label Settings, and Help Settings

General Settings

The Choose Remote System i (i5/OS) Install Folder action includes the following options on the **General Settings** tab:

Table 7-148 • Choose Remote System i (i5/OS) Install Folder / General Settings Options

Option	Description
Title	Type a new title or keep the default title: Choose a Remote IFS Folder
Instructions	Type new instructions or retain the default instructions: Please choose a destination IFS folder for this installation on the remote System i (i5/OS). To hide this field in the installer panel, leave this text box empty.
Prompt	Type a new prompt or keep the existing value: To &which IFS directory would you like to install?  Tip • Notice that the ampersand in the default prompt creates a mnemonic for quick keyboard access to this field.  Note • This panel sets the value for the InstallAnywhere variable \$OS400_INSTALL_DIR\$.

Image Settings, Label Settings, and Help Settings

For information on using the options on the **Image Settings**, **Label Settings**, and **Help Settings** tabs, see [Panel Action Settings](#).

Get System i (i5/OS) Login Credentials

The Get System i (i5/OS) Login Credentials action requests i5/OS login credentials from the user. This action includes the following options:

- General Settings
- Image Settings, Label Settings, and Help Settings

General Settings

The Get System i (i5/OS) Login Credentials action includes the following options on the **General Settings** tab:

Table 7-149 • Get System i (i5/OS) Login Credentials / General Settings Options

Option	Description
Title	Keep the default panel title or customize it by editing the text. The default value is: <i>Enter Sign on Credentials (i5/OS System, User, and Password)</i>
Instructions	Keep the default instructions or customize them by editing the text. The default instructions are: <i>This installation requires an iSeries system name, user name, and password to continue.</i>
System Name Prompt	Enter the text to show when this action requests a system name. The default value is: <i>Please Enter the S&ystem Name:</i>
User Name Prompt	Enter the text to show when this action requests a user name. The default value is: <i>Please Enter the &User Name:</i>
Password Prompt	Enter the text to show when this action requests a password. The default value is: <i>Please Enter the Pass&word:</i>
Echo character	Click the check box to obscure the password characters a user types with the character in the text box. The default value is: *.
Connections to i5/OS Servers	Controls allow you to preset the initial and maximum number of connections to the i5/OS server: <ul style="list-style-type: none">• Maximum number—Enter the number that represents the maximum number of connections to the i5/OS server. The default value is 10.• Initial number—Enter the number that represents the initial number of connections to the i5/OS server. The default value is 2.

Image Settings, Label Settings, and Help Settings

For information on using the options on the **Image Settings**, **Label Settings**, and **Help Settings** tabs, see [Panel Action Settings](#).

System i (i5/OS) Command

The **System i (i5/OS) Command** action calls an i5/OS command with the specified parameters. Specify the following options.

Table 7-150 • System i (i5/OS) Command Options

Option	Description
Enter Command Name	Enter the name of the i5/OS command you want to call.
Enter Command Parameters	Enter the parameters you want to pass to the command.
Enter Optional Description	Enter comments to clarify the use of the command.



Note • See the *i5/OS Control Language Programming Guide* for more information on i5/OS commands and parameters.

System i (i5/OS) Find Component in RAIR

The System i (i5/OS) Find Component in RAIR action queries RAIR to find components in registry by their Product Name, Component Name, Component Version, and Feature Name (the first feature of the component).

This action then sets two variables to record the results of the query (\$RAIR_COMPONENT_COUNT\$ and \$RAIR_INSTANCE_VALUES\$, by default).

Table 7-151 • System i (i5/OS) Find Component in RAIR Options

Option	Description
Comment	Enter the text the installer shows when this action runs.
Find component	<p>Use these controls to define the comparisons that comprise the query:</p> <ul style="list-style-type: none"> • Compare product name—Enables or disables a comparison of the Product Name with the value you enter in the associated text box. • Compare component name—Enables or disables a query for the Component Name based on the value you enter in the associated text box. • Compare component version—Enables or disables a query for the Component Version based on the values you provide in the associated combination box and text box. Versions are conventionally represented in the following format: [Major] . [Minor] . [Revision] . [Subrevision]. For example: 1.0.2.1047. • Compare feature name—Enables or disables a comparison between the component feature (the first feature of the component) and the value you enter in the associated text box.
Set variables	<p>Use these controls to record query results for later access:</p> <ul style="list-style-type: none"> • Components found count—Identifies the InstallAnywhere variable that records the number of components that match the comparison criteria. The default value is \$RAIR_COMPONENT_COUNT\$. • Components instance values—Identifies the InstallAnywhere variable that records the locations of the components that match the comparison criteria. The instances result contains a comma-separated list of all the instance values (locations) of all the components registered. The default value is \$RAIR_INSTANCE_VALUES\$.

System i (i5/OS) Install File

The System i (i5/OS) Install File action enables files to be installed in the Integrated File System (IFS) on a remote i5/OS system. At build time, the IFS file is added to the installer. At install time, the IFS file is restored on the target system. This action includes the following options:

Table 7-152 • System i (i5/OS) Install File Options

Option	Description
Original	Identify the original file by selecting one of the following options: <ul style="list-style-type: none"> Source File—After selecting this option, click Choose File, navigate to the file you want to install, and click Select. The path and file name for the Merge Module will then appear in the Source file field. Existing File—After selecting this option, specify the IFS file and the path of that file on the target system.
Path	Specify the destination by entering in this text box the location you want to store the file on the target system. By default, the path is set to the <code>InstallAnywhere</code> variable: <code>\$OS400_INSTALL_DIR\$\$/\$</code>
Rename to	Select to rename the file that results on the target system.
Do not uninstall	Select to prevent the uninstaller from removing this file from the target system.
If the file already exists on the end user's system	Use this option to define the installer behavior when it attempts to install a file that already exists on the target system. Select one of the following options: <ul style="list-style-type: none"> Use project default Always overwrite Never overwrite Overwrite if older, do not install if newer Overwrite if older, prompt if newer Prompt if older, do not install if newer Always prompt user
Text Conversion	Use this option to enable or disable text conversion (based on Coded Character Set Identifiers) for the file you want to install. <ul style="list-style-type: none"> Source CCSID—Enter the Coded Character Set Identifier for the source file. This number identifies the existing encoding of the file on the source system. Target CCSID—Enter the Coded Character Set Identifier for the target file. This number identifies the encoding to which the installer converts the file on the target system. For example, use the default values 819 (source) and 37 (target) to convert from ASCII to EBCDIC encoding.

System i (i5/OS) Integrated File System (IFS)

Use the System i (i5/OS) Integrated File System (IFS) action to enable directories to be installed in the Integrated File System (IFS) on a remote i5/OS system. At build time, the Save (SAV) command is called to save the IFS directory from the i5/OS build system. At install time, the Restore (RST) command is called to restore the directory on the target system.

Table 7-153 • System i (i5/OS) Integrated File System (IFS) Options

Option	Description
IFS Directory Name	Specify the IFS source directory.
Include Sub Tree(s)	Specify parameters to control the inclusion or exclusion of sub trees. These parameters are used when the SAV command is called. Choose one of the following options: <ul style="list-style-type: none"> • *ALL • *DIR • *NONE • *OBJ • *STG
Destination	Specify the location of the directory on the target system. Type the location you want to store the file on the target system. For example, you can use the InstallAnywhere variable: \$OS400_INSTALL_DIR\$
Additional Save Parameters	Enter additional parameters to pass to the SAV command at build time.
Additional Restore Parameters	Enter additional parameters to pass to the RST command at install time.



Note • See the *i5/OS Control Language Programming Guide* for more information on i5/OS commands and parameters.

System i (i5/OS) Library

Enables libraries to be restored on an i5/OS system. At build time, the Save Library (SAVLIB) command is called to save the library from the i5/OS build system. At install time, the Restore Library (RSTLIB) command is called to restore the library to the target system.

Table 7-154 • System i (i5/OS) Library Options

Option	Description
Source Library Name	Enter the name of the library available on the i5/OS build system.
Destination Library Name	Enter the name of the library on the target system. This setting specifies whether the library contents are restored to the same library from which they were saved or to a different library.
Target Release	Specify the minimum supported release level of the operating system on which you intend to restore the library. Choose one of the following options: <ul style="list-style-type: none"> • *CURRENT • *PRV • Other—Enter a six-character release value in the format VxRxMx, where x is a digit (0-9).
Additional Save Parameters	Enter additional parameters to pass to the SAVLIB command at build time.
Additional Restore Parameters	Enter additional parameters to pass to the RSTLIB command at install time.



Note • See the *i5/OS Control Language Programming Guide* for more information on i5/OS commands and parameters.

System i (i5/OS) Licensed Program

The System i (i5/OS) Licensed Program action enables licensed programs to be restored on an i5/OS system. At build time, the Save Licensed Program (SAVLICPGM) command is called to save the licensed program on the i5/OS build system. At install time, the Restore Licensed Program (RSTLICPGM) command is called to restore the licensed program to the target system.

Table 7-155 • System i (i5/OS) Licensed Program Options

Option	Description
Licensed Program Identifier	Specifies the seven-character identifier of the licensed program that is saved. In i5/OS, the Display Software Resources (DSPSFWRSC) command can be used to find the identifiers for installed licensed programs.
Language	Specifies the native language version (NLV) used for the save operation. Specify one of the following options: <ul style="list-style-type: none"> *PRIMARY—Select to user the system's primary language. *ALL—Select to use all languages that the licensed program supports. Other—Select and specify a four-character NLV identifier. In i5/OS, the Display Software Resources (DSPSFWRSC) command can be used to find the available NLVs for a licensed program.  <p>Note • This parameter is ignored if the Object Type is *PGM.</p>
Release Level	Specifies the version, release, and modification level (VRM) of the licensed program to save. <ul style="list-style-type: none"> *ONLY—Use if only one VRM is installed for the licensed program. Other—Select and specify the release level in the format VxRxMx, where x is a digit (0-9). In i5/OS, the Display Software Resources (DSPSFWRSC) command can be used to find the installed release levels for a licensed program.
Object Type	Specifies which objects are saved. <ul style="list-style-type: none"> *ALL—Use to save both the program and the language objects. *PGM—Use to save only the program objects. *LNG—Use to save only the language objects.
Option	Specifies the licensed program options to save. <ul style="list-style-type: none"> Enter *BASE to save the base part of the licensed program. Specify the option number to save (01-99). In i5/OS, the Display Software Resources (DSPSFWRSC) command can be used to find the available options of a licensed program.

Table 7-155 • System i (i5/OS) Licensed Program Options

Option	Description
Target Release	Specifies the minimum supported release level of the operating system on which you intend to restore and use the licensed program. Select one of the following options: <ul style="list-style-type: none"> • *CURRENT • *PRV • Other—Enter a six-character release value in VxRxMx format (where x is a digit 0-9).
Re-save on Each Build	Choose one of the following options: <ul style="list-style-type: none"> • Yes—The licensed program is re-saved and transferred during the build. Choose this option if the licensed program has been changed and repackaged since the last build. • No—The build process uses the last copy of the licensed program that it saved and transferred to the local system (if one exists).
Additional Save Parameters	Enter additional parameters to pass to the SAVLICPGM command at build time.
Additional Restore Parameters	Enter additional parameters to pass to the RSTLICPGM command at install time.



Note • See the *i5/OS Control Language Programming Guide* for more information on *i5/OS commands and parameters*.

System i (i5/OS) Object

The System i (i5/OS) Object action enables one or more objects in the same library to be restored to an i5/OS system. At build time, the Save Objects (SAVOBJ) command is called to save the objects from the i5/OS build system. At install time, the Restore Objects (RSTOBJ) command is called to restore the objects to the target system.

Table 7-156 • System i (i5/OS) Object Options

Option	Description
Source Object Names	<p>Enter the names of the objects to save from the source library. Both generic names and specific names can be used.</p> <ul style="list-style-type: none"> Generic names are specified with an asterisk (for example, A*). The default value is *ALL. In i5/OS, the Display Library (DSPLIB) command can be used to find the available objects in a library.
Object Type	<p>Enter the types of objects to save.</p> <ul style="list-style-type: none"> Use *ALL to save all object types. This is the default value. Or you can enter the specific object type you want to save. In i5/OS, the Display Library (DSPLIB) command can be used to find the available object types in a library.
Source Library Name	<p>Enter the name of the library from which the objects are saved on the i5/OS build system.</p> <ul style="list-style-type: none"> Both generic names and specific names can be used. Generic names are specified with an asterisk (for example, A*). In i5/OS, the Display Library (DSPLIB) command can be used to find the available objects in a library.
Destination Library Name	<p>Enter the name of the library to which the objects are restored on the target system. Use *SAVLIB to restore objects to the same library. This is the default value.</p>
Target Release	<p>Specify the minimum supported release level of the operating system on which you intend to restore the objects. Choose one of the following options:</p> <ul style="list-style-type: none"> *CURRENT *PRV Other—Enter a six-character release value in the format VxRxMx, where x is a digit (0-9)).
Additional Save Parameters	<p>Enter additional parameters to pass to the SAVLIB command at build time.</p>

Table 7-156 • System i (i5/OS) Object Options

Option	Description
Additional Restore Parameters	Enter additional parameters to pass to the RSTLIB command at install time.



Note • See the *i5/OS Control Language Programming Guide* for more information on *i5/OS commands and parameters*.

System i (i5/OS) Process Remote Install (Merge Modules)

The **System i (i5/OS) Process Remote Install (Merge Modules)** action installs an InstallAnywhere Merge Module on the target system. The System i (i5/OS) Process Remote Install (Merge Modules) action customizer includes the following options:

Table 7-157 • System i (i5/OS) Process Remote Install (Merge Modules) Options

Option	Description
Source	Click Choose Merge Module , navigate to the Merge Module you want to include, and click Open . The path and file name for the Merge Module appears in the Source field.
Execute Pre-Install Actions	Select this option to run the Merge Module's Pre-Install actions.
Execute Post-Install Actions	Select this option to run the Merge Module's Post-Install actions.
Inherit parent install folder	Select this option to set the Merge Module to use the same value for the install folder that the parent installer is using. \$OS400_INSTALL_DIR\$ should never be advertised (or passed from parent installer to Merge Module) when the Merge Module is running on the i5/OS machine. If the Merge Module is run in Windows with the parent installer, then it is valid to pass \$OS400_INSTALL_DIR\$ as a parameter, but InstallAnywhere never passes that value automatically.
Show progress dialog	Select this option to show an indeterminate progress bar during the Merge Module install.
InstallAnywhere Variables to be passed between this installer and the Merge Module	Click Edit Variables to open the Edit Advertised Variables dialog box. See Edit Advertised Variables Dialog Box for more information.



Note • It is important to use the **Get System i (i5/OS) Login Credentials** panel in conjunction with this action; otherwise, the installation may fail due to an authentication failure.



Tip • For remote installations it may be important to prevent the local installer from altering the product registry on the local machine. To leave the product registry unchanged, remember to click **Do not update the product registry** in the **Project > Info** task.

System i (i5/OS) Program

The System i (i5/OS) Program action saves an i5/OS program (*PGM) object from the i5/OS build system and then restores and calls the program during the installation. The program is saved using the Save Object (SAVOBJ) command. At install time, the program is restored into the QTEMP library by the Restore Object (RSTOBJ) command, and then called by the CALL command.

Table 7-158 • System i (i5/OS) Program Options

Option	Description
Program Name	Enter the name of the *PGM object to save.
Program Development Library	Enter the library that contains the *PGM object.
Parameter List	Enter the parameters to pass to the CALL command.
Target Release	<p>Specify the minimum supported release level of the operating system on which you intend to restore and call the program. Choose one of the following options:</p> <ul style="list-style-type: none"> • *CURRENT • *PRV • Other—Enter a six-character release value in the format VxRxMx, where x is a digit (0-9).

System i (i5/OS) Program Temporary Fix (PTF)

The System i (i5/OS) Program Temporary Fix (PTF) action saves an i5/OS program temporary fix (PTF) from the i5/OS build system and then loads and applies the PTF during the installation. At build time, this action copies (CPYPTF) and saves (SAVLIB) the PTF. At install time, this action restores the PTF library on the target machine, loads (LODPTF) the PTF, and applies it (APYPTF).

Table 7-159 • System i (i5/OS) Program Temporary Fix (PTF) Options

Option	Description
Licensed Program Identifier	Enter the seven-character identifier of the product for which the PTFs are saved.
PTF ID(s)	Specify which PTFs are saved. Each PTF ID value must be seven-characters or less.
Release Level	Specify the release level of the licensed program for which the PTFs are saved. Enter one of the following: <ul style="list-style-type: none"> Enter *ONLY when only one release of the program is installed or supported. This is the default value. Otherwise, specify the release level in the format VxRxMx, where x is a digit (0-9).
Delay Apply	Specify whether immediate PTFs are applied at install time or whether immediate and delayed PTFs are applied during the next unattended IPL. Select one of the following options: <ul style="list-style-type: none"> *NO—Select to apply all immediate PTFs at install time and ignore delayed PTFs. *YES—Select to apply all PTFs during the next unattended IPL. *IMMDLY—Select to apply all immediate PTFs at install time and all delayed PTFs during the next unattended IPL.
Additional Save Parameters	Enter additional parameters to pass to the SAVLIB command at build time.
Additional Restore Parameters	Enter additional parameters to pass to the RSTLIB command at install time.
Target Release	Specify the minimum supported release level of the operating system on which you intend to restore and apply the PTFs. Select one of the following options: <ul style="list-style-type: none"> *CURRENT *PRV Other—Enter a six-character release value in the format VxRxMx, where x is a digit (0-9).



Note • See the *i5/OS Control Language Programming Guide* for more information on *i5/OS commands and parameters*.

Plug-In Actions

Developers can register custom code as plug-ins with the InstallAnywhere Advanced Designer. This feature allows properly packaged custom code to be integrated into the design environment, where it will appear on the **Choose an Action** dialog box under the **Plug-Ins** tab. Plug-ins are stored within the <*InstallAnywhere*>/plugins folder. The advantages of packaging custom code are:

- **Can be added as regular actions**—The developer can add them as regular actions without having to specify the JAR and class.
- **Utilizes InstallAnywhere variables**—Custom code usually utilizes InstallAnywhere variables for parameters and return values. If a developer wants to execute custom code multiple times in a project, it often means the global scope of InstallAnywhere variables forces the developer to be very careful with their parameters and return values. Plug-ins have a local scope for parameters.
- **Easily portable**—Plug-ins are easily portable across development teams.



Note • *InstallAnywhere Support is interested in distributing plug-ins developed by our users. If you have written a plug-in that you think would be useful to other developers, and you would like to share it, please contact the InstallAnywhere Support team.*

The following three plug-ins are installed with InstallAnywhere:

- [ExecuteAsRoot](#)
- [ExtractToFile](#)
- [PropertiesFileReader](#)

ExecuteAsRoot

The **ExecuteAsRoot** custom code plug-in action enables you to run a command on a Unix system as root.

Classname

com.zerog.ia.customcode.unix.ExecuteAsRoot

Instructions

The **ExecuteAsRoot** plug-in action requires that Expect, a public domain software tool for automating interactive applications, already be installed on the target machine, and also requires that your 'su' command support the -c switch. Expect can be downloaded from <http://expect.nist.gov>.

InstallAnywhere Input Variables

The following input variables are used with the **ExecuteAsRoot** plugin action:

Table 7-160 • InstallAnywhere Input Variables Used with the ExecuteAsRoot Plug-In

Variable	Description
\$EXECUTE_AS_ROOT_COMMAND\$	The command to run with root privileges.
\$EXECUTE_AS_ROOT_PASSWORD\$	The root password.
\$EXECUTE_AS_ROOT_EXPECT_PATH\$	(Optional) The location of the expect binary on the system. The default value is /usr/bin/expect.
\$EXECUTE_AS_ROOT_WAIT_TIME\$	(Optional) The time out for the command that is running. The default value is 5 seconds.
\$EXECUTE_AS_ROOT_TMP_PATH\$	(Optional) The location to store the temporary expect script. The default value is /tmp.

InstallAnywhere Output Variables

The following output variables are used with the **ExecuteAsRoot** plugin action:

Table 7-161 • InstallAnywhere Output Variables Used with the ExecuteAsRoot Plug-In

Variable	Description
\$EXECUTE_AS_ROOT_EXIT_CODE\$	The exit code, where 0 indicates success and 100 indicates incorrect password.
\$EXECUTE_AS_ROOT_ERROR_MESSAGE\$	Description of any problems that occurred.

ExtractToFile

The **ExtractToFile** custom code plug-in action enables you to open a URL resource and save the content to a new file. The URL resource can be inside the installer archive, on the user's hard drive, or on a network location. This plug-in is useful for installing files from a remote or Internet location or for extracting files from the installer archive for special processing.

Classname

com.zerog.ia.customcode.util.fileutils.ExtractToFile

Instructions

For the **ExtractToFile** plug-in action to work correctly, you need to set the ExtractToFile_Source and ExtractToFile_Destination variables to the expression you want to evaluate.

InstallAnywhere Input Variables

The following input variables are used with the **ExtractToFile** plugin action:

Table 7-162 • InstallAnywhere Input Variables Used with the ExtractToFile Plug-In

Variable	Description
ExtractToFile_Source	Path and name of the file as stored in ExtractToFileCustomCode.jar. For example: <code>com/acme/myfile.txt</code>
ExtractToFile_Destination	Path and name of new file to create. For example: <code>\$USER_INSTALL_DIR\$/myfile.txt</code>

InstallAnywhere Output Variables

None.

PropertiesFileReader

The **PropertiesFileReader** custom code plug-in action enables you to turn properties from a properties file into InstallAnywhere variables.

Classname

`com.zerog.ia.customcode.action.PropertiesFileReader`

InstallAnywhere Input Variables

The following input variables are used with the **PropertiesFileReader** plugin action:

Table 7-163 • InstallAnywhere Input Variables Used with the PropertiesFileReader Plug-In

Variable	Description
<code>\$PROPERTIES_FILE_LOCATION\$</code>	The location of the properties file.
<code>\$SUBSTITUTE_VARIABLES\$</code>	(Optional) Set to <code>TRUE</code> if you want to substitute InstallAnywhere variables found in values. Set to <code>FALSE</code> if you do not want to substitute. <code>TRUE</code> is the default value.
<code>\$PROPERTIES_TO_IGNORE\$</code>	(Optional) Set to a comma-separated list of properties in the properties file to ignore. For example: <code>USER_INSTALL_DIR,MY_PROP_1</code>
<code>\$OVERRIDE_EXISTING_VARIABLES\$</code>	(Optional) Set to <code>TRUE</code> if you want to override existing variables with properties from the properties file. Set to <code>FALSE</code> if you do not want to override existing variables.

InstallAnywhere Output Variables

None.

Common Action Properties

Many InstallAnywhere action customizers include the same controls. Information on these common settings and controls is grouped into the following sections.

Table 7-164 • Common Action Properties

Section	Description
Common Customizer Settings	Describes some of the most common controls and settings in InstallAnywhere action customizers.
Panel Action Settings	Explains the features common to most panel action customizers.
Get User Input Panels	Provides details about the controls and settings on the Get User Input - Simple and Get User Input - Advanced panel action customizers.

Common Customizer Settings

The following list contains common properties found in action customizers in InstallAnywhere.

Table 7-165 • Common Properties

Property	Description
Comment	Sets the name of the action in the visual tree
Do not uninstall	Tells an action to not attempt to undo the results of the action at uninstall time.
If file already exists on end user's system	Overrides the default behavior for how to resolve conflicts between installed files and pre-existing files
In Classpath	Puts the item on the classpath for all LaunchAnywhere executables installed
Installed File/Existing File	Determines whether the file is being installed, or already exists on the end user's system
Override default Unix /Mac OS X permissions	Sets the file permissions to a specific value for this action
Path	Shows the path where the action will be installed
Show Indeterminate Dialog	Brings up an indeterminate progress bar to show progress to the end user while a external process is executing

Table 7-165 • Common Properties (cont.)

Property	Description
Source	Shows the path where the item currently exists on the developer's system (displays the source path if source paths are being used)
Store process's exit code in	Sets the value of the InstallAnywhere variable to the process's exit code
Store process's stderr in	Sets the value of the InstallAnywhere variable to the process's standard error
Store process's stdout in	Sets the value of the InstallAnywhere variable to the process's standard out
Suspend installation until process completes	Pauses the installer until the launched process completes

Panel Action Settings

Panel actions (commonly called panels) are the means for requesting user input through a graphical interface.

Graphic installers may show the installation steps through a set of labels—words which represent the step—or by displaying specific images for the steps. Whether an installer displays labels or images is determined by the **Type of Additions to Installer Panels** setting on the **Installer UI > Look & Feel** subtask:

- **Images**—When **Images** is selected, the customizer for the panel in the **Pre-Install** and **Post-Install** task enables the use of the **Image Settings** tab. See [Image Settings](#).
- **List of installer steps**—If **List of Installer Steps** is selected, the **Label Settings** tab is enabled. See [Label Settings](#).



Note • These settings are unavailable to panel actions in the Uninstaller. Panel actions in the Uninstaller use the default values set in the **Installer UI > Look & Feel** task.

Image Settings

Use the **Image Settings** tab to specify what image your installers show on this panel. You may choose to use the default panel image, display an image specific to that panel, or display no image at all. Choose one of the following options:

Table 7-166 • Image Settings Tab

Option	Description
Use the Installer's Default Image	Select this option to use the image specified in Default Installer/Uninstaller Panel Image on the Installer Steps tab.

Table 7-166 • Image Settings Tab

Option	Description
Use the Same Image as the Previous Panel	Select this option to use the image specified for the panel immediately prior to this panel.
Do not Display an Image	Select this option to show no additional images in this panel.
Specify an Image (170x305)	<p>Select this option to use an image file you provide.</p>  <p>To specify an image:</p> <ol style="list-style-type: none"> 1. Click Choose. 2. In the Select an Image File dialog box, navigate to the image you want to use and click Open. 3. To verify the image appearance, click Preview.  <p>Note • The size of the install progress panel is 170 pixels wide by 305 pixels tall. Installer dimensions may change slightly by platform to better display text and different fonts.</p>

Label Settings

The **Label Settings** tab in the customizer enables developers to preview the labels and the icon images. The labels are highlighted, and marked as the installation progresses. The installer build process will auto populate the list based on the panel titles.

Table 7-167 • Label Settings Tab

Option	Description
Highlight the Same Label as the Previous Panel	Select this option to keep the label the same as the previous panel during the install.
Choose a Label from the List of Install Step Labels	Select this option to select a label from the list to display during the installation. To verify the label appearance, click Preview .



Note • Using the **Installer UI > Look & Feel** task's **Installer Steps** tab and the **Label Settings** tab found on each individual panel's customizer, developers can assign multiple panels to the same label. Thus, if there are numerous steps, or if the installer has several panels for the same step the interface can be adjusted as needed.



Note • To control label order, or to edit the content of the label, in the **Installer UI > Look & Feel** task's **Installer Steps** tab use the arrows and other control buttons found to the left of the list of panels.

Help Settings

On the **Help Settings** tab, you can specify customized help for this action.



Note • The **Help Settings** tab is only enabled if the **Enable installer help** and the **Use different help text for each panel** options are selected on the **Installer UI > Help** subtask. See [Enabling Installer Help](#).

Enter a **Title** for the help, which will appear in the help dialog box title bar, and then enter the content of the help in the **Help Text** box. Click **Preview** to preview how the help will be displayed in the pop-up help dialog box.

Enabling Installer Help

You enable installer help on the **Installer UI > Help** subtask. You can choose to use the same text for all help panels or customized help text for each panel. To enable installer help, perform the following steps:

**Task:****To enable installer help:**

1. In the Advanced Designer, open the **Installer UI > Help** subtask.
2. Select the **Enable installer help** option.
3. Under **Help Text Format**, select either **HTML** or **Plain text**. Selecting **HTML** allows greater formatting control of the message using HTML formatting tags. For example:

HTML	Display
MyHelp <I> Information</I>	MyHelp Information

4. Select one of the following:
 - **Use different help text for each panel**—Select this option if you want to enter help text for each installer panel on the **Help Settings** tab of the action customizer for **Pre-Install** and **Post-Install** panels.
 - **Use the same help text for all panels**—Select this option if you want to define a single help message that will be used on all panels. If you select this option, enter a **Title** and the **Help Text**. If you select this option, the **Help Settings** tab for panel customizers will be disabled.

Get User Input Panels



Edition • The Get User Input panels are available only in InstallAnywhere Enterprise edition.

There are two versions of the Get User Input panel that exist as separate actions:

Action	Description
Get User Input - Simple	Allows developers to request input from the end user.
Get User Input - Advanced	Allows developers to get input from the user using multiple input types and setting multiple variables.

Get User Input - Simple

This panel provides a way for users to provide simple responses that you can use to make decisions during the install process. With the Get User Input - Simple panel, you provide a panel title, prompt, and one or more text fields, check boxes, radio buttons, popup menus, or lists. However, this panel only allows you to provide one type of control. (To provide a mix of different controls—such as text fields, check boxes, and popup menus—use the Get User Input - Advanced panel instead.)

Table 7-168 • Get User Input - Simple Panel Controls

Control	Description
Title	The title the installer will show for this panel.
Prompt	The message with which you can specify the type of information the installer requests and the purpose for or results of those responses.
Input Items	Shows the input items by their Label, Default Value, and Text Reading Order in a table.

Table 7-168 • Get User Input - Simple Panel Controls

Control	Description
Input Method	<p>Lists the types of input methods the Get User Input - Simple panel can render.</p> <ul style="list-style-type: none"> • Textfields—Shows one or more text boxes, accepts text typed into the field, and stores those strings in the results variable for this panel. • Checkboxes—Shows one or more check boxes, accepts user input (checked: True, unchecked: False), and stores those values in the results variable. • Radio buttons—Shows one or more radio buttons (option buttons), accepts user choices (mutually exclusive), and records those choices in the results variable. • Popup menu—Shows a popup list from which the end user can choose a single option. This choice is recorded in the results variable, and can be stored in a user-specified variable. • List—Shows options in a list from which a user can select one or more options. These choices are recorded in the results variable or a variable you specify in the Configure Input Items dialog box.
Configure	<p>Opens the Configure Input Items dialog box. The Configure Input Items dialog box provides controls to fully describe the contents of the Get User Input panel. In general, this dialog box allows you to:</p> <ul style="list-style-type: none"> • Add or remove controls of the type set in Input Method. • Define labels and default values for each control. • Specify the text orientation and text reading direction for each control. • Set, for all panel controls, the typeface, size, text color, and background color. <p>The specific settings available on the Configure Input Items dialog box depend on the current Input Method setting.</p>
Preview	Shows a preview of the panel as it is currently configured.
Results Variable	Names the results variable in which the user input is stored.

Get User Input - Advanced

This panel allows users to provide more complex input responses that you can use to guide the install process at run time. With the Get User Input - Advanced panel, you provide a panel title, prompt, and one or more input components. These can be text fields, choice groups, labels, and more.

Table 7-169 • Get User Input - Advanced Panel Controls

Control	Description
Title	The title the installer will show for this panel.
Prompt	The message with which you can specify the type of information the installer requests and the purpose for or results of those responses.
Components	Lists the components, by their type, number, and label, included on the current Get User Input - Advanced panel.
Add Textfield	Adds a Textfield component to the Components list.
Add Choice Group	Adds a Choice Group component to the Components list.
Add Label	Adds a Label component to the Components list.
Add File Chooser	Adds a File Chooser component to the Components list.
Move Up/Move Down	Moves the component selected in the Components table up or down one row.

Table 7-169 • Get User Input - Advanced Panel Controls

Control	Description
Configure Selection	<p>Opens a Configure dialog box specific to the component type selected in the Component Type list. These dialog boxes set the number, layout, orientation, color, font, default value, results variable for controls within the component.</p> <ul style="list-style-type: none"> • Configure Textfield Dialog Box—Allows you to provide optional captions and labels, set the label location and echo character (normal/shadowed), enter an optional default value, define the results variable, and set typeface, size, and colors for the caption, label, and text field. • Configure Choice Group Dialog Box—Allows you to choose from Checkbox, Radio Button, Popup Menu, and List control types; choose a horizontal or vertical orientation; provide an optional caption; define text (Bidi) orientation, and set typeface, size, and colors for the choice group caption and controls. This dialog box also includes a Configure Components table in which you add or remove the individual choice group controls and define their labels, defaults, results variables, text reading order, and subcomponents. (Subcomponents allow you to define sets of child controls dependent on the activation of their parent control. For example, a check box might activate a set of related text fields when clicked.) • Configure Label Dialog Box—Allows you to provide an optional caption, choose from Plain Label or Wrapping Label types, define the label text, and set typeface, size, and colors for both the caption and the label. • Configure File Chooser Dialog Box—Allows you to provide optional captions and labels, choose from File or Directory chooser types, set a default location, define the results variable, and set typeface, size, and colors for the File Chooser component.
Remove Selection	Deletes the currently selected component from the Components list.
Preview Panel	Shows a preview of the panel as it is currently configured.

Rules Reference

Rules can be configured and attached to nearly any element of the installer: an Action or Action Group, a Component or Feature, or the entire installer itself. Rules return either a true or false value depending on the state of the computer running the installer. For example, a **Check Platform** rule can be set to evaluate to true only if the installer is being run on Solaris. This rule can then be attached to a Solaris-specific binary file that has no business being installed on a Windows or Mac OS X platform.

When you add multiple rules, you can build complex rule expressions that use multiple logical operators—including AND (`&&`), OR (`||`), and NOT (`!`)—and precedence operators (parentheses) to express the relationship between two or more rules. By default, rules are joined by the AND operator. For more information, see [Building Complex Rule Expressions](#).



Tip • When a rule is attached to an action, the installer evaluates the rule before the action runs. Therefore, you cannot check the result or return value of an action with a rule you attach to that action. If you want to check the return value of an action, you must do so in a rule attached to a subsequent action.

Table 7-170 • Built-in Rules

Rule	Editions	Description
Check File/Folder Attributes Rule	E	This rule allows developers to check the attributes of a file or directory that already exists on the target system. Developers can check if the object exists, whether it is a file or a folder/directory, and whether it is readable or writable. The in-use/not in-use options are tested on Windows systems only.
Check If File/Folder Exists Rule	E S	This rule applies to individual file/folder install actions. The Check if File/Folder Exists rule checks if the file or folder to which it is attached already exists in the specified install location. Developers can decide to install if it does exist or does not exist at that location.
Check Platform Rule	E S	The Check Platform rule allows developers to specify action or files to be run/installed only on specific platforms. Developers can choose platforms from the default list or define custom platforms using character strings or regular expressions. The platform of the target machine is determined by the Java virtual machine and reported to the installer.
Check Running Mode Rule		The Check Running Mode rule enables you to customize the events that occur when an end user launches Maintenance Mode.

Chapter 7: Reference

Rules Reference

Table 7-170 • Built-in Rules (cont.)

Rule	Editions	Description
Check System Architecture Rule	E S	The Check System Architecture rule allows developers to specify action or files to be run/installed only on specific system architecture (32-bit or 64-bit). The system architecture is normally determined by the Java virtual machine and reported to the installer, but in some cases may be specified in a registry action.
Check User-Chosen Language Rule	E S	Developers can use Check User-Chosen Language to make installation decision based on the locale chosen by the end user at installation time.
Compare File Modification Timestamp Rule	E	This rule allows developers to compare the timestamp of an existing target file in order to make an overwrite decision.
Evaluate Custom Rule Rule	E	Custom rules built using the specifications outlined in the InstallAnywhere API can be tailored to fit the needs of the installation.
Compare InstallAnywhere Variables Numerically Rule		Use to compare two InstallAnywhere variables numerically or to compare an InstallAnywhere variable against a specific value.
Compare InstallAnywhere Variables Rule	E S	This rule allows developers to make a simple string comparison of any InstallAnywhere variable. Developers can check if a variable equals, does not equal, contains, or does not contain a value.
Match Regular Expression Rule	E	The Match Regular Expression rule allows developers to compare a string, or InstallAnywhere Variable to a regular expression of the developers choosing. Regular Expressions (regexp) are an industry-standard method of expressing a variable string. Developers can find considerable information on regular expressions, including archives of use full expressions, and web applications that can be used to verify the validity of the expression on the Web.
System i (i5/OS) Licensed Program Exists Condition Rule	E	This rule allows developers to determine if a licensed program, matched by the Licensed Program Identifier, Option, and Release Level, exists on the target i5/OS system. Any action to which this rule is assigned can be conditionally performed based on whether a matching licensed program is found or not found on the i5/OS system.

Table 7-170 • Built-in Rules (cont.)

Rule	Editions	Description
System i (i5/OS) Primary Language Install Condition Rule	E	This rule allows developers to compare one or more primary language settings with the primary language of a target i5/OS system. Any action to which this rule is assigned can be conditionally performed based on whether the primary language was matched on the target i5/OS system.
System i (i5/OS) Program Temporary Fix (PTF) Condition Rule	E	This rule allows developers to determine if a PTF, matched by the Licensed Program Identifier, PTF ID, Release Level, and PTF Status, exists on the target i5/OS system. Any action to which this rule is assigned can be conditionally performed based on whether a matching PTF is found or not found on the i5/OS system.

Check File/Folder Attributes Rule

The Check File/Folder Attributes rule verifies attributes of a given file or folder and allows the action to which it is applied to run only if the rule evaluates to true.

Table 7-171 • Controls on the Check File/Folder Attributes Rule Customizer

Control	Description
File/Folder Path	Specifies the file or folder to check. Type the path, file name, or path and file name for the file or folder you want to evaluate. (The File/Folder Path text box can include Magic Folders and other InstallAnywhere variables. The installer resolved these variables at install time.)
This Install Path	Inserts the install path for the currently selected file or folder action. (Only available when this rule is assigned to a file- or folder-related action in the Install task.)

Table 7-171 • Controls on the Check File/Folder Attributes Rule Customizer

Control	Description
Perform only if the file/folder ... <p>Enables or disables a particular criteria set. For example, if the Perform only if the file/folder is a control is not checked, the rule does not evaluate whether the current value of the File/Folder Path text box is a file or a folder.</p> <p>Already Exists Does Not Already Exist</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> Click Already Exists if you want the action to run only if the file or folder currently specified in the File/Folder Path text box already exists. Click Does Not Already Exist if you want the action to run only if the file or folder currently specified in the File/Folder Path text box does not exist. <p>File Folder</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> Click File if you want the action to run only if the file you specified in the File/Folder Path text box is, in fact, a file on the target system. Click Folder if you want the action to run only if the folder you specified in the File/Folder Path text box is, in fact, a folder on the target system. <p>Readable Writable Both</p> <p>Select one or both of the following options:</p> <ul style="list-style-type: none"> Click Readable to allow the action to run only if the file or folder you specified in the File/Folder Path text box is readable. Click Writable to allow the action to run only if the file or folder you specified in the File/Folder Path text box is writable. <p>In-Use Not In-Use</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> Click In-Use to allow the action to run only if the file or folder you specified in the File/Folder Path text box is in use at install time. Click Not In-Use to allow the action to run only if the file or folder you specified in the File/Folder Path text box is not in use at install time. 	

Check If File/Folder Exists Rule

The Check If File/Folder Exists rule checks to see if a file or folder to be installed already exists on the target computer.



Tip • This rule is ideal for avoiding accidental overwrites that might occur when your installer attempts to deploy a file that already exists on the target system.

Table 7-172 • Check If File/Folder Exists Rule Customizer

Option	Description
It does not already exist on the user's system	Sets the rule to prevent the install of a file that already exists on the target system.
It already exists on the user's system	Sets the rule to permit the install of a file that already exists on the target system.

Check Platform Rule

The Check Platform rule checks the platform (operating system) on which the installer is running.



Note • The Check Platform rule evaluates the platform of a target machine based on the text returned by the Java VM's `System.getProperty (os.name)` method call.

The Check Platform Rule customizer includes the following options:

Table 7-173 • Check Platform Rule Customizer

Option	Description
Do Not Perform On	<p>Lists platforms on which the action is not performed.</p> <p>The installer evaluates the platforms in this list at install-time before it evaluates the Perform On platforms. Therefore, to install on some Windows systems and not others, for example, remember to remove the Windows (All) platform expression from the Do Not Perform On. Otherwise, all Windows target systems will return a False result even if specific Windows platforms appear in the Perform On list.</p>
Perform On	<p>Lists platforms on which the action is permitted.</p>
-->	<p>Moves the platforms currently selected in the Do Not Perform On list to the Perform On list.</p>

Chapter 7: Reference

Rules Reference

Table 7-173 • Check Platform Rule Customizer

Option	Description
<--	Moves platforms currently selected in the Perform On list to the Do Not Perform On list.
More Platforms	Opens the Add Platform dialog box. This dialog lists some pre-set platforms you can add to the rule and also allows you to create custom platform expressions. Choose a platform from the list, enter a custom platform expression, or click This Platform ; then click OK .
Remove Platforms	Deletes the currently selected platforms from both the Do Not Perform On and the Perform On lists.

Check Running Mode Rule

As described in [About Maintenance Mode Action Groups](#), when you enable Maintenance Mode, **Check Running Mode** rules are automatically added to the Action Groups that are created in the **Pre-Install** and **Pre-Uninstall** tasks. However, you can also manually apply a Check Running Mode rule to any Action Group, Action, or Panel in your installation project to specify whether or not that element should be executed when Maintenance Mode is run. This enables you to customize the events that occur when an end user launches Maintenance Mode.



Important • When a Check Running Mode rule is assigned to an Action Group, it is applied to all panels and actions in that Action Group.

The Check Running Mode rule customizer includes the following options:

Table 7-174 • Check Running Mode Rule Customizer

Option	Description
Installation	Select this option to associate the selected element with the Installation phase of Maintenance Mode. This option is available in the following tasks: <ul style="list-style-type: none">• Pre-Install• Install• Post-Install For example, when an end user chooses to run Maintenance Mode to install an application, only those actions and panels with a Check Running Mode rule set to Installation will be executed.

Table 7-174 • Check Running Mode Rule Customizer

Option	Description
Add Features	<p>Select this option to associate the selected element with the Add Features phase of Maintenance Mode. This option is available in the following tasks:</p> <ul style="list-style-type: none"> • Pre-Install • Install • Post-Install <p>For example, when an end user chooses to run Maintenance Mode to add features, only those actions and panels with a Check Running Mode rule set to Add Features will be executed.</p>
Repair Installation	<p>Select this option to associate the selected element with the Repair Installation phase of Maintenance Mode. This option is available in the following tasks:</p> <ul style="list-style-type: none"> • Pre-Install • Install • Post-Install <p>For example, when an end user chooses to run Maintenance Mode to repair an installation, only those actions and panels with a Check Running Mode rule set to Repair Installation will be executed.</p>
Uninstallation	<p>Select this option to associate the selected element with the Uninstallation phase of Maintenance Mode. This option is available in the following tasks:</p> <ul style="list-style-type: none"> • Pre-Uninstall • Uninstall • Post-Uninstall <p>For example, when an end user chooses to run Maintenance Mode to uninstall an application, only those actions and panels with a Check Running Mode rule set to Uninstallation will be executed.</p>
Remove Features	<p>Select this option to associate the selected element with the Remove Features phase of Maintenance Mode. This option is available in the following tasks:</p> <ul style="list-style-type: none"> • Pre-Uninstall • Uninstall • Post-Uninstall <p>For example, when an end user chooses to run Maintenance Mode to remove a feature, only those actions and panels with a Check Running Mode rule set to Remove Features will be executed.</p>



Important • You should always add a Check Running Mode rule to the Uninstaller action for your product, and you should set that Check Running Mode rule to **Installation** so that it executes only once.



Note • If a panel in the **Pre-Install** task does not have a Check Running Mode rule assigned either to itself or any of its parents, then this panel will be run during Install, Maintenance Mode/Add Features, and Maintenance Mode/Repair Installation.



Note • If a panel in the **Pre-Uninstall** task does not have a Check Running Mode rule assigned either to itself or any of its parents, then this panel will be run during Uninstall and Maintenance Mode/Remove Features.

Evaluate Custom Rule Rule

Custom rules built using the specifications outlined in the InstallAnywhere API can be tailored to fit the needs of the installation.

The process of creating a custom rule is similar to creating a custom action. To create a rule, you create a custom class that extends `com.zerog.ia.api.pub.CustomCodeRule`, and this class must implement an `evaluateRule` method that returns true if the rule succeeds and false if the rule fails.

At run time, if the rule succeeds, the action associated with it will be installed or performed, and if the rule fails the action will be skipped.

The **Evaluate Custom Rule** customizer includes the following options:

Table 7-175 • Evaluate Custom Rule Customizer

Option	Description
Path	Click the Choose JAR or ZIP button and browse for the .jar or .zip file.
Class	<p>Enter the fully qualified custom rule class name (such as <code>com.acme.MyCustomCodeRule</code>).</p>  <p>Note • This class must extend <code>com.zerog.ia.api.pub.CustomCodeRule</code> and must implement a public boolean <code>evaluateRule()</code> method.</p>
Configure Dependencies	Click to open the Custom Rules Dependencies dialog box, where you can select a .JAR or .ZIP file which contains classes referenced by your custom rule so that those classes are included in the archive and are available to the rule at runtime.
Open Javadocs	Click to open the documentation for the InstallAnywhere API.

Evaluate Custom Rule Examples

The following are examples of rules that could be used in an **Evaluate Custom Rule** rule:

- [Example: Rule That Always Succeeds](#)
- [Example: Ensuring Minimum Target System Screen Resolution](#)

Example: Rule That Always Succeeds

The implementation of a rule that always succeeds would appear similar to the following:

```
import com.zerog.ia.api.pub.*;  
  
public class AlwaysSucceedsRule extends CustomCodeRule  
{  
    public boolean evaluateRule( )  
    {  
        return true; // always succeed  
    }  
}
```

You compile the rule class the same way you compile other custom code (by including `IAClasses.zip` in the compiler classpath), and package the `.class` file in a `.jar` file or `.zip` file.

Example: Ensuring Minimum Target System Screen Resolution

Suppose you want to ensure the target system screen resolution is above a given minimum. A custom code rule that succeeds if the target system's resolution is at least 1024 by 768 might appear as follows.

```
import com.zerog.ia.api.pub.*;  
import java.awt.*;  
  
public class CheckScreenDimensionsRule extends CustomCodeRule  
{  
    private static final int MIN_WIDTH = 1024;  
    private static final int MIN_HEIGHT = 768;  
  
    public boolean evaluateRule( )  
    {  
        Toolkit tk = Toolkit.getDefaultToolkit( );  
        Dimension d = tk.getScreenSize( );  
  
        // fail if either dimension is below minimum  
        if ((d.width < MIN_WIDTH) || (d.height < MIN_HEIGHT))  
            return false;  
        else  
            return true;  
    }  
}
```

You compile the class and package it in a `.jar` or `.zip` file.

After you build and run the installer, the action containing the custom code rule will be skipped if the target system does not meet the minimum screen resolution.

Proxy Variable ruleProxy

The CustomCodeRule class provides the proxy variable ruleProxy (of type CustomCodeRuleProxy). With ruleProxy you can obtain the values of InstallAnywhere variables with the same methods getVariable and substitute available to custom code actions. For example, a refinement for the previous rule might be to enable the rule to automatically succeed if the installer is running in silent mode or console mode, using the \$INSTALLER_UI\$ variable.

Compare InstallAnywhere Variables Numerically Rule

You can use the **Compare InstallAnywhere Variables Numerically** rule to compare two InstallAnywhere variables numerically or to compare an InstallAnywhere variable against a specific value.

When this rule is applied, it converts the content of the variables into numeric values and then compares them. Integer, floating point, long and double operations are supported in this rule. However if the value represented by the variable is not parseable as a numeric value, then the rule returns false.

The **Compare InstallAnywhere Variables Numerically** customizer includes the following options:

Table 7-176 • Compare InstallAnywhere Variables Numerically Customizer

Option	Description
Operand 1 Operand 2	Enter the two InstallAnywhere variables that you want to compare, or enter an InstallAnywhere variable as one operand and enter a specific value as the second operand. For the operands, you can enter either a variable or (such as \$MY_VARIABLE\$) or a literal, specific value (such as 4502).
[Operator]	From the list, select one of the following operators: <ul style="list-style-type: none"> • greater than or equal to (>=) • equal to (==) • less than (<) • less than or equal to (<=) • greater than (>) • greater than or equal to (>=)

Variables

InstallAnywhere variables include custom, user-defined variables, standard variables, LaunchAnywhere executable (LAX) properties, and the variables associated with Magic Folders.

Table 7-177 • InstallAnywhere Variables Reference

Type	Description
Standard InstallAnywhere Variables	Lists and describes standard variables employed by InstallAnywhere.
LAX Properties	Describes the LaunchAnywhere Executable properties.
Magic Folders	Describes the Magic Folders supported by InstallAnywhere.

Standard InstallAnywhere Variables

InstallAnywhere includes a number of default variables that store information essential to the installation process. The following variables are standard InstallAnywhere variables:

Table 7-178 • Standard InstallAnywhere Variables

Variable	Description	Status
\$;\$ or \$:\$	The \$;\$ and \$:\$ variables represent platform-specific path separators.	Read-only
\$/\$ or \$\\$	The \$/\$ and \$\\$ variables represent platform-specific directory separators, most useful to refer to paths in a platform-independent manner. These variables have the same value as the Java property file separator.	Read-only
\$CHOSEN_DIALOG_BUTTON\$	The \$CHOSEN_DIALOG_BUTTON\$ variable reflects the return value set by the end user's choice in the Show Message dialog box action. If, for example, the end user chooses button 1, this variable is set to 1.	

Chapter 7: Reference

Variables

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$CHOSEN_FEATURE_LIST\$	<p>The \$CHOSEN_FEATURE_LIST\$ variable is a comma-separated list of all the end-user-selected features (short names).</p> <p>You can alter the features that will be installed by altering the value of this variable; however, you must understand that the string value of this variable does not clearly represent feature dependencies or parent-child relationships. You can infer these relationships from the string value of \$CHOSEN_FEATURE_LIST\$ if you fully understand the relationships between the installer's features and the function of the Choose Install Set panel.</p> <p>Installers interpret the string value of this variable and discard values that do not make sense. Hence, if you add a feature that does not exist in the installer, that portion of the \$CHOSEN_FEATURE_LIST\$ is ignored. Likewise if you alter this string in a way that violates feature dependencies or parent-child relationships, you may get a different result when you read back the value of this variable.</p> <p></p> <p>Note • Use this variable, along with \$CHOSEN FEATURE_n\$ and \$CHOSEN FEATURE_NUM\$, to track the features your end users select during install and uninstall.</p> <p></p> <p>Note • A similar set of variables exists (\$CHOSEN_INSTALL_FEATURE_LIST\$, \$CHOSEN_INSTALL_FEATURE_NUM\$, \$CHOSEN_INSTALL_FEATURE_n\$, and \$FEATURE_UNINSTALL_LIST\$). However, the CHOSEN_FEATURE set of variables can be consistently used by the installer author at install and uninstall by specifying features' short names, which are not localizable and thus serve as consistent IDs no matter the locale in which the installer or uninstaller is running.</p>	

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$CHOSEN_FEATURE_n\$	<p>One \$CHOSEN_FEATURE_n\$ variable is created for each feature counted in the variable \$CHOSEN_FEATURE_NUM\$. These variables hold the short name of a feature to be installed.</p> <p>For example, if \$CHOSEN_FEATURE_NUM\$ equals 2, then two variables of this form are created: \$CHOSEN_FEATURE_1\$ and \$CHOSEN_FEATURE_2\$.</p>  <p>Note • Use this variable, along with \$CHOSEN_FEATURE_LIST\$ and \$CHOSEN_FEATURE_NUM\$, to track the features your end users select during install and uninstall.</p>  <p>Note • A similar set of variables exists (\$CHOSEN_INSTALL_FEATURE_LIST\$, \$CHOSEN_INSTALL_FEATURE_NUM\$, \$CHOSEN_INSTALL_FEATURE_n\$, and \$FEATURE_UNINSTALL_LIST\$). However, the CHOSEN_FEATURE set of variables can be consistently used by the installer author at install and uninstall by specifying features' short names, which are not localizable and thus serve as consistent IDs no matter the locale in which the installer or uninstaller is running.</p>	
\$CHOSEN_FEATURE_NUM\$	<p>This variable holds the total number of features (as a string) that the end user chooses to install.</p> <p>For example, if the end user chooses 3 features during the install, the value of CHOSEN_FEATURE_NUM is 3, and the short names of these features can be referenced in the variables \$CHOSEN_FEATURE_1\$, \$CHOSEN_FEATURE_2\$, and \$CHOSEN_FEATURE_3\$.</p>  <p>Note • Use this variable, along with \$CHOSEN_FEATURE_n\$ and \$CHOSEN_FEATURE_LIST\$, to track the features your end users select during install and uninstall.</p>  <p>Note • A similar set of variables exists (\$CHOSEN_INSTALL_FEATURE_LIST\$, \$CHOSEN_INSTALL_FEATURE_NUM\$, \$CHOSEN_INSTALL_FEATURE_n\$, and \$FEATURE_UNINSTALL_LIST\$). However, the CHOSEN_FEATURE set of variables can be consistently used by the installer author at install and uninstall by specifying features' short names, which are not localizable and thus serve as consistent IDs no matter the locale in which the installer or uninstaller is running.</p>	

Chapter 7: Reference

Variables

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$CHOSEN_INSTALL_BUNDLE_LIST\$	<p>This variable is a comma-separated list of all install features (short name) chosen for the installation.</p>  <p>Note • This variable is deprecated. Instead, use \$CHOSEN_INSTALL_FEATURE_LIST\$.</p>	Read-only
\$CHOSEN_INSTALL_BUNDLE_n\$	<p>If Choose Product Features is enabled, one instance of \$CHOSEN_INSTALL_BUNDLE_n\$ is created for each feature as given in the variable \$CHOSEN_INSTALL_BUNDLE_NUM\$. This holds the short name of a feature to be installed.</p> <p>For example, if \$CHOSEN_INSTALL_BUNDLE_NUM\$ equals 2, then two variables of this form are created: \$CHOSEN_INSTALL_BUNDLE_1\$ and \$CHOSEN_INSTALL_BUNDLE_2\$.</p>  <p>Note • This variable is deprecated. Instead, use \$CHOSEN_INSTALL_FEATURE_n\$.</p>	Read-only
\$CHOSEN_INSTALL_BUNDLE_NUM\$	<p>If Choose Product Features is enabled, this variable holds the total number of components (as a string) that the end user chooses to install.</p>  <p>Note • This variable is deprecated. Instead use \$CHOSEN_INSTALL_FEATURE_NUM\$.</p>	Read-only

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$CHOSEN_INSTALL_FEATURE_LIST\$	<p>This variable is a comma-separated list of all the 'selected' features (short names). You can alter the features that will be installed by altering the value of this variable; however, you must understand that the string value of this variable does not clearly represent feature dependencies or parent-child relationships.</p> <p>(You can infer these relationships from the string value of \$CHOSEN_INSTALL_FEATURE_LIST\$ if you fully understand the relationships between the installer's features and the function of the Choose Install Set panel.)</p> <p>In order to preserve the functional continuity of the graphical installer and any automation efforts that use InstallAnywhere variables, installers interpret the string value of this variable and discard values that do not make sense. Hence, if you add a feature that does not exist in the installer, that portion of the \$CHOSEN_INSTALL_FEATURE_LIST\$ is ignored. Likewise if you alter this string in a way that violates feature dependencies or parent-child relationships, you may get unexpected results.</p>	
\$CHOSEN_INSTALL_FEATURE_n\$	If Choose Product Features is enabled, one \$CHOSEN_INSTALL_FEATURE_n\$ variable is created for each feature as given in the variable \$CHOSEN_INSTALL_FEATURE_NUM\$. These variables hold the short name of a feature to be installed. For example, if \$CHOSEN_INSTALL_FEATURE_NUM\$ equals 2, then two variables of this form are created: \$CHOSEN_INSTALL_FEATURE_1\$ and \$CHOSEN_INSTALL_FEATURE_2\$	
\$CHOSEN_INSTALL_FEATURE_NUM\$	If Choose Product Features is enabled, this variable holds the total number of features (as a string) that the end user chooses to install.	
\$CHOSEN_INSTALL_SET\$	If Choose Product Features is enabled, this variable holds the short name of the install set chosen by the end user. If the end user chose to customize the install, this variable holds the string CUSTOM.	Set before install time
\$CMD_LINE_ARGUMENTS\$	A special InstallAnywhere variable resolved by the launcher and not by the installer. If this variable is in the LAX property lax.command.line.args, it resolves to the arguments sent to the LaunchAnywhere executable.	Used by LaunchAnywhere
\$COMMAS\$	This resolves to a comma (,).	Read-only

Chapter 7: Reference

Variables

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$componentname_DEPENDENCY_STATE_VARIABLE\$	If the dependency check passes this variable contains an empty string. If the dependency check fails the variable contains the Dependency Failed Message. Set at the beginning of the Install phase or after the Evaluate Dependencies action in Pre-Install .	
\$componentname_MATCHED_KEY_FILE\$	This variable contains the location of where the key file of the dependency is installed. If the dependency check has failed, this variable contains an empty string. Set at the beginning of the Install phase or after the Evaluate Dependencies action in Pre-Install .	
\$DEPENDENCY_FAILURES\$	This is a comma separated list of Dependencies that have failed and are not installed. Set at the beginning of the Install phase or after the Evaluate Dependencies action in Pre-Install .	
\$DEPENDENCY_REPORT\$	This variable contains a report of all the dependencies that have failed. Set at the beginning of the Install phase or after the Evaluate Dependencies action in Pre-Install .	
\$DEPENDENCY_STATUS\$	Returns success or failure, depending on the entire dependency evaluation. If any of the dependencies in the installer fail, this variable is set to failure. Set at the beginning of the Install phase or after the Evaluate Dependencies action in Pre-Install .	
\$DEPENDENCY_SUCCESSES\$	This is a comma separated list of Dependencies that have passed and are installed. Set at the beginning of the Install phase or after the Evaluate Dependencies action in Pre-Install .	
\$DEVELOPER_DISK_SPACE_ADDITIONAL\$	This variable specifies an arbitrary additional value, as a string representing the additional bytes, that the Disk Space Check action then adds to the computed required disk space for the installation. By default this variable has a value of zero.	Developers may set
\$DOLLAR\$	This resolves to \$.	Read-only

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$EMPTY_STRING\$	This variable represents a value of null. This makes \$EMPTY_STRING\$ useful to determine whether or not any variables have been initialized. Variables that have not yet been initialized will have this as their value.	Read-only
\$EXTRACTOR_DIR\$	A full path to the directory containing the self-extractor executable (from where it was launched).	Read-only
\$EXTRACTOR_EXECUTABLE\$	A full path to the self-extractor executable (from where it was launched).	Read-only
\$FEATURE_UNINSTALL_LIST\$	The \$FEATURE_UNINSTALL_LIST\$ a comma-delimited list of features (display name, not short name) the user chose to uninstall.	
\$FREE_DISK_SPACE_BYTES\$ \$FREE_DISK_SPACE_KILOBYTES\$ \$FREE_DISK_SPACE_MEGABYTES\$ \$FREE_DISK_SPACE_GIGABYTES\$	The free disk space available on the destination install volume, as a string representing the free bytes, as determined by the Disk Space Check action. The variable gains its value immediately before the installation of any files or folder listed in the Install task.	Read-only
\$IA_BROWSER_FOLDERS\$	Controls the use of Swing or Native resources for rendering the Browse for Folder dialog box. For more information, see \$IA_BROWSER_FOLDERS\$ Variable .	
\$IA_CLASSPATH\$	The classpath as specified in the InstallAnywhere Advanced Designer environment.	Developers may set in the Advanced Designer
\$IA_INSTALL_LOG\$	Setting this variable generates an XML-formatted installation log in the \$USER_INSTALL_DIR\$ location. The log details the installation along with warnings and errors.	Read-only
\$IA_MAINTENANCE_MODE\$	This variable identifies whether the Uninstaller is running in Maintenance Mode: <ul style="list-style-type: none"> • TRUE—Uninstaller is running in Maintenance Mode. • FALSE or empty—Uninstaller is running in standard mode. 	

Chapter 7: Reference

Variables

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$IA_MAINTENANCE_OPTION\$	If the Uninstaller is running in Maintenance Mode (\$IA_MAINTENANCE_MODE\$=TRUE), this variable identifies which mode was selected: <ul style="list-style-type: none"> • ADD—Add Features mode. • REMOVE—Remove Features mode. • REPAIR—Repair Features mode. • UNINSTALL—Uninstall mode. 	
\$IA_ROLLBACK\$	Used when the Installer starts to roll back an installation either on cancel or when encountering a fatal error. Is set to TRUE when a rollback is triggered, or when CustomCode throws a FatalInstallException. Otherwise, it is set to FALSE or is empty.	
\$INSTALL_LOG_DESTINATIONS\$	As the creation of the install log is the last action of an installation, this variable can be set anytime during Pre-Install, Install, or Post-Install. The end user input can choose the installation log location.	Developers may set
\$INSTALL_LOG_NAME\$	This variable sets the name of the installation log. If not set, the installation log name is based on the product name.	Developers may set
\$INSTALL_SUCCESS\$	This variable alerts the end user if the installation was successfully completed or if it failed. There are four possible values for this variable: SUCCESS, WARNING, NONFATAL_ERROR, and FATAL_ERROR.	Read-only
\$INSTALLER_LAUNCH_DIR\$	This is a full path to the installer's self-extractor.	Read-only
\$INSTALLER_LOCALE\$	The locale as a string (see <code>java.util.Locale.toString()</code>) that the end user selected at the beginning of the installation.	Read-only
\$INSTALLER_TITLE\$	This is the title of the installer displayed in the title bar.	Developers may set

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$INSTALLER_UI\$	<p>This resolves to the UI mode for the installer.</p>  <p>Note • The \$INSTALLER_UI\$ variable is case sensitive when used in a rule, but is not case sensitive when used in a response file:</p> <ul style="list-style-type: none"> • Rule—When using the \$INSTALLER_UI\$ variable in a rule, the second operand in the comparison, between the \$INSTALLER_UI\$ variable and a valid value, is case sensitive. In this scenario, \$INSTALLER_UI\$=SILENT would evaluate correctly, but \$INSTALLER_UI\$=silent would not. • Response file—However, when using the \$INSTALLER_UI\$ variable in a response file and passing the response file to the installer, setting \$INSTALLER_UI\$=silent would evaluate correctly. 	Read-only, but developers may set it at the start.
\$JAVA_DOT_HOME\$	This is what the Java property java.home reports.	Read-only
\$JAVA_EXECUTABLE\$	This is the path to the chosen Java executable.	Read-only
\$JAVA_HOME\$	This is the root of the Java installation.	Read-only
\$JDK_HOME\$	This is the path to the root of a JDK installation. This variable is set only if the chosen VM is a JDK. If it is not a JDK, then this variable is blank.	Read-only
\$!ax.nl.env.ENVIRONMENT_VARIABLE_NAME\$	 <p>Note • Windows and UNIX only.</p> <p>Access any system environment variable (for example, access PATH via \$!ax.nl.env.PATH\$) by specifying the variable name as an all upper case string. These variables are resolved at application runtime, when LaunchAnywhere executes. Developers can also access system environment variables via LaunchAnywhere properties.</p>	Read-only
\$!ax.nl.env.exact_case.ENVIRONMENT_VARIABLE_NAME\$	 <p>Note • Windows and UNIX only.</p> <p>Access any system environment variable (for example, access Path via \$!ax.nl.env.exact_case.Path\$) by specifying the variable name as a string of the exact case as it is defined in the environment. Note that these variables are resolved at application runtime when LaunchAnywhere executes. Developers can also access system environment variables via LaunchAnywhere properties.</p>	Read-only

Chapter 7: Reference

Variables

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$NEVER_UNINSTALLS_VM\$	Tells the uninstaller to never uninstall a Java Virtual Machine. This is useful when multiple projects and uninstallers share 1 virtual machine. Set to TRUE to never uninstall a JVM.	
\$NULL\$	This is equivalent to \$EMPTY_STRING\$.	Read-only
\$PRODUCT_ID\$	Resolves to the value of the Product ID in the Project > Description task	Read-only
\$PRODUCT_NAME\$	This is the product name.	Developers may set
\$PRODUCT_VERSION_NUMBER\$	Resolves to the value of the Product Version in the Project > Description task	Read-only
\$PROMPT_USER_CHOSEN_OPTION\$	This variable reflects the return value set by the end user's choice in the Show Message Console dialog box action. If the end user chooses Option 1, this variable is set to 1.	Read-only
\$prop.JAVA_PROPERTY\$	Access any Java property through InstallAnywhere Variables. An example is \$prop.os.name\$ which returns the value of the os.name property.	Read-only
\$REGISTER_UNINSTALLER_WINDOWS\$	This variable tells the uninstaller if it should register itself with Windows Add/Remove Programs. Set to FALSE to not have the uninstaller register.	
\$REQUIRED_DISK_SPACE_BYTES\$ \$REQUIRED_DISK_SPACE_KILOBYTES\$ \$REQUIRED_DISK_SPACE_MEGABYTES\$ \$REQUIRED_DISK_SPACE_GIGABYTES\$	The disk space required by the installer, as a string representing the required bytes, as determined by the Disk Space Check action. The variable gains its value immediately before the installation of any files or folder listed in the Install task.	Read-only
\$RESTART_NEEDED\$	Tells the installer or uninstaller if the system needs to restart to complete the installation.	Read-only
\$SHORTCUT_NAME\$	This variable resolves to "Shortcut" on Win32 systems, "Alias" on Mac OS X systems, and "Link" on all other systems.	Read-only

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$SKIP_UNINSTALL\$	Tells the uninstaller whether or not to perform the uninstall step. To cause the uninstaller to skip the uninstall step, set \$SKIP_UNINSTALL\$ to true . To allow the uninstall step to occur, set \$SKIP_UNINSTALL\$ to false . (\$SKIP_UNINSTALL\$ is false by default.) If the installer skips the uninstall step, it sets \$UNINSTALL_SUCCESS\$ to SKIPPED .	
	 Note • The Uninstall Complete panel tests the value of \$UNINSTALL_SUCCESS\$ variable and provides controls for you to customize the message the panel shows in the event that the uninstall step was intentionally skipped.	
\$UNINSTALL_SUCCESS\$	The same as \$INSTALL_SUCCESS\$, but for the uninstaller. There are three possible values for this variable: SUCCESS , SKIPPED , or INCOMPLETE .	Read-only
\$UNINSTALLER_TITLE\$	This is the title of the uninstaller displayed in the title bar.	Developers may set
\$IA_INSTANCE_MANAGEMENT_OPTION\$	<p>This variable is set by the choice an end user makes on the Manage Instances dialog box (or equivalent console panel). Possible values include:</p> <ul style="list-style-type: none"> • NEW—Indicates that the user chose the Install a New Instance option. • MODIFY—Indicates that the user chose the Modify an Existing Instance option. • NOT_DEFINED (or empty)—Variable is not defined.  Note • This is a read only value and will be available for the installer author as a standard InstallAnywhere variable. This variable is not serialized into the <code>installvariables.properties</code> or the response files.	Read-Only

Table 7-178 • Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
\$IA_INSTANCE MODIFY PATH\$	<p>This variable is set when the end user specifies the path to an instance that they want to modify on the Manage Instances dialog box (or equivalent console panel). Possible values include:</p> <ul style="list-style-type: none"> • [Path to instance being modified]—When the user has chosen an instance to modify • NOT_DEFINED (or empty)—Variable is not defined.  <p>Note • This is a read only value and will be available for the installer author as a standard InstallAnywhere variable. This variable is not serialized into the <code>installvariables.properties</code> or the response files.</p>	Read-Only
\$IA_INSTALL_INSTANCE_NUM\$	<p>This variable reflects the number of the instance of the product that is being installed. Possible values include:</p> <ul style="list-style-type: none"> • [Number of the instance being installed]—The number of the instance of the product currently being installed • NOT_DEFINED (or empty)—Variable is not defined.  <p>Note • This is a read only value and will be available for the installer author as a standard InstallAnywhere variable. This variable is not serialized into the <code>installvariables.properties</code> or the response files.</p>	Read-Only

\$IA_BROWSE_FOLDERS\$ Variable

The \$IA_BROWSE_FOLDERS\$ variable controls whether InstallAnywhere installers will use **Swing** or **Native** resources to render the **Browse for Folder** dialog box (also called the **Select Folder** dialog box). This distinction is important when localizing an installer.

Table 7-179 • \$IA_BROWSE_FOLDERS\$ Variable Values

Value	Description
Native	<p>When the \$IA_BROWSE_FOLDERS\$ variable is set to Native, the installer renders a Browse for Folder dialog box that has a look and feel consistent with the end user's operating system. However, if you choose the Native option, installers running in languages other than the target system's locale will display text in that system's locale rather than in the installer-selected language.</p> <p>For example, if an end user installing an application on a target operating system using the English locale chooses German as the selected language in the installer, some English strings will appear on the Browse for Folder dialog box.</p>

Table 7-179 • \$IA_BROWSE_FOLDERS\$ Variable Values

Value	Description
Swing	When the \$IA_BROWSE_FOLDERS\$ variable is set to Swing , the installer renders a Browse for Folder dialog box that uses the correct locale for all languages, and it is consistent across different platforms. However, the dialog box may not provide a fully native experience for end users. It may have a different layout than the standard Browse for Folder dialog box rendered by the end user's operating system.

Comparison Between Native and Swing Settings

The following images illustrate the differences between the **Browse for Folder** dialog box rendered with **Native** resources versus **Swing** on Windows and Linux systems.

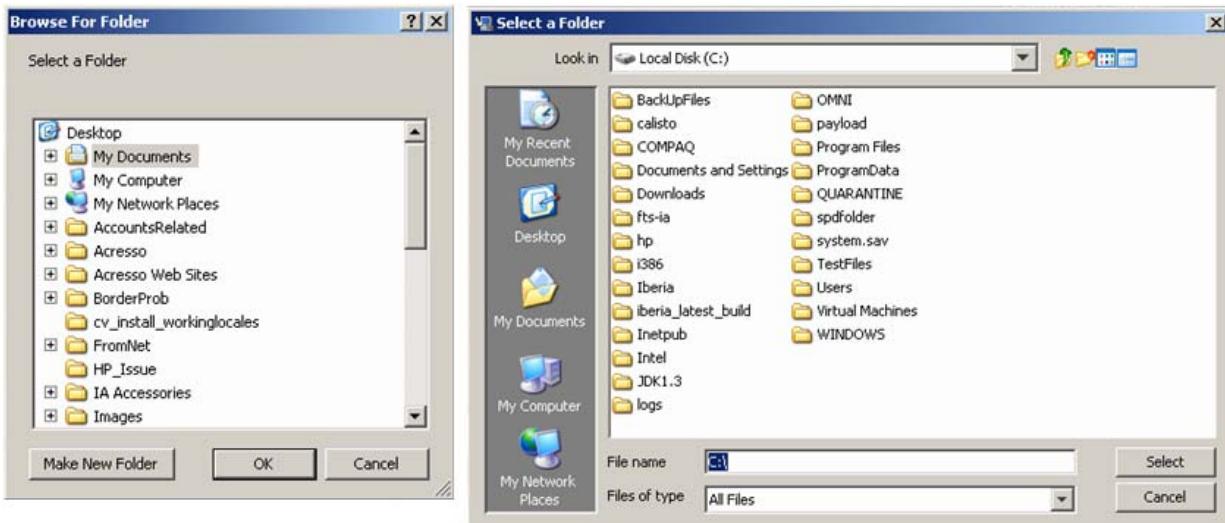


Figure 7-82: Native (Left) vs. Swing (Right) on Windows

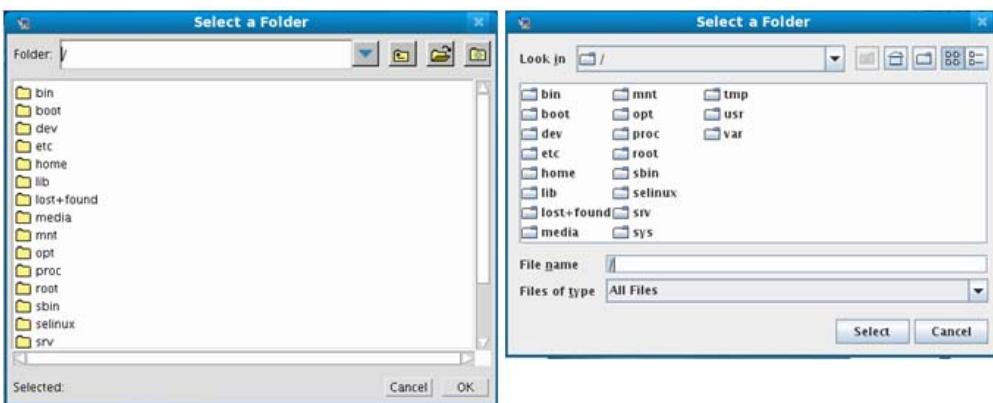


Figure 7-83: Native (Left) vs. Swing (Right) on Linux



Note • Any value other than **Native** or **Swing** for the \$IA_BROWSER_FOLDERS\$ variable causes installers to default to **Native** behavior.

LAX Properties

LaunchAnywhere properties can be set in the **LaunchAnywhere Properties** and **Uninstaller Properties** dialog boxes, accessible in the **Create LaunchAnywhere for Java Application** customizer and the **Create Uninstaller** customizer, both in the **Install** task.

Table 7-180 • LAX Properties

Property	Definition
lax.application.name	The name of the application that the launcher executes.
lax.class.path	<p>The classpath for the application. By default, set to \$IA_CLASSPATH\$ (the classpath specified in the InstallAnywhere Designer environment.)</p> <p>When you specify the classpath, use either forward (/) or backward (\) slashes to separate directories within a path. Use either colons or semicolons to separate multiple paths—LaunchAnywhere substitutes the proper characters for the installation platform at install time.</p>  <p>Tip • Never use colons as directory separators in lax.class.path. InstallAnywhere treats colons as path separators. Instead, use slashes to separate directories in a path. LaunchAnywhere replaces the slashes, as necessary, with platform-specific directory separators at install time.</p>
lax.command.line.args	<p>A list of arguments passed to the application's main method. These are specified exactly the same way as they would be on the command line. For example, if the application is invoked as</p> <pre>java myApp arg1 arg2</pre> <p>set this property to</p> <pre>"arg1 arg2"</pre> <p>Be sure to place quotes around any arguments that have spaces.</p> <p>When it is necessary to pass in an argument that is only known at install time (for instance, the installation directory), use an InstallAnywhere variable.</p>

Table 7-180 • LAX Properties (cont.)

Property	Definition
<code>lax.dir</code>	The path to directory holding LaunchAnywhere's native launcher. If you are specifying a path in Windows, make sure to use escaped backslashes (such as C:\\Program Files\\OfficeSuite.exe).
<code>lax.java.compiler</code>	The JIT compiler that is being used for execution of this application.  Note • <i>Runtime only.</i> The <code>lax.java.compiler</code> property cannot be set via the LaunchAnywhere Properties or Uninstaller Properties dialog boxes.
<code>lax.main.method</code>	The name of the application's starting method that this LaunchAnywhere Executable invokes.
<code>lax.main.class</code>	The class that gets launched by this LaunchAnywhere Executable. This class must contain a method with a name defined by the lax.main.method property.
<code>lax.nl.current.vm</code>	The full path to the VM executable to be used. If the LaunchAnywhere Executable cannot find the VM specified, it searches the system for the VMs in lax.nl.valid.vm.list .
<code>lax.nl.env.variable_name</code> <code>lax.nl.env.exact_case.variable_name</code>	Use to access any system environment variable. <ul style="list-style-type: none"> • Case insensitive—If you want to specify the property name as a case-insensitive string, use <code>lax.nl.env.variable_name</code>. For example, to access the system environment variable PATH, you could specify <code>lax.nl.env.PATH</code> or <code>lax.nl.env.path</code>. • Case sensitive—If you want to specify the property name as a string in the exact case as it is defined in the environment, use <code>lax.nl.env.exact_case.variable_name</code>. For example, to access system environment variable Path, specify <code>lax.nl.env.exact_case.Path</code>.  Note • These properties are resolved at application runtime, when LaunchAnywhere executes. You can also get access to system environment variables via InstallAnywhere variables.  Note • Windows and UNIX only.

Chapter 7: Reference

Variables

Table 7-180 • LAX Properties (cont.)

Property	Definition
lax.nl.env.path	<p>The system PATH for the computer this application is running on.</p>  <p>Note • <i>Runtime only.</i> The <code>lax.nl.env.path</code> property cannot be set via the LaunchAnywhere Properties or Uninstaller Properties dialog boxes.</p>
lax.nl.java.compiler	This property defines the name of the JIT (Just-In-Time compiler) that your application should use. Set this option to no value (that is, blank) to use the default JIT. You can also specify the name of a specific JIT by defining it in this property. Set lax.nl.java.compiler to off if no JIT is to be used.
lax.nl.java.launcher.main.class	The class that contains the main method called by LaunchAnywhere.
lax.nl.java.launcher.main.method	The name of the main method called by LaunchAnywhere.
lax.nl.java.option.additional	LaunchAnywhere writes the value of this property to the command line verbatim. Java VM properties or settings not directly supported by current LAX configuration properties can be included as part of the command line used to invoke Java. For example, to pass a custom variable as part of the command line, set lax.nl.java.option.additional to <code>-Dmyvariable=value</code> .
lax.nl.java.option.check.source	Set to on or off to tell the VM to verify bytecodes.
lax.nl.java.option.debugging	Turns debugging on or off in the Virtual Machine so that an application can be debugged. Set to on to enable debugging.
lax.nl.java.option.garbage.collection.background.thread	Defines whether or not to have a low-priority background thread that does garbage collection. Set to on or off.
lax.nl.java.option.garbage.collection.extent	Sets the behavior for garbage collection: min: Garbage collect everything except classes. full: Garbage collect everything.
lax.nl.java.option.java.heap.size.initial	Defines the initial heap size for the installer that will be invoked. This number is always specified in bytes, not in kilobytes or megabytes, and is analogous to the VM parameter <code>-ms</code> or <code>Xms</code> . The default is 16777216 (16 MB).

Table 7-180 • LAX Properties (cont.)

Property	Definition
<code>lax.nl.java.option.java.heap.size.max</code>	Defines the maximum heap size in bytes for the installer that will be invoked. This number is always specified in bytes, not in kilobytes or megabytes, and is analogous to the VM parameter <code>-mx</code> or <code>Xmx</code> . The default is 50331648 (48 MB).
<code>lax.nl.java.option.verbose</code>	Defines the level of content in output messages: <code>gc</code> : Output garbage collection messages. <code>normal</code> : Output all normal verbose messages. <code>all</code> : Output all normal and garbage collection messages. <code>none</code> : Do not output any verbose messages.
<code>lax.nl.java.option.verify.mode</code>	Sets when Java will verify classes for security and errors. Values can be <code>remote</code> , <code>all</code> , or <code>none</code> .
<code>lax.nl.message.vm.not.loaded</code>	Defines the message to show the end user in a dialog box if no VM can be found.

Chapter 7: Reference

Variables

Table 7-180 • LAX Properties (cont.)

Property	Definition
lax.nl.valid.vm.list	<p>The list of VMs that this LaunchAnywhere Executable allows the Java application to be run against. The value for this property can be any space- or comma-delimited combination of valid VM criteria supported by InstallAnywhere. Some common criteria include</p> <ul style="list-style-type: none"> ALL (any VM) JDK (any Java JDK) JRE (any Java JRE) IBM, SUN, HP, APPLE 1.4.* 1.5+ 1.6* <p>For strict selection criteria, connect vendor, type, and version operators with an underscore character (<vendor>_<type>_<version>). For example, SUN_JDK_1.4* selects VMs from Sun that are JDKs of version 1.4.0_0 or newer.</p> <p>The value of this property overrides the value listed in lax.nl.current.vm if the VM listed in that property is not of a valid type. The order of the valid VM list specifies the precedence in which VMs found on the system should be chosen if a valid VM is not listed in lax.nl.current.vm.</p>  <p>Note • For details on valid VM selection criteria, see About Java VM Selection Criteria.</p>
lax.resource.dir	The platform name in exact case.
lax.root.install.dir	The root directory of the entire installation (same as \$USER_INSTALL_DIR\$).
lax.stderr.redirect	<p>The location of your application's stderr output. Set to null to suppress, to console to write to a console window, or to any file name to output to a file.</p> <p>This property maps to the corresponding text box in the Advanced Designer (Project > Log Settings): Send stderr to. See Log Settings.</p>

Table 7-180 • LAX Properties (cont.)

Property	Definition
lax.stdout.redirect	The location of your application's stdout output. Set to null to suppress, console to write to a console window, or to any file name to output to a file. This property maps to the corresponding text box in the Advanced Designer (Project > Log Settings): Send stdout to . See Installer Settings Tab .
lax.user.dir	The working directory for your application. The default value for this property is . (period). Leave as is to set the working directory to the directory where the LaunchAnywhere Executable resides. To override the default behavior, specify an absolute or relative path. (A relative path is relative to the LaunchAnywhere Executable location.)
lax.version	The version number of the LaunchAnywhere.
LISTPROPS	This property lists all system properties available to the Java application (it can take any value.) You must redirect stdout and stderr to see the results of this output.

Magic Folders

Magic Folders represent a specific location, such as the user selected installation directory, the desktop, or the location for library files. At install time, the installer determines which operating system it is running on, and sets the magic folders to the correct absolute paths. Many Magic Folders are platform specific and many are predefined by InstallAnywhere to standard locations across InstallAnywhere supported platforms.

Table 7-181 • Magic Folders

Folder Name	InstallAnywhere Variable	Destination
User Installation Directory	\$USER_INSTALL_DIR\$	The installation folder, as specified by the end user. Developers can specify a default value for this variable in the Project Info screen in the Advanced Designer and choosing a location in the Default Install Folder area of the screen. The value of this variable changes if the end-user selects a non-default install location.

Table 7-181 • Magic Folders (cont.)

Folder Name	InstallAnywhere Variable	Destination
Programs Folder	\$PROGRAMS_DIR\$	The default application directory on the destination system. (Program Files on Windows, in the Applications folder on Mac OS X, and the logged-in-end-user's home account on Unix.)
Shortcuts	\$USER_SHORTCUTS\$	The folder specified by the end user as the shortcuts/links/aliases location. The value of this location can be changed by the end user if the Choose Alias, Link, Shortcut Folder action is turned on in the installer. Developers can specify a default value for this variable on a per-platform basis by selecting the Platforms task in the Advanced Designer.
Desktop	\$DESKTOP\$	This variable represents the desktop on the target machine. This folder only resolves on Windows, Linux, and Mac systems.
Temp Directory	\$TEMP_DIR\$	This variable represents the temp directory on the target machine. When running the pure Java installer on Windows, \$TEMP_DIR\$ will resolve the user's home directory.
Installer Temp Directory	\$INSTALLER_TEMP_DIR\$	The temp directory for use by the installer. This is deleted when the installation is complete, assuming no items are in use.
Startup	\$STARTUP\$	The automatic startup folder for items that are launched automatically during operating system boot up. This folder only resolves on Windows systems.
Installation Drive Root	\$INSTALL_DRIVE_ROOT\$	The root directory on the volume where the installation is taking place.
Home Directory	\$USER_HOME\$	The home directory of the end user running the installer. For users who have already included the variable \$UNIX_USER_HOME\$, this variable will continue to function with the same definition as \$USER_HOME\$.

Table 7-181 • Magic Folders (cont.)

Folder Name	InstallAnywhere Variable	Destination
Programs Folder (64-bit)	\$PROGRAMS_DIR_64\$	The default 64-bit application directory on a 64-bit system.
Programs Folder (32-bit)	\$PROGRAMS_DIR_32\$	The default 32-bit application directory on a system.
System Drive Root	\$SYSTEM_DRIVE_ROOT\$	The root directory of the system drive.
Java Home	\$JAVA_HOME\$	The home directory of the Java Virtual Machine to be used.
Windows	\$WIN_WINDOWS\$	The Windows directory on Windows computers.
System	\$SYSTEM\$	This variable represents the System folder on the target machine.
System Folder (64-bit)	\$SYSTEM_64\$	The default 64-bit System folder on a 64-bit system.
System Folder (32-bit)	\$SYSTEM_32\$	The default 32-bit System folder on a system.
Start Menu	\$WIN_START_MENU\$	The Start menu directory on Windows computers.
Quick Launch Bar	\$WIN_QUICK_LAUNCH_BAR\$	The Quick Launch Bar on Windows. On Windows 2000 or newer, the location is relative to the UserProfile environment variable.
Do Not Install	\$DO_NOT_INSTALL\$	A special Magic Folder for installer files that are not needed on the target platform. \$DO_NOT_INSTALL\$ is most commonly applied to files (typically localized license agreements and graphics) that are used during installation but need not remain on the target system.

Table 7-181 • Magic Folders (cont.)

Folder Name	InstallAnywhere Variable	Destination
USER_MAGIC_FOLDER_n	\$USER_MAGIC_FOLDER_n\$	<p>These variables are user-defined install-destination Magic Folders. They install to whichever directories their variable name has been set. To set these variables, use one of the Set InstallAnywhere Variable actions.</p> <p>For Unix systems, if the leading / is omitted in the path before the Magic Folder location, the location resolves to</p> <pre><installer location>/ USER_MAGIC_FOLDER_n.</pre> <p>Unix-based operating systems assume that any path not preceded by a / is below the current directory.</p>  <p>Note • In previous releases, you were limited to a maximum of 10 user-defined Magic Folders. Starting with InstallAnywhere 2011, the number of permitted user-defined Magic Folders has increased to 25.</p>
Programs Menu	\$WIN_PROGRAMS_MENU\$	The Programs menu (in the Start menu) on Windows systems.
All Users Start Menu	\$WIN_COMMON_START_MENU\$	The All Users Start menu directory for Windows computers.
All Users Programs Menu	\$WIN_COMMON_PROGRAMS_MENU\$	The All Users Programs menu (in the Start menu) directory for Windows computers.
All Users Startup	\$WIN_COMMON_STARTUP\$	The All Users Startup folder (in the Start menu) on Windows computers.
All Users Desktop	\$WIN_COMMON_DESKTOP\$	The Windows Common Desktop folder on Windows computers.
Fonts	\$FONTS\$	<p>The Fonts directory on Windows computers. \$WIN_WINDOWS\$\Fonts, for example.</p> <p>See Installing Fonts for more information.</p>
User Applications	\$MACX_USER_APPLICATIONS\$	The User Applications directory of the end user running the installer (Mac OS X only).

Table 7-181 • Magic Folders (cont.)

Folder Name	InstallAnywhere Variable	Destination
The Dock	\$MACX_DOCK\$	The Mac OS X Dock (for Shortcuts only). Other types of files cannot be installed to the Dock.
/usr/local/bin	\$UNIXUSR_LOCAL_BIN\$	The /usr/local/bin directory (Unix computers only).
/opt	\$UNIX_OPT\$	The /opt directory (UNIX computers only).
/usr/bin	\$UNIXUSR_BIN\$	The /usr/bin directory (Unix computers only).



Edition • Developer-defined Magic Folders (\$USER_MAGIC_FOLDER_n\$) are not available in InstallAnywhere Standard Edition.

Localization Reference

InstallAnywhere's Enterprise edition allows developers to build installers for up to 31 different languages.

InstallAnywhere's Standard edition enables developers to build installers for up to 9 languages.

Section	Description
Language Codes	Lists languages and language codes supported by InstallAnywhere Enterprise edition.
Common Localizable Elements	Identifies many elements that most frequently require localization.

Language Codes

InstallAnywhere supports the following languages.



Edition • These are the languages supported in the Enterprise Edition. Standard Edition supports only nine languages: Simplified Chinese, Traditional Chinese, English, French, German, Italian, Japanese, Spanish, and Swedish.

Table 7-182 • Language Codes

Language Code	Language
ar	Arabic
ca	Catalan
cs	Czech
da	Danish
de	German
el	Greek
en	English
es	Spanish
eu	Basque
fi	Finnish
fr	French
fr_CA	French (Canada)
hu	Hungarian
in	Indonesian
it	Italian
iw	Hebrew
ja	Japanese
ko	Korean
nl	Dutch
no	Norwegian

Table 7-182 • Language Codes (cont.)

Language Code	Language
pl	Polish
pt	Portuguese
pt_BR	Portuguese (Brazil)
ru	Russian
sk	Slovak
sl	Slovenian
sv	Swedish
th	Thai
tr	Turkish
zh_CN	Chinese (Simplified)
zh_TW	Chinese (Traditional)



Note • Some codes shown in the Language Code column, above, actually include a language code plus a country code to fully specify the supported language. For example, pt_BR for Brazilian Portuguese and zh_TW for traditional Chinese.

Common Localizable Elements



Important • Installers deployed to non-Latin systems require an international Java Virtual Machine.

This list identifies and describes many of the built-in localizable elements in an InstallAnywhere installer. As you build your installer, the localizable elements in your language files changes to match the panels, actions, and custom code you employ.

Element keys typically use the following format:

<component>.<referenceID>.<element>

The referenceID is automatically generated by InstallAnywhere and unique for each project.

Example: `InstallerInfoData.68a9edb1a601.installerTitle`.

Table 7-183 • Language Resource Properties

Property	Definition
<code>Installer.#.ProductName</code>	Name of product displayed on installer title bar.
<code>Installer.#.splashScreenGUITitle</code>	<p>Title of the installer splash screen.</p> <p>The splash screen title bar only appears when the installer supports more than one locale; otherwise only the splash screen image is shown.</p> <p>The ability to localize the splash screen is available only in InstallAnywhere Enterprise edition.</p>
<code>Installer.#.splashScreenGUImagePath</code>	Path to the splash screen image file.
<code>Installer.#.splashScreenGUIName</code>	<p>File name of the splash screen image file.</p> <p>The ability to localize the splash screen is available only in InstallAnywhere Enterprise edition.</p>
<code>Installer.#.splashScreenGUIConfirm</code>	<p>Text for splash screen button that confirms the installer locale setting. (Default: OK)</p> <p>The confirmation button only appears on the splash screen when the installer supports more than one locale.</p> <p>The ability to localize the splash screen is available only in InstallAnywhere Enterprise edition.</p>
<code>Installer.#.splashScreenGUINSTRUCTIONS</code>	<p>The text that appears to describe how to choose a locale on the splash screen. (Default: blank.)</p> <p>These instructions only appear on the splash screen when the installer supports more than one locale.</p> <p>The ability to localize the splash screen is available only in InstallAnywhere Enterprise edition.</p>
<code>Installer.#.splashScreenConsoleTitle</code>	<p>The text for the title of the Choose Locale console. (Default: Choose Locale...)</p> <p>The Choose Locale console only appears when the console installer supports more than one locale.</p> <p>The ability to localize the splash screen is available only in InstallAnywhere Enterprise edition.</p>

Table 7-183 • Language Resource Properties (cont.)

Property	Definition
Installer.#.splashScreenConsolePrompt	<p>The text for the prompt on the Choose Locale console. (Default: CHOOSE LOCALE BY NUMBER)</p> <p>The Choose Locale console only appears when the console installer supports more than one locale.</p> <p>The ability to localize the splash screen is available only in InstallAnywhere Enterprise edition.</p>
Installer.#.RulesFailedMessage	Message displayed if specified rules keep the installer from running.
Installer.#.ShortcutDestinationPathMacOS	Path to where aliases are created during installation on Mac OS (relative to the end-user-selected alias folder chosen on the Choose Alias Location step).
Installer.#.ShortcutDestinationPathWin32	Path to where shortcuts are created during installation on Windows (relative to the end-user-selected shortcut folder chosen on the Choose Alias, Link, Shortcut Folder step).
Installer.#.ShortcutDestinationPathSolaris	Path to where links are created during installation on Unix (relative to the end-user-selected links folder chosen on the Choose Link Location step).
InstallSet.#.Description	Description of one of the installer's features.
InstallSet.#.InstallSetName	Name of one of the installer's features.
IntroAction.#.message	Text to display on the installer's Introduction step.
Intro.#.stepTitle	Title to display on the installer's Introduction step.
LicenseAgr.#.FileName	<p>Name of localized license agreement to be displayed as the installer is preparing itself.</p>  <p>Note • This is a filename only—not a fully qualified absolute path name.</p>
LicenseAgr.#.Path	<p>Path name to localized license agreement to be displayed as the installer is preparing itself.</p>  <p>Note • This is only a path name and does not include the filename itself.</p>

Table 7-183 • Language Resource Properties (cont.)

Property	Definition
LicenseAgr.#.Title	Title of License Agreement step in the installer.
MakeExecutable.#.destinationName	Name of the LaunchAnywhere Executable to be created on the destination computer.
MakeRegEntry.#.Value	Value to be written into the Win32 Registry.
ShortcutLoc.#.macTitle	Title of Mac OS X Choose Alias Location step in the installer.
ShortcutLoc.#.SolarisTitle	Title of Unix Choose Alias, Link, Shortcut Folder step in the installer.
ShortcutLoc.#.Win32Title	Title of Win32 Choose Alias, Link, Shortcut Folder step in the installer
Billboard.#.ImageName	<p>Name of billboard image file to be displayed as the installer is preparing itself.</p>  <p>Note • This is a filename only—not a fully qualified absolute path name.</p>
Billboard.#.ImagePath	<p>Path name to billboard image to be displayed as the installer is preparing itself.</p>  <p>Note • This is only a path name and does not include the file name itself.</p>
ChooseInstallSet.#.Title	Title of Choose Feature step in the installer.
ChooseJavaVM.#.Title	Title of Choose Java Virtual Machine step in the installer.
CreateShortcut.#.DestinationName	Name of the shortcut/alias/link to be created on the destination computer.
human.readable.language.name	The name of the language represented by the data in this resource file (for example, English, Espanol, and so on).

Table 7-183 • Language Resource Properties (cont.)

Property	Definition
ImportantNote.#.FileName	Name of text file to be displayed on the Important Note step of the installer.  Note • This is a filename only—not a fully qualified absolute path name.
ImportantNote.#.Path	Path name to text file to be displayed on the Important Note step of the installer.  Note • This is only a path name and does not include the filename itself.
ImportantNote.#.Title	Title of Important Note step in installer
InstallBundle.#.BundleName	Name of component.
InstallBundle.#.Description	Description text describing component.
InstallComplete.#.DisplayText	Text to display on the Install Complete step of the installer.
InstallComplete.#.Title	Title of the Install Complete step in the installer.
InstallDir.#.Title	Title of the Choose Installation Directory step of the installer.
Installer.#.InstallerName	Name of the installer.



Note • External resource bundles are an alternative to InstallAnywhere's built-in locale files. External resource bundles allow you to define your own keys in a collection of locale properties files and reference them using a `$L{bundle_name.key}` syntax. For more information, see [External Resource Bundles](#).

Files and File Formats

The topics in this section describe some of the output and working files that InstallAnywhere creates.

Table 7-184 • InstallAnywhere Files and Formats

Topic	Description
Product Registry	Discusses the content and location (by platform) for the InstallAnywhere product registry.
Install Log File Format	Describes the purpose, format options, content, and naming conventions of the install logs InstallAnywhere creates.
Manifest Files	Describes the use and content of manifest files in InstallAnywhere.
Response Files	Discusses the purpose, format, and content of the installer.properties files InstallAnywhere creates and consumes as response files for silent installs.
BuildProperties.xml File	Explains how to use a build properties XML file, such as BuildProperties.xml, to pass build properties to the command-line builder in a single file.
buildproperties.properties File	Explains how to use a .properties file, such as buildproperties.properties, to pass build properties to the command-line builder in a single file.



Note • For information about the content and structure of debug output—both from the Project > Log Settings settings and from the Output Debug Information action—see [Reviewing Debug Information](#).

Product Registry

The product registry is essentially a product configuration database which keeps track of features and components of products and accomplishes tasks such as associating file name extensions with applications. Proper use of product IDs and versions are essential to the operation of the registry in the future—for example, the use of the Find Component in Registry action.

InstallAnywhere finds the locations of components in the registry by checking the Product ID. To make reliable use of the Find Component in Registry action, you must enter the Product ID and Version correctly.

The product registry stores the information from Project > Description. The location of the registry depends on the platform to which you are installing.

Table 7-185 • Product Registry Location by Platform

Platform	Location
Windows	C:\Program Files\Zero G Registry\.com.zerog.registry.xml (hidden directory)
UNIX-based	If logged in as root, the global registry is located in \var. If logged in as a user, it is located in the user's home directory.
Mac OS X	/Library/preferences/



Note • It is useful to keep the Vendor ID the same for all the products from a single organization, but it isn't required. Keeping a common Vendor ID allows you to identify all of your organization's products installed on a target machine.

Install Log File Format

InstallAnywhere installers can generate install logs to record the results of a particular install. These files can be generated in either plain text or XML format. No matter which format you choose, the log content is essentially the same.



Note • *The generation of install logs is set at the project level on the **Project > Info** task.*

Install Log Content

Both logs begin with a date stamp and a collection of startup information followed by sections for user interactions, action summary, installation summary, and install log details.

Table 7-186 • Install Log Content

Section	Description
Startup Information	With a few minor differences, the initial content of the install log is similar to the content of the installer debug output produced when you provide a file name for Send stderr to and Send stdout to on the Project > Log Settings task. It includes <ul style="list-style-type: none">• Time and date at which the install was run• Memory statistics: Free Memory and Total Memory• Java class path• ZGUtil class path• Sun boot class path• Java extensions path• Java properties
User Interactions	Records the name of the panel or action that acquired user input, the variable set by that input, and the value of that variable.
Action Summary	Summarizes actions by listing the number of action successes, warnings, non-fatal errors, and fatal errors. This section also lists any action notes.
Installation Summary	Shows the resulting status of the install as well as the install start and end times.
Install Log Details	Lists the actions in the sequence they occurred, the targets and destination directories, and the status of the action. Each entry also includes additional notes, such as the disk space required and disk space available that is included for the Disk Space Check action.

Install Log Naming Conventions

InstallAnywhere names the install log based on the value of \$PRODUCT_NAME\$:

- \$PRODUCT_NAME\$_InstallLog.log for the plain text logs
- \$PRODUCT_NAME\$_InstallLog.xml for the XML logs

Manifest Files

Manifest files are text files which specify a list of files and directories. Using a manifest file you create outside of InstallAnywhere, you can identify files and directories to be included in the build. At build time, the manifest file is analyzed and its contents are placed into the installer.



Note • In addition to creating the manifest file, you must also add an **Install from Manifest** action. See [Install Actions](#).

Manifest File Format

The manifest file format specifies the file's source, its destination (which is relative to the location of the action in the Visual Tree of the Install task), and optionally, which Unix file permissions it should have and whether it should be placed on the classpath.

Syntax for File References

```
F,[SOURCEPATH]relative_path_to_source_file,./relative_path_to_destination_file  
F,absolute_path_to_source_file,./relative_path_to_destination_file
```

Syntax for Putting Files in the Classpath

```
F,absolute_path_to_source_file,./relative_path_to_destination_file,cp
```

Syntax for Setting File Permissions on UNIX

```
F,[SOURCEPATH]relative_path_to_source_file,./relative_path_to_destination_file,755
```

Syntax for Directory References

```
D,[SOURCEPATH]relative_path_to_source_dir[/],./relative_path_to_destination_dir[/]  
D,absolute_path_to_source_dir[/],./relative_path_to_destination_dir[/]
```

Examples

```
F,$IA_HOME$/path/to/source/file.txt,./destination/path/thisfile.txt  
F,/absolute/path/to/source/file.txt,./destination/path/thisfile.txt,cp,655  
D,$IA_HOME$/path/to/dir,./destination/path/dir  
D,/absolute/path/to/dir,./destination/path/dir
```

Chapter 7: Reference

Files and File Formats



Tip • On the General tab of the InstallAnywhere Preference dialog box, you can specify whether a build should stop or continue if a manifest file referenced in the project is not available. See [General Settings](#).

Response Files

Response files are installer properties files that are generated by capturing the default variable values and user responses from the execution of an installer. The record of these responses can be used to control subsequent installer executions. Typically, response files use the default name `installer.properties` and provide settings to support an installer running in silent mode. Response files are essentially just text files that can provide settings for an installer.

Generating Response Files

You can choose to generate a response file by selecting an option in the Advanced Designer or by using the `-r` command line switch.

- **Selecting an option in the Advanced Designer**—You can specify that a response file is generated each time an installation is run by selecting the **Always Generate Response File** option on the **Project > Info** subtask of the Advanced Designer.

If you generate a response file by selecting this option, the response file is always named `installer.properties` or `[installername].properties` and will be created in the same directory as the installer.

- **Using the `-r` command line switch**—You can also generate a response file when running an installation via command line by using the `-r` command line switch followed by the path and file name of the response file you want to generate, such as:

```
install.exe -r "/Users/MyName/myresponse.properties"
```

If you do not specify the path and file name of the response file, it will be named `installer.properties` or `[installername].properties` and will be created in the same directory as the installer.

Using Response Files

Response files can be used to drive an installer in one of two ways:

- **Put in same directory as installer**—Place a response file named `installer.properties` in the same directory as the installer, and the installer will attempt to use that file as input to the installer.
- **Specify using the `-f` command line switch**—Use the `-f` command-line switch when you run the installer to specify a response file for the installer to use.

Choose Install Sets Variables

The **Choose Install** Set panel/console variables are recorded in response files. Here is an example:

Chapter 7: Reference

Files and File Formats

```
#Choose Product Features
#-----
CHOSEN_FEATURE_LIST=Application,Help
CHOSEN_INSTALL_FEATURE_LIST=Application,Help
CHOSEN_INSTALL_SET=Typical
```

Sample Response File

The following is a sample `installer.properties` file:

```
# Mon Jul 14 17:45:12 CDT 2010
# Replay feature output
# -----
# This file was built by the Replay feature of InstallAnywhere.
# It contains variables that were set by Panels, Consoles or Custom Code.

#Choose Install Folder
#-----
USER_INSTALL_DIR=C:\\Program Files\\OfficeSuite

#Choose Shortcut Folder
#-----
USER_SHORTCUTS=C:\\Program Files\\OfficeSuite\\OfficeSuite

#Choose Product Features
#-----
CHOSEN_FEATURE_LIST=Application,Help
CHOSEN_INSTALL_FEATURE_LIST=Application,Help
CHOSEN_INSTALL_SET=Typical

#Set Eclipse Location
#-----
USER_INPUT_RESULT_0=C:\\Eclipse

#Install
#-----
-fileOverwrite_c:\\progfile.txt=Yes
```

Recording User Response to File Overwrite Prompts in Response File

If you set a file's **If the file already exists on the end user's system** option in the **Install File** customizer of the **Install** task to **Always prompt user**, the overwrite choice that the user makes is recorded in the response file using the `-fileOverwrite` command.

When the **Always prompt user** option is selected and a file overwrite situation occurs during installation, the user is then prompted to select one of the following options: **Yes**, **Yes to All**, **No**, or **No to All**.

The selection that the user makes in response to being prompted for file overwrite behavior is recorded in the response file by the `-fileOverwrite` command using the following syntax:

```
-fileOverwrite_<name of file>=value
```

where `value` would be one of the following:

- YesToAll
- Yes

- No
- NoToAll

For example:

```
-fileOverwrite_c:\\abc.txt=Yes
```

When an application is installed for the first time, the -fileOverwrite command for each file is set to Yes.

BuildProperties.xml File

You can use a build properties XML file, such as BuildProperties.xml, to pass build properties to the command-line builder in a single file.

To use this properties file, you use the -p argument followed by the path to the Build Properties file:

```
build.exe <projectfile> -p <path to BuildProperties.xml>
```

If you do not provide an absolute path to your build properties file, the builder will look for it in the same directory as your project.



Important • The BuildProperties.xml file supports the specification of multiple Build Configurations. The Build Configurations specified in this file will override those defined in the project.

A Build Properties file template named BuildProperties.xml can be found at:

```
<InstallAnywhere folder>/resource/build/BuildProperties.xml
```

It provides a sample of all possible build settings and can be customized to suit your build requirements.

Ant Properties

Starting with InstallAnywhere 2011, ant properties are now supported in buildproperties.xml. An ant property can be defined in the buildproperties.xml using:

```
<property name="ia.home" value="D:\\IA_Codebase\\main" />
```



Note • For more information, see [InstallAnywhere Ant Task Reference](#).

BuildProperties.xml File Settings in the Advanced Designer

Most of the build settings that you can make in the Advanced Designer can also be set in a BuildProperties.xml file. The following tables correlate the options in the Advanced Designer with settings in the BuildProperties.xml file.

- [Build Task](#)
- [Installer UI > Look & Feel Subtask](#)

Chapter 7: Reference

Files and File Formats

- [Organization > Components](#)
- [Project > Platforms Task](#)
- [Project > Log Settings Task](#)
- [Project > JVM Settings Task](#)

Build Task

The following tables correlate the options in the **Build** task of the Advanced Designer with settings in the `BuildProperties.xml` file.

- [Build Configurations Tab](#)
- [Build Targets Subtab](#)
- [Distribution Subtab](#)

Build Configurations Tab

The following table correlates the settings that can be made on the **Build Configurations** tab of the Advanced Designer **Build** task with settings in a `BuildProperties.xml` file:

Table 7-187 • Build Configurations Tab / BuildProperties.xml File

Option	BuildProperties.xml File Setting
Select Build Configuration	<p>Create a <code><configuration></code> element for each Build Configuration that you want to build, using the following syntax:</p> <pre><configuration name="name_of_build_configuration"> <webInstaller enable="true/false" optimize="true/false"> <language>en/ja</language> </webInstaller> <cdRomInstaller enable="true/false" optimize="true/false" /> <mergeModule enable="true/false" optimize="true/false" readOnly="true/false" /> <target platform="aix/hpux/solaris/hpux/unixwithvm/macosx /java/unix/windows" buildWithVM="true/false" buildWithNoVM="true/false" outputDir="myOutputDir" bundledVM="path_to_file" /> </configuration></pre> <p>Regarding each <code><target></code> element, note the following:</p> <ul style="list-style-type: none"> • Each <code><target></code> element appends one new target to those already in the Build Configuration. • There can be multiple <code><target></code> elements. • The <code>outputDir</code> and <code>bundledVM</code> attributes are optional. • Each <code><target></code> element must include a <code>buildWithVM</code> attribute, a <code>buildWithNoVM</code> attribute, or both.  <p>Important • In previous releases, <code>buildWithVM</code>, <code>BuildWithNoVM</code>, <code>outputDir</code>, and <code>bundledVM</code> were subelements of the <code><target></code> element. Starting with InstallAnywhere 2011, they have been promoted to be attributes of the <code><target></code> element. Therefore, scripts written for previous releases that contain these items will need to be updated.</p>  <p>Important • The following subelements of the <code><configuration></code> element can also be specified outside of a <code><configuration></code> element. However, those elements defined within a <code><configuration></code> element override those defined elsewhere.</p>
Build Output Location	<code>BuildOutputLocation="<path_to_directory>"</code>

Chapter 7: Reference

Files and File Formats

Table 7-187 • Build Configurations Tab / BuildProperties.xml File

Option	BuildProperties.xml File Setting
[Working Directory]	BuildWorkdirLocation=" <path_to_directory>"  Note • The working directory, which is used to store all temporary files (such as the build log and the locales directory) is set by default to the InstallAnywhere project directory. It is not specifically set in the Advanced Designer user interface.</path_to_directory>

Build Targets Subtab

The following table correlates the settings that can be made on the **Build Targets** subtab of the Advanced Designer **Build** task with settings in a `BuildProperties.xml` file:

Table 7-188 • Build Targets Subtab / BuildProperties.xml File

Option	BuildProperties.xml File Setting
Max OSX	BuildMacOSX=" <true false="">" WantAuthenticationMacOSX="<true false="">" WantAuthenticationMacOSXShowGUI="<true false="">"</true></true></true>
Windows	BuildWindowsWithVM=" <true false="">" BuildWindowsWithoutVM="<true false="">" WindowsVMpackLocation="<path_to_file>"</path_to_file></true></true>
AIX	BuildAIXWithVM=" <true false="">" BuildAIXWithoutVM="<true false="">" AIXVMpackLocation="<path_to_file>"</path_to_file></true></true>
HP-UX	BuildHPUXWithVM=" <true false="">"- BuildHPUXWithoutVM="<true false="">" HPUXVMpackLocation="<path_to_file>"</path_to_file></true></true>
Linux	BuildLinuxWithVM=" <true false="">" BuildLinuxWithoutVM="<true false="">" LinuxVMpackLocation="<path_to_file>"</path_to_file></true></true>
Solaris	BuildSolarisWithVM=" <true false="">" BuildSolarisWithoutVM="<true false="">"- SolarisVMpackLocation="<path_to_file>"</path_to_file></true></true>

Table 7-188 • Build Targets Subtab / BuildProperties.xml File

Option	BuildProperties.xml File Setting
Unix (All)	BuildUnixAll=" <true false="">"</true>
UNIX_with_VM	BuildNamedUnixWithVM=" <true false="">" BuildNamedUnixWithoutVM="<true false="">" NamedUnixVMpackLocation="<path_to_file>" NamedUnixTitle="<name_of_unix>"</name_of_unix></path_to_file></true></true>
Other Java-Enabled Platforms	BuildPureJava=" <true false="">"</true>
[Not available in the Advanced Designer]	OverrideAllPlatformSettings=" <true false="">"</true>

Distribution Subtab

The following table correlates the settings that can be made on the **Distribution** subtab of the Advanced Designer **Build** task with settings in a **BuildProperties.xml** file:

Table 7-189 • Distribution Subtab / BuildProperties.xml File

Option	BuildProperties.xml File Setting
Build Web Installers	BuildWebInstaller=" <true false="">"</true>
Optimize [Web] Installer Size by Platform and Tags	OptimizeWebInstaller=" <true false="">"</true>
Web page displays in	This option is set within a <code><webInstaller></code> element inside of a Build Configuration element, <code><configuration></code> : <code><configuration name="<name_of_build_configuration>"></name_of_build_configuration></code> <code> <webInstaller></code> <code> <language><en/ja></language></code> <code> </webInstaller></code> <code> ...</code> <code></configuration></code> 
	<p>Note • See Select Build Configuration for more information.</p>
Build CD-ROM installers	BuildCDROMInstaller=" <true false="">"</true>
Optimize [CD-ROM] Installer Size by Platform and Tags	OptimizeCDROMInstaller=" <true false="">"</true>
Build Merge Module Template	BuildMergeModule=" <true false="">"</true>

Chapter 7: Reference

Files and File Formats

Table 7-189 • Distribution Subtab / BuildProperties.xml File

Option	BuildProperties.xml File Setting
Optimize Merge Module/ Template Size by Platform and Tags	OptimizeMergeModule="<true/false>"
Read Only	BuildReadOnlyMergeModule="<true/false>"

Installer UI > Look & Feel Subtask

The following table correlates an option on the **Installer Steps** tab of the **Installer UI > Look & Feel** subtask of the Advanced Designer with a setting in the BuildProperties.xml file:

Table 7-190 • Installer UI > Look & Feel Subtask (Installer Steps Tab) / BuildProperties.xml File

Option	BuildProperties.xml File Setting
Auto populate labels when saving	AutoPopulateLabels="<true/false>"

Organization > Components

The following table correlates an option on the **Organization > Components** subtask of the Advanced Designer with a setting in the BuildProperties.xml file:

Table 7-191 • Organization > Components / BuildProperties.xml File

Option	BuildProperties.xml File Setting
Auto-clean when building	AutoCleanComponents="<true/false>"

Project > Platforms Task

The following table correlates options in the **Project > Platforms** subtask of the Advanced Designer with settings in the BuildProperties.xml file.

Table 7-192 • Project > Platform Task / BuildProperties.xml File

Platform	Option	BuildProperties.xml File Setting
UNIX	Default Unix Installer UI Mode / Installer UI Mode	UnixDefaultUI="<Silent/GUI/Console>"
Windows	Default Windows UI Mode / Installer UI Mode	WindowsDefaultUI="<Silent/GUI/Console>"

Project > Log Settings Task

The following table correlates options in the **Project > Log Settings** task of the Advanced Designer with settings in the BuildProperties.xml file.

Table 7-193 • Project > Log Settings Task / BuildProperties.xml File

Option	BuildProperties.xml File Setting
Send stderr to	InstallerStdErrRedirect=<path_to_file>"
Send stdout to	InstallerStdOutRedirect=<path_to_file>"

Project > JVM Settings Task

The following table correlates options in the **Project > JVM Settings** task of the Advanced Designer with settings in the BuildProperties.xml file.

Table 7-194 • Project > JVM Settings Task / BuildProperties.xml File

Option	BuildProperties.xml File Setting
Valid VM list	InstallerValidVMList=<1.3+, 1.6*>"
Minimum Heap Size	InstallerInitialHeapSize=<16777216>"
Maximum Heap Size	InstallerMaxHeapSize=<50331648>"

buildproperties.properties File

You can choose to use a buildproperties.properties file to build installers for a project. To do this, you use the -p argument followed by the path to the buildproperties.properties file:

```
build.exe <projectfile> -p <path to the buildproperties.properties>
```

The settings specified in this properties file would override build settings in the project.

A sample buildproperties.properties file can be found at:

```
<InstallAnywhere folder>/resource/build/buildproperties.properties
```

buildproperties.properties File Settings in the Advanced Designer

Most of the build settings that you can make in the Advanced Designer can also be set in a buildproperties.properties file. The following tables correlate the options in the Advanced Designer with settings in the buildproperties.properties file.

- [Build Task](#)
- [Installer UI > Look & Feel Subtask](#)
- [Organization > Components](#)
- [Project > Platforms SubTask](#)
- [Project > Log Settings Task](#)

Build Task

The following tables correlate the options in the **Build** task of the Advanced Designer with settings in the buildproperties.properties file.

- [Build Configurations Tab](#)
- [Build Targets Subtab](#)
- [Distribution Subtab](#)
- [Locales Subtab](#)

Build Configurations Tab

The following table correlates the settings that can be made on the **Build Configurations** tab of the Advanced Designer **Build** task with settings in a buildproperties.properties file:

Table 7-195 • Build Configurations Tab / buildproperties.properties File

Option	buildproperties.properties File Setting
Select Build Configuration	<pre>number.of.configs=2 config.1.name=c2 config.2.name=c1</pre> <p>In the buildproperties.properties file, the build target, distribution, and locale settings include a number to identify its associated Build Configuration, such as:</p> <pre>config.1.com.zerog.ia.build.platform.windows.novm=false</pre> <p>For more information, see Build Targets Subtab, Distribution Subtab, and Locales Subtab.</p>
Build Output Location	<code>com.zerog.ia.build.options.output.location=path_to_directory</code>
[Working Directory]	<code>com.zerog.ia.build.options.workdir.location=path_to_directory</code>  <p>Note • The working directory, which is used to store all temporary files (such as the build log and the locales directory) is set by default to the InstallAnywhere project directory. It is not specifically set in the Advanced Designer user interface.</p>

Build Targets Subtab

The following table correlates the settings that can be made on the **Build Targets** subtab of the Advanced Designer **Build** task with settings in a buildproperties.properties file:

Table 7-196 • Build Targets Subtab / buildproperties.properties File

Option	buildproperties.properties File Setting
Max OSX	<code>config.1.com.zerog.ia.build.platform.macosx.novm=true</code>
Windows	<code>config.1.com.zerog.ia.build.platform.windows.novm=false</code> <code>config.1.com.zerog.ia.build.platform.windows.vm=false</code> <code>config.1.com.zerog.ia.build.vmpack.windows.path=</code>
AIX	<code>config.1.com.zerog.ia.build.platform.aix.novm=true</code> <code>config.1.com.zerog.ia.build.platform.aix.vm=false</code> <code>config.1.com.zerog.ia.build.vmpack.aix.path=</code>
HP-UX	<code>config.1.com.zerog.ia.build.platform.hpx.novm=false</code> <code>config.1.com.zerog.ia.build.platform.hpx.vm=false</code> <code>config.1.com.zerog.ia.build.vmpack.hpx.path=</code>

Chapter 7: Reference

Files and File Formats

Table 7-196 • Build Targets Subtab / buildproperties.properties File

Option	buildproperties.properties File Setting
Linux	config.1.com.zerog.ia.build.platform.linux.novm=false config.1.com.zerog.ia.build.platform.linux.vm=false config.1.com.zerog.ia.build.vmpack.linux.path=
Solaris	config.1.com.zerog.ia.build.platform.solaris.novm=false config.1.com.zerog.ia.build.platform.solaris.vm=false config.1.com.zerog.ia.build.vmpack.solaris.path=
UNIX_with_VM	config.1.com.zerog.ia.build.platform.named_unix.vm=false config.1.com.zerog.ia.build.platform.named_unix.novm=false config.1.com.zerog.ia.build.platform.unix.novm=false config.1.com.zerog.ia.build.vmpack.unix.path=
Other Java-Enabled Platforms	config.1.com.zerog.ia.build.platform.java.novm=false
[Not available in the Advanced Designer]	com.zerog.ia.build.options.ignoreAllPlatformSettings=true

Distribution Subtab

The following table correlates the settings that can be made on the **Distribution** subtab of the Advanced Designer **Build** task with settings in a buildproperties.properties file:

Table 7-197 • Distribution Subtab / buildproperties.properties File

Option	buildproperties.properties File Setting
Build Web Installers	config.1.com.zerog.ia.build.options.output.web=false
Optimize [Web] Installer Size by Platform and Tags	config.1.com.zerog.ia.build.options.optimization.platform.web=false
Web page displays in	config.1.com.zerog.ia.build.options.webpage.language=en
Build CD-ROM installers	config.1.com.zerog.ia.build.options.output.cdrom=true
Optimize [CD-ROM] Installer Size by Platform and Tags	config.1.com.zerog.ia.build.options.optimization.platform.cdrom=true
Build Merge Module Template	config.1.com.zerog.ia.build.options.output.merge=true
Optimize Merge Module/Template Size by Platform and Tags	config.1.com.zerog.ia.build.options.optimization.platform.merge=false

Table 7-197 • Distribution Subtab / buildproperties.properties File

Option	buildproperties.properties File Setting
Read Only	config.1.com.zerog.ia.build.options.output.merge.read.only=false

Locales Subtab

The following table correlates the settings that can be made on the **Locales** subtab of the Advanced Designer **Build** task with settings in a buildproperties.properties file:

Table 7-198 • Locales Subtab / buildproperties.properties File

Option	buildproperties.properties File Setting
Locale List	config.1.com.zerog.ia.build.options.locales=en,ja,de,ar,fr

Installer UI > Look & Feel Subtask

The following table correlates an option on the **Installer Steps** tab of the **Installer UI > Look & Feel** subtask of the Advanced Designer with a setting in the buildproperties.properties file:

Table 7-199 • Installer UI > Look & Feel Subtask (Installer Steps Tab) / buildproperties.properties File

Option	buildproperties.properties File Setting
Auto populate labels when saving	com.zerog.ia.build.options.auto-populate.labels=true

Organization > Components

The following table correlates an option on the **Organization > Components** subtask of the Advanced Designer with a setting in the buildproperties.properties file:

Table 7-200 • Organization > Components / buildproperties.properties File

Option	buildproperties.properties File Setting
Auto-clean when building	com.zerog.ia.build.options.auto-claen.components=true

Project > Platforms SubTask

The following table correlates options in the **Project > Platforms** subtask of the Advanced Designer with settings in the buildproperties.properties file.

Table 7-201 • Project > Platform Subtask / buildproperties.properties File

Platform	Option	buildproperties.properties File Setting
UNIX	Default Unix Installer UI Mode / Installer UI Mode	default.ui.mode.unix=GUI

Chapter 7: Reference

Files and File Formats

Table 7-201 • Project > Platform Subtask / buildproperties.properties File

Platform	Option	buildproperties.properties File Setting
Windows	Default Windows UI Mode / Installer UI Mode	default.ui.mode.windows=GUI

Project > Log Settings Task

The following table correlates options in the **Project > Log Settings** task of the Advanced Designer with settings in the buildproperties.properties file.

Table 7-202 • Project > Log Settings Task / buildproperties.properties File

Option	buildproperties.properties File Setting
Send stderr to	com.zerog.ia.installer.options.stderr.redirect=path_to_file
Send stdout to	com.zerog.ia.installer.options.stdout.redirect=path_to_file

Project > JVM Settings Task

The following table correlates options in the **Project > JVM Settings** task of the Advanced Designer with settings in the buildproperties.properties file.

Table 7-203 • Project > JVM Settings Task / buildproperties.properties File

Option	buildproperties.properties File Setting
Valid VM list	com.zerog.ia.installer.options.valid.vm.list=1.4.*, 1.5+
Minimum Heap Size	com.zerog.ia.installer.options.heap.size.initial=16777216
Maximum Heap Size	com.zerog.ia.installer.options.heap.size.max=50331648

Command-Line Reference

InstallAnywhere's installers and uninstallers, build program, and launchers accept command-line arguments that dictate, to some extent, how those programs run.

Table 7-204 • InstallAnywhere Command-Line Reference

Section	Description
Installer and Uninstaller Command-Line Arguments	Describes the command-line arguments for the installers and uninstallers that InstallAnywhere builds.
Maintenance Mode Command Line Options	Describes the command-line arguments for running Maintenance Mode.
Build Command-Line Arguments	Describes the build and platform arguments for the command-line build.
Launcher Command-Line Arguments	Describes the LAX_VM argument that InstallAnywhere launchers take. All other command-line arguments applied to a launcher are passed to the launched application.

Installer and Uninstaller Command-Line Arguments

InstallAnywhere installers and uninstallers can be run using the following command-line arguments.

Table 7-205 • Command-Line Arguments for InstallAnywhere Installers

Argument	Description
-i	Sets the installer interface mode: silent/console/gui. c:\myinstall.exe -i silent sh ./install1.bin -i console
-f	Sets the location of a response file (installer.properties file) for the installer to use. (See Silent Installers and Response Files .) c:\myinstall.exe -f c:\tmp\installer.properties



Note • This path can be absolute or relative. (Relative paths are relative to the location of the installer.)

Table 7-205 • Command-Line Arguments for InstallAnywhere Installers (cont.)

Argument	Description
-r	<p>Creates a response file. (See Generating Response Files.)</p> <p>c:\myinstall.exe -r c:\temp\myinstaller.properties</p> <p>In the example above, a response file named myinstaller.properties will be written to the c:\temp directory. If you do not enter a path and file name for the response file, the file will be named installer.properties or [installername].properties and it will be created in the same directory as the installer.</p>  <p>Note • Response files can be used to provide input for silent installers.</p>
-D	<p>Passes custom command-line arguments.</p> <p>c:\myinstall.exe -Dmyvar=myvalue</p>
-l	<p>Uses the specified language code (and optional country code) to set the locale for the InstallAnywhere installer. (See Language Codes.)</p> <p>c:\myinstall.exe -l en</p> <p>sh ./install1.bin -l pt_BR</p> <p>The required language code is a two-character (commonly lowercase) code defined by the ISO-639 standard. InstallAnywhere accepts both old (iw, ji, and in) and new (he, yi, and id) language codes.</p> <p>The optional country code is a two-character (commonly uppercase) code defined by the ISO-3166 standard.</p>  <p>Note • Locale options are only respected if the installer includes localizations for the locale you specify.</p>
-? -help	<p>Shows help for the InstallAnywhere installer.</p>  <p>Note • On Windows, -help only works from the console launcher. Make sure to set the LaunchAnywhere to Console on the Windows tab of the Project > Platforms subtask. (For an installed LaunchAnywhere to provide this information, you need to make sure it is explicitly set to Console Launcher on the action.)</p>

Maintenance Mode Command Line Options

To run the Maintenance Mode Launcher via command line, use the following command:

Change *Product_Name* Installation.exe

The Maintenance Mode launcher supports the following command line options:

Table 7-206 • Maintenance Mode Command Line Options

Option	Description
-add	Use to add features. Follow this option with a list of features that you want to add. For example: Change MyProduct Installation.exe -add feature1 feature2
-remove	Use to remove features. Follow this option with a list of features that you want to remove. For example: Change MyProduct Installation.exe -remove feature3 feature4
-uninstall	Use to uninstall a product. Change MyProduct Installation.exe -uninstall
-repair	Use to repair an installed product. Change MyProduct Installation.exe -repair

If no options are specified, then the Maintenance Mode launcher will default to the uninstall mode.



Important • All the above options are mutually exclusive of each other. Therefore, no two of the above options can be used together in a command line executing the Maintenance Mode Launcher.



Important • Maintenance Mode is not supported in silent mode. If you run the Maintenance Mode launcher in Silent mode, the launcher will uninstall the product.



Note • When running Maintenance Mode in console mode, the user cannot go back to the following console mode panels using the BACK command line option:

1. Install/Modify Instance Console
2. Choose Instance to Modify Console
3. Choose Maintenance Option Console

Build Command-Line Arguments

You can invoke InstallAnywhere's installer builder by calling build.exe from the command line and specifying the project that you want to build. For example:

```
build.exe C:\my_setups\MyProduct.iap_xml
```

When building using the command line, you have several options regarding specifying which Build Configurations to build. See [Build Command Line Examples](#) for detailed information and examples.

Information about Build command line arguments is organized into the following sections:

- [Build Arguments](#)
- [Platform Arguments](#)
- [Build Command Line Examples](#)
- [Sample BuildProperties.xml File](#)
- [Sample buildproperties.properties File](#)



Note • For Windows systems, InstallAnywhere provides two versions of the command-line build tool: build.exe and buildasInvoker.exe. Using buildasInvoker.exe is recommended for Windows users who do not have administrative privileges on the build system.

Build Arguments

Using command-line build options, you can also show the InstallAnywhere product version, pass an alternative BuildProperties.xml file, or apply specific platform build settings (see [Platform Arguments](#)).

Table 7-207 • Command-Line Build Arguments

Option	Description
-v	Print the InstallAnywhere product version. build.exe -v
-all	Using the build.exe command without this -all option will build only those Build Configurations that have been added to the project (those that have the Add to project build option selected). If you want to build all of a project's Build Configurations, even those that do not have the Add to project build option selected, use the -all option.



Note • For more information, see [Build Command Line Examples](#).

Table 7-207 • Command-Line Build Arguments (cont.)

Option	Description
-generateHostID	<p>Use to generate a host ID for a machine, which is used to create the license file.</p>  <p>Note • The <code>-generateHostID</code> argument has a known failure if the machine is not connected to a network (but not necessarily the Internet). For more information and an alternate method, see Alternative Method to Generate Host ID.</p>
-p	<p>As an alternative to specifying arguments to the build executable on the command line, you can store the settings in a build properties file, and use the <code>-p</code> switch to point to the desired build-properties file. For example:</p> <pre>build.exe SampleApp.iap_xml -p BuildProperties.xml</pre>  <p>Note • For more information, see Sample BuildProperties.xml File.</p>
-d	<p>Use specified working directory (relative or full path) for the build. This is the directory <code>build.exe</code> uses to store all temporary files: the build log, locales directory, and others.</p> <pre>build.exe MyProduct.iap_xml -d ..\mydir</pre> <p>The default value for the working directory is InstallAnywhere project file directory.</p>  <p>Note • Relative paths are relative to the location of the project file.</p>  <p>Tip • The same result can be achieved by passing a Build Properties file with the working directory set by a <code>BuildWorkdirLocation</code> setting.</p>
-i	<p>Use specified login credentials to access System i (i5/OS) machine. Some System i (i5/OS) installers must bundle resources from an i5/OS machine at build time. Provide the address, user ID, and password for the i5/OS machine:</p> <pre>build.exe MyProduct.iap_xml -i 172.17.1.110/PN113/PASS123</pre>  <p>Note • Address, user ID, and password values are separated by a slash character (/).</p>

Table 7-207 • Command-Line Build Arguments (cont.)

Option	Description
-ls	<p>Registers the command-line build with the specified license server. (This option is only for users who purchase InstallAnywhere under a concurrent license plan.) Provide the location (IP address or hostname) and port (optional) for the license server. For example, if the license server was operating on port 8888 of localhost, you might use:</p> <pre>build.exe -ls 127.0.0.1:8888</pre>  <p>Note • It is possible for the build to fail if the build's license lease expires, if insufficient licenses are available, or if the license server is inaccessible at build time.</p>
+	<p>Add platform to build. (See Platform Arguments, below.) For example, to build for Windows, no VM, use:</p> <pre>build.exe C:\my_setups\MyProduct.iap_xml +W</pre>  <p>Note • Using this option activates all build targets in your InstallAnywhere project for the specified platform/VM combination.</p>  <p>Note • The + and - options can be used for:</p> <ul style="list-style-type: none"> • The entire project (without specifying the -all option). • Platform targets for individual configurations.
-	<p>Remove platform from build. (See Platform Arguments, below.) For example, to suppress the build of Mac OS X, use:</p> <pre>build.exe C:\my_setups\MyProduct.iap_xml -X</pre>  <p>Note • Using this option deactivates all build targets in your InstallAnywhere project for the specified platform/VM combination.</p>  <p>Note • The + and - options can be used for:</p> <ul style="list-style-type: none"> • The entire project (without specifying the -all option). • Platform targets for individual configurations.



Note • You can use several different methods to set the project version at build time. See [Setting Project Version at Build Time](#).

Alternative Method to Generate Host ID

The `-generateHostID` argument has a known failure if the machine is not connected to a network (but not necessarily the Internet). If the machine is not connected to a network, then Java's API (`java.net.InetAddress`) will not be able to generate the Host ID of the machine.

An alternative way to generate the host ID is as follows.



Task: *To generate a machine's Host ID using ipconfig:*

1. Open the command prompt and enter:

```
ipconfig /all
```

Configuration information is listed:

```
Mark C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\P7104903>ipconfig /all

Windows IP Configuration

    Host Name . . . . . : BLR-P7104903
    Primary Dns Suffix . . . . . : nesscorp
    Node Type . . . . . : Hybrid
    IP Routing Enabled. . . . . : No
    WINS Proxy Enabled. . . . . : No
    DNS Suffix Search List. . . . . : nesscorp

Ethernet adapter Wireless Network Connection:

    Media State . . . . . : Media disconnected
    Description . . . . . : Intel(R) WiFi Link 5300 AGN
    Physical Address. . . . . : 00-21-6A-78-FF-FC

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix . . . . . :
    Description . . . . . : Intel(R) 82567LM Gigabit Network Con
nection
    Physical Address. . . . . : 00-24-E8-DC-55-9B
    Dhcp Enabled. . . . . : Yes
    Autoconfiguration Enabled . . . . . : Yes
    IP Address. . . . . : 172.18.24.122
    Subnet Mask . . . . . : 255.255.254.0
    Default Gateway . . . . . : 172.18.24.1
    DHCP Server . . . . . : 192.168.150.21
    DNS Servers . . . . . : 192.168.150.22
                                         192.168.150.21
    Lease Obtained. . . . . : Friday, April 01, 2011 10:28:43 AM
    Lease Expires . . . . . : Friday, April 01, 2011 2:28:43 PM

Ethernet adapter Network Connect Adapter:

    Media State . . . . . : Media disconnected
    Description . . . . . : Juniper Network Connect Virtual Adapter
    Physical Address. . . . . : 00-FF-68-2D-C8-88

C:\Documents and Settings\P7104903>
```

2. Locate the `Ethernet adapter Local Area Connection` section.
3. Copy the entry in the `Physical Address` field, which will be in the format of `00-24-E8-DC-55-9B`.
4. To obtain the Host ID, remove the hyphens from the `Physical Address` and convert the letters to lower case. In this example, the Host ID would be:

0024e8dc559b

Platform Arguments

The following arguments modify the add and remove platforms arguments:

Table 7-208 • Platform Arguments

Argument	Description
a,A	AIX without VM option
av,AV	AIX with VM option
h,H	HP-UX without VM option
hv,HV	HP-UX with VM option
j,J,o,O	Pure Java option
I,L	Linux without VM option
lv,LV	Linux with VM option
s,S	Solaris without VM option
sv,SV	Solaris with VM option
u,U	Generic UNIX without VM option
n,N	Named UNIX without VM option
nv,NV	Named UNIX with VM option
w,W	Windows without VM option
wv,WV	Windows with VM option
x,X	Mac OS X without VM option
web	Build Web installers
cd	Build CD-ROM installers
opt	Optimize by platform
merge	Build Merge modules

Build Command Line Examples

When building via command line, you have several options regarding specifying which Build Configuration files to build:

Table 7-209 • Specifying Build Configurations to Build in Command Line

Example	Description
Build All Build Configurations That Have Been Saved to the Project <pre>build.exe MyProduct.iap.xml build.exe MyProduct.iap.xml +x +l -j -w -u</pre>	If you simply use the build command without the <code>-all</code> option, all of the Build Configurations defined in the project that have their Add to project build option selected will be built. Additional arguments can be added to this command line.
Build All Build Configurations <pre>build.exe MyProduct.iap.xml -all</pre>	If you use the build command with the <code>-all</code> option, all of the Build Configurations defined in the project will be built, even those that do not have the Add to project build option selected.
Specify Which Build Configurations to Build <pre>build.exe MyProduct.iap.xml conf1 conf2 conf3 build.exe MyProduct.iap.xml conf1 +s conf2 -w</pre>	Instead of building all of the Build Configurations saved to the project or all defined Build Configurations, you can specify in the command line exactly which Build Configurations you want to build. To add additional arguments to this command line, you would enter them directly after each Build Configuration name.

Sample BuildProperties.xml File

You can choose to use a Build Properties XML file to build installers for a project. To do this, you use the `-p` argument followed by the path to the Build Properties file:

```
build.exe <projectfile> -p <path to BuildProperties.xml>
```

The settings specified in the Build Properties file would override the build settings in the Build Configuration files that have been added to the project.

A Build Properties file template named `BuildProperties.xml` can be found at:

```
<InstallAnywhere folder>/resource/build/BuildProperties.xml
```

It provides a sample of all possible build settings and can be customized to suit your build requirements.



Note • For more information, see [BuildProperties.xml File](#).

Chapter 7: Reference

InstallAnywhere Ant Task Reference

Sample buildproperties.properties File

You can choose to use a buildproperties.properties file to build installers for a project. To do this, you use the -p argument followed by the path to the buildproperties.properties file:

```
build.exe <projectfile> -p <path to the buildproperties.properties>
```

The settings specified in this properties file would override build settings in the project.

A sample buildproperties.properties file can be found at:

```
<InstallAnywhere folder>/resource/build/buildproperties.properties
```



Note • For more information, see [buildproperties.properties File](#).

Launcher Command-Line Arguments

InstallAnywhere launchers accept command-line arguments, but all arguments except LAX_VM are passed on to the launched application. LAX_VM specifies a Java virtual machine for the launched application and corresponds to the LAX property lax.n1.current.vm.

Syntax

```
<LaunchAnywhere> LAX_VM <full_path_to_java_executable>
```

Example

```
OfficeSuite LAX_VM "C:\Program Files\Java\jre1.6.0_05\bin\java.exe"
```

All arguments passed to a LaunchAnywhere launcher (other than LAX_VM) are stored in the launcher's special \$CMD_LINE_ARGUMENTS\$ variable (which corresponds to the LAX property lax.command.line.args).

InstallAnywhere Ant Task Reference

Ant is a powerful, Java based build tool developed by the Apache Foundation's Jakarta Project. It can be used to control complex build tasks in Java and other development environments. Ant manages specific actions through "tasks" which can either be part of the core Ant distribution or available as extensions.

InstallAnywhere comes with an Ant task to build installers from Ant. The InstallAnywhere Ant task (iaant.jar) is located in your InstallAnywhere application folder:

```
<InstallAnywhere>\resource\build\iaant.jar
```



Tip • To integrate the InstallAnywhere Ant task in an Ant project, remember to set the classpath of the InstallAnywhere Ant task to the location of iaant.jar.



Note • The use of iaant.jar requires Java 1.4 or newer.

Ant uses an XML file to specify the order of tasks for your build process. More information on Ant can be found on the Apache Foundation's Ant Project Web site (<http://ant.apache.org>).

Information about the InstallAnywhere Ant task is organized in the following sections:

- [Task Definition](#)
- [Task Settings](#)
- [Building Multiple Build Configurations](#)
- [Setting Source Path Variables Using the Ant Task](#)
- [Ant Task Parameters](#)
- [Configurations](#)
- [InstallAnywhere Ant Task Example](#)

Task Definition

Add a task definition to your Ant project for the InstallAnywhere Ant task.

```
<taskdef name="buildinstaller" classname="com.zerog.ia.integration.ant.InstallAnywhereAntTask">
  <classpath>
    <pathelement path="<InstallAnywhere>/resource/build/iaant.jar" />
  </classpath>
</taskdef>
```

Task Settings

After defining the task, specify any parameter necessary for the build settings.

```
<buildinstaller IAProjectFile="Project_File_Path"
  IALocation="IA_HOME"
  i50SLogin="hostname_or_ip/userName/password">
  <configuration name="Configuration_Name"
    webenabled="true/false"
    weboptimize="true/false"
    webpagelanguage="en/ja"
    cdenabled="true/false"
    cdoptimize="true/false"
    mergeenabled="true/false"
    mergeoptimize="true/false"
    mergereadonly="true/false">
    <locales>
      <localeSuffix>en</localeSuffix>
      <localeSuffix>de</localeSuffix>
      <localeSuffix>ja</localeSuffix>
      ...
      ...
    </locales>
  <target platform="windows"
```

Chapter 7: Reference

InstallAnywhere Ant Task Reference

```
        outputDir="outputDir"
        buildWithVM="true/false"
        buildWithNoVM="true/false"
        bundledVM="path_to_file" />
<target platform="linux"
        outputDir="outputDir"
        buildWithVM="true/false"
        buildWithNoVM="true/false"
        bundledVM="path_to_file" />
...
...
</configuration>
...
...
</buildinstaller>
```

Replace the `IAlocation` value with the absolute path to your own InstallAnywhere application folder.

Specify the path and file name of the project to build in the `IAProjectFile` parameter.

All other properties are optional. The parameters closely match the properties found in the `BuildProperties.xml` file. The available parameters are explained in [Ant Task Parameters](#).



Important • In previous releases, `buildWithVM`, `BuildWithNoVM`, `outputDir`, and `bundledVM` were subelements of the `<target>` element. Starting with InstallAnywhere 2011, they have been promoted to be attributes of the `<target>` element. Therefore, scripts written for previous releases that contain these items will need to be updated.



Important • If you want to build specific Build Configurations or multiple Build Configurations, follow the instructions in [Building Multiple Build Configurations](#).

Building Multiple Build Configurations

You can build multiple Build Configurations that have been defined in the project by using an external `BuildProperties.xml` / `buildproperties.properties` file and the `-p` command line argument. For example:

```
<buildinstaller
    IALocation="InstallAnywhere Home Location"
    IAProjectFile="C:\Projects\myproject.iap_xml"
    additionalparameter=value
>
    <arg value="-p"/>
    <arg value=<path_ to_ BuildProperties.xml_or_buildproperties.properties_files>/>
</buildinstaller>
```



Note • For more information see [BuildProperties.xml File](#) and [buildproperties.properties File](#).

Setting Source Path Variables Using the Ant Task

You can use Source Paths to specify where payload files should be picked up at build time. Source Paths can be set at build time using an environment variable to specify the Source Path's value.

The InstallAnywhere Ant task extends the Exec Ant task that is part of Ant's core set of tasks. The Exec task allows environment variables to be passed to the command using `<env>` elements. Because the InstallAnywhere Ant task extends the Exec task, it also allows environment variables to be set using `<env>` elements.



Note • For information on the Exec Ant task, see <http://ant.apache.org/manual/Tasks/exec.html>.

For each Source Path that needs to be modified, an environment variable such as `IA_PATH_SOURCE_PATH` needs to be set, where `IA_PATH_SOURCE_PATH` is the name of the source path being referenced.

Example

The following Ant target example sets the Source Path `BILLBOARDS_PATH` to the path `X:\billboards\BillboardProblem` by using the `<env>` element:

```
<env key="BILLBOARDS_PATH" value="X:\billboards\BillboardProblem" />
```

All references to the `BILLBOARDS_PATH` Source Path in the project will be substituted with this path during the build.

```
<target name="build-installer" description="Run Install Anywhere build only." >
  <buildinstaller
    IAlocation="${IA_HOME}"
    failonerror="true"
    IAProjectFile="${IA_PROJECT_FILE}"
  >
    <configuration name="Default Configuration">
      <target platform="Windows"
        outputDir="windows"
        buildWithVM="false"
        buildWithNoVM="true" />
    </configuration>
    <env key="BILLBOARDS_PATH" value="X:\billboards\BillboardProblem" />
  </buildinstaller>
</target>
```

Ant Task Parameters

The following Ant task parameters are available:

- [Build Parameters](#)
- [Platform Options](#)
- [Build Options](#)
- [Installer Options](#)

Chapter 7: Reference

InstallAnywhere Ant Task Reference

Build Parameters

Use the following parameters to control the build process for the InstallAnywhere Ant task.

Table 7-210 • Build Parameters

Attribute	Description	Required
IAProjectFile	The location of the InstallAnywhere project that you want to build.	Yes
IALocation	The location where InstallAnywhere is installed. 	No Note • If IALocation is not specified, the task will search for a copy of InstallAnywhere to run against. If InstallAnywhere is not installed in one of the default locations, then the task checks the InstallAnywhere product registry for a valid location.
i5OSLogin	The host name, user name and password of an iSeries machine in the following format: hostname_or_ip/userName/password	No
propertiesfile	The location of a <code>BuildProperties.xml</code> file. (If used, all other attributes are ignored.)	No
failOnError	Stop the build process if the command exits with a return code other than 0. Defaults to false.	No

Platform Options

Platform options correspond to platform targets on the InstallAnywhere Build Targets tab. By default, the platform options you set in the InstallAnywhere Ant task supersede the corresponding settings in your InstallAnywhere project file and activate or deactivate all targets associated with the platform/VM combination you specify.



Note • Platform options do not activate or deactivate the targets you define in the Ant project's Configuration section. (See [Configurations](#).) The InstallAnywhere Ant task will always build the targets you define in the Configurations section regardless of platform options defined here or in the InstallAnywhere project file.

Table 7-211 • Platform Options

Parameter	Description
BuildLinuxWithVM	True/False
BuildLinuxWithoutVM	True/False

Table 7-211 • Platform Options (cont.)

Parameter	Description
LinuxVMPackLocation	Path
BuildHPUXWithVM	True/False
BuildHPUXWithoutVM	True/False
HPUXVMPackLocation	Path
BuildAIXWithVM	True/False
BuildAIXWithoutVM	True/False
AIXVMPackLocation	Path
BuildSolarisWithVM	True/False
BuildSolarisWithoutVM	True/False
SolarisVMPackLocation	Path
BuildNamedUNIXWithVM	True/False
BuildNamedUNIXWithoutVM	True/False
NamedUNIXVMPackLocation	Path
NamedUNIXTitle	String
BuildWindowsWithVM	True/False
BuildWindowsWithoutVM	True/False
WindowsVMPackLocation	Path
BuildUNIXAll	True/False  Note • This parameter corresponds to Unix (All)/Generic Unix build target on the Build Task in the Advanced Designer.
BuildMacOSX	True/False
WantAuthenticationMacOSX	True/False
WantAuthenticationMacOSXShowGUI	True/False

Chapter 7: Reference

InstallAnywhere Ant Task Reference

Table 7-211 • Platform Options (cont.)

Parameter	Description
BuildPureJava	True/False
OverrideAllPlatformSettings	True/False  Note • When you set <i>OverrideAllPlatformSettings</i> to true, Ant overrides all the platform settings in the InstallAnywhere project with the platform options provided in the Ant task. For example, if your InstallAnywhere project targets Windows and Linux systems and your Ant task adds a Mac OS X target (<code>BuildMacOSX="true"</code>), the Ant task builds installers for Windows, Linux, and Mac OS X. However, if you also set <i>OverrideAllPlatformSettings</i> to true, Ant builds an installer for Mac OS X only.

Build Options

Build options specify build distribution settings. These settings correspond to related command-line build arguments documented in [Build Command-Line Arguments](#).

Table 7-212 • Build Options

Parameter	Description
BuildCDROMInstaller	True/False
BuildWebInstaller	True/False
BuildMergeModule	True/False
BuildReadOnlyMergeModule	True/False
BuildWorkdirLocation	Path  Note • <i>BuildWorkdirLocation</i> overrides the default working directory with a relative or full path. (Relative paths are relative to the location of the project file.) Incidentally, the working directory is also the default build output location. If you use both the <i>BuildWorkdirLocation</i> and <i>BuildOutputLocation</i> options, the <i>BuildOutputLocation</i> takes precedence over the working directory for build output.
BuildOutputLocation	Path
OptimizeCDROMInstaller	True/False

Table 7-212 • Build Options (cont.)

Parameter	Description
OptimizeWebInstaller	True/False
OptimizeMergeModule	True/False
AutoPopulateLabels	True/False
AutoCleanComponents	True/False

Installer Options

Installer options affect the behavior of the installers the project builds.

Table 7-213 • Installer Options

Parameter	Description
InstallerStdErrRedirect	Path
InstallerStdOutRedirect	Path
InstallerValidVMList	String
InstallerInitialHeapSize	Int
InstallerMaxHeapSize	Int
UNIXDefaultUI	Silent/Console/GUI
WindowsDefaultUI	Silent/Console/GUI



Note • In order to redirect InstallAnywhere's build process output to Ant's log, you must specify a redirector as described in the Ant documentation (*Exec task*). For information about using Ant, see <http://ant.apache.org/manual/>.

Configurations

Configurations support InstallAnywhere projects that include multiple build targets for the same platform but use different VMs. Each target in a configuration identifies a build target to add to those that already exist in the InstallAnywhere project file. Ant always builds the targets you define in the InstallAnywhere Ant task's Configuration section.

Table 7-214 • Configuration Elements and Attributes

Tag	Description
configuration	<p>Contains targets that append additional build targets to those already defined in the InstallAnywhere project file.</p> <p>Attributes</p> <p>The following are attributes of the configuration element:</p> <ul style="list-style-type: none"> • name—Name of the build configuration to build. • webenabled—True/False. • weboptimize—True/False. • webpagelanguage—en/ja. • cdenabled—True/False. • cdoptimize—True/False. • mergeenabled—True/False. • mergeoptimize—True/False. • mergereadonly—True/False.
locales	<p>Contains targets that append additional build targets to those already defined in the InstallAnywhere project file.</p> <p>Nested Element(s)</p> <p>The following are nested element(s) of the locales element:</p> <ul style="list-style-type: none"> • localeSuffix—Set to en/ja/de/fr/ar... any of 31 available runtime locales.

Table 7-214 • Configuration Elements and Attributes (cont.)

Tag	Description
target	<p>Contains the settings to define a single build target. Targets must include a platform setting.</p> <p>Attributes of target Element</p> <p>The following are attributes of the target element:</p> <ul style="list-style-type: none"> • platform—Specifies the target platform. Possible values include windows, linux, macosx, solaris, aix, hpx, unix, unixwithvm, and java. • buildWithNoVM—Sets whether or not this target includes a bundled VM. Use true to build an installer without a bundled VM; otherwise, use false. • buildWithVM—Sets whether or not this target includes a bundled VM. Use true to build an installer with a bundled VM; otherwise, use false. • bundledVM—Specifies the VM to bundle with the installer. • outputDir—Specifies the name of the build output directory. <p> Important • In previous releases, <i>buildWithVM</i>, <i>BuildWithNoVM</i>, <i>outputDir</i>, and <i>bundledVM</i> were subelements of the <i><target></i> element. Starting with InstallAnywhere 2011, they have been promoted to be attributes of the <i><target></i> element. Therefore, scripts written for previous releases that contain these items will need to be updated.</p>

InstallAnywhere Ant Task Example

The following is an example of an InstallAnywhere Ant task:

```

<taskdef name="buildinstaller" classname="com.zerog.ia.integration.ant.InstallAnywhereAntTask">
  <classpath>
    <pathelement path="c:\ant\lib\ext\iaant.jar"/>
  </classpath>
</taskdef>
...
<buildinstaller
  IALocation="IA_HOME_LOCATION"
  IAProjectFile="C:\Projects\myproject.iap_xml"
  BuildLinuxWithoutVM="true"
  BuildWindowsWithoutVM="true"
  BuildWebInstallers="true"
  OptimizeWebInstallers="true"
  InstallerStdErrRedirect="C:\console.txt"
/>
<configuration name="MacBuildConfig">
  <locales>
    <localeSuffix>en</localeSuffix>
    <localeSuffix>de</localeSuffix>
    <localeSuffix>ja</localeSuffix>
  </locales>
  <target platform="macosx" buildWithNoVM="true"/>
</configuration>

```

<buildinstaller/>

Custom Code APIs

InstallAnywhere includes several APIs that allow developers to extend its functionality and automate most IDE and GUI testing tasks.

Table 7-215 • InstallAnywhere APIs

API	Description
General API	Encapsulated in com.zerog.ia.api.pub, this API provides a collection of interfaces that support custom code actions, panels, consoles, and rules, as well as access to build settings, build distribution settings, and more. See the com.zerog.ia.api.pub package-summary page in the javadocs.
Product Registry API	A subset of the general, public API, the Product Registry API provides a way to query (read-only) the product registry file (registry.xml) used by InstallAnywhere. The product registry contains information about installed products (version, install date, components, features, and so on). See the package summary page for the Product Registry API (com.zerog.ia.api.pub.registry) for more information.
Services API	This API provides services such as File, Security, System, Win32, Win32 Registry, and Windows Account Privileges. Specific service support and service packages are listed on the javadocs index.html page.
Project Automation API	This API supports the ability to edit projects programmatically. Although this API does support projects with existing references to Merge Modules (dynamic and static), it does not support importing, editing, or removing Merge Modules, nor does it support adding plugins. See Project Automation API for Maintenance Mode and Instance Management and the Project Automation javadocs for more information.  Note • If your project is created with a version of InstallAnywhere prior to InstallAnywhere 2010 SP1 and you want to use the latest Project Automation API to modify the project, it is recommended that you first open and save your project in the InstallAnywhere 2011 Advanced Designer.
Test Automation API	This API is a framework for automated testing of InstallAnywhere GUI installers. See test automation readme.txt and javadocs for more information.



Note • For more information about working with custom code, see [Packaging and Executing Custom Code](#) and [Packaging Custom Code as a Plug-in](#).

Project Automation API for Maintenance Mode and Instance Management

InstallAnywhere provides API support to perform project automation for Maintenance Mode and Instance Management features. Using project automation API, you can perform the following tasks:

Table 7-216 • Project Automation API for Maintenance Mode and Instance Management

Category	Automated Tasks
Maintenance Mode	<ul style="list-style-type: none">Enabling/Disabling Maintenance Mode (<code>setMaintModeSupportEnabled</code>)Choosing Maintenance Mode Options
Instance Management	<ul style="list-style-type: none">Enabling/Disabling Instance Management (<code>setEnableInstanceManagement</code>)Specifying Instance Management Type (<code>setInstanceType</code>)Setting Instance Definition (<code>setInstanceDefinitionBy</code>)Enable Instance Overwrite Warning (<code>setEnableOvertopCheck</code>)
CheckRunningMode Rule	<ul style="list-style-type: none">Associating CheckRunningMode Rules to Individual Actions



Note • If your project is created with a version of InstallAnywhere prior to InstallAnywhere 2010 SP1 and you want to use the project automation API for Maintenance Mode and Instance Management to modify the project, it is recommended that you first open and save your project in the InstallAnywhere 2011 Advanced Designer.

Maintenance Mode Tasks

The Maintenance Mode tasks that you can perform using project automation API are:

- Enabling/Disabling Maintenance Mode (`setMaintModeSupportEnabled`)
- Choosing Maintenance Mode Options

Maintenance Mode methods begin with the `getMaintModeConfigs()` accessor.

Enabling/Disabling Maintenance Mode (`setMaintModeSupportEnabled`)

To enable or disable Maintenance Mode, use the following method:

Chapter 7: Reference

Custom Code APIs

```
project.getMaintModeConfigs().setMaintModeSupportEnabled(<Boolean value>);
```



Note • In this method, the <Boolean value> will be either **true** or **false**.

Choosing Maintenance Mode Options

To choose a Maintenance Mode option, use the following methods:

Table 7-217 • Methods to Choose Maintenance Mode Options

Option	Method
Uninstall Product	project.getMaintModeConfigs().setUninstProductEnabled (<Boolean value>);
Remove Features	project.getMaintModeConfigs().setRemoveFeaturesEnabled (<Boolean value>);
Add Features	project.getMaintModeConfigs().setAddFeaturesEnabled (<Boolean value>);
Repair Product	project.getMaintModeConfigs().setRepairInstallsEnabled (<Boolean value>);



Note • In these methods, the <Boolean value> will be either **true** or **false**.

Instance Management Tasks

The Instance Management tasks that you can perform using project automation API are:

- [Enabling/Disabling Instance Management \(setEnableInstanceManagement\)](#)
- [Specifying Instance Management Type \(setInstanceType\)](#)
- [Setting Instance Definition \(setInstanceDefinitionBy\)](#)
- [Enable Instance Overwrite Warning \(setEnableOvertopCheck\)](#)

Instance Management methods begin with the `getInstanceDefinition()` accessor.

Enabling/Disabling Instance Management (`setEnableInstanceManagement`)

To enable or disable Instance Management, use the following method:

```
project.getInstanceDefinition().setEnableInstanceManagement (<Boolean value>);
```



Note • In this method, the <Boolean value> will be either **true** or **false**.

Specifying Instance Management Type (setInstanceType)

To specify the type of Instance Management to implement, use the following method:

```
project.getInstanceDefinition().setInstanceType(value);
```

In this method, the *value* identifies the Instance Management option that is being selected:

Table 7-218 • Values for the setInstanceType Method

Value	Option
1	Restrict to a Single Instance
2	Do Not Limit Instances
3	Restrict to Number of Instances

Specifying Number of Permitted Instances (setNumInstances)

If the *setInstanceType* above is set to **3** (Restrict to Number of Instances), the number of instances to restrict to can be set using the *setNumInstances* method:

```
project.getInstanceDefinition().setNumInstances(value);
```

In this method, the *value* identifies the number of instances that will be permitted.

Setting Instance Definition (setInstanceDefinitionBy)

The user can specify how to identify an instance of a product by using the *setInstanceDefinitionBy* method:

```
project.getInstanceDefinition().setInstanceDefinitionBy(value);
```

In this method, the *value* identifies the instance definition that is being selected:

Table 7-219 • Values for the setInstanceType Method

Value	Option
10	Define by Installation Location Only
11	Define by Installation Location and Version

Set Instance Definition By Version (setVersionPartIdentifiesInstance)

If you specify **11** (Define by Installation Location and Version) for the setInstanceDefinitionBy method, the version number can be set using the setVersionPartIdentifiesInstance method:

```
project.getInstanceDefinition().setVersionPartIdentifiesInstance(value);
```

In this method, the *value* identifies the version field level that is being selected:

Table 7-220 • Values for the setVersionPartIdentifiesInstance Method

Value	Option
20	Major
21	Minor
22	Revision
23	Sub-Revision



Note • Versions are conventionally represented in the following format: [Major].[Minor].[Revision].[Subrevision], such as: 1.0.2.1047.

Enable Instance Overwrite Warning (setEnableOvertopCheck)

To enable or disable the Enable Instance Overwrite Warning feature, use the following method:

```
project.getInstanceDefinition().setEnableOvertopCheck(<Boolean Value>);
```



Note • In this method, the <Boolean value> will be either **true** or **false**.

Set Overwrite Behavior (setOvertopBehaviour)

If you specify **true** for the setEnableOvertopCheck method, the overwrite behavior can be set using the setOvertopBehaviour method:

```
project.getInstanceDefinition().setOvertopBehaviour(value);
```

In this method, the *value* identifies whether the user will be permitted to overwrite the instance:

Table 7-221 • Values for the setOvertopBehaviour Method

Value	Option
30	Allow user to continue with overwrite
31	Do not allow user to overwrite

Associating CheckRunningMode Rules to Individual Actions

To make sure that individual actions in any of the tasks (Install, Pre-Install, Post-Install, Uninstall, Pre-Uninstall, Post-Uninstall) work with Maintenance Mode, you can associate the CheckRunningMode rule using the following methods:

```
CheckRunningModeRule crmAdd=new CheckRunningModeRule();
crmAdd.setRunningMode((short)value);
```

In this method, the `value` identifies the selected mode:

Table 7-222 • Values for the setRunningMode Method

Value	Mode
1	Add
2	Remove
3	Repair
4	Normal Uninstall
0	Normal Install



Note • To add a rule to the an action, append the above `CheckRunningModeRule` method to the vector of rules associated with the action.

Chapter 7: Reference

Custom Code APIs

Index

Symbols

.jym file [252](#)

A

action execution sequence [31](#)
Action Group action [523](#)
action groups [30](#)
 and Maintenance Mode [153](#)
actions [29](#)
 action execution sequence [31](#)
 action groups [30](#)
 adding to Install task [122](#)
 assigning a rule to [137](#)
 assigning a rule to a group of actions [138](#)
 common customizer properties [575](#)
 common properties [575](#)
 Console [29, 555](#)
 customizers [31](#)
 execution sequence [31](#)
 General [29, 523](#)
 Install [29](#)
 overview [29](#)
 Panel [29, 545](#)
 Plug-In [571](#)
 Properties tab [31](#)
 Rollback tab [31](#)
 Rules tab [31](#)
 System i (i5/OS) [556](#)
 Tags tab [31](#)
 Trigger Rollback [288](#)
 types of [29](#)

Add Comment [523](#)
Add Features [152, 361](#)
Add Files [447](#)
Add Jump Label [523](#)
Additions to GUI Installer Panels [43](#)
Advanced Designer [21, 81, 285, 308, 593](#)
 adding files [99](#)
 adding files to a project [121](#)
 adding LaunchAnywhere executable [101](#)
 adding Post-Install actions [105](#)
 adding Pre-Install actions [96](#)
 Build task [424](#)
 building the installer [110](#)
 Install task [98, 412](#)
 Installer UI task [373](#)
 new project [93](#)
 Organization task [391](#)
 organizing features and components [128](#)
 Post-Install task [105, 417](#)
 Post-Uninstall task [423](#)
 Pre-Install task [96, 411](#)
 Pre-Uninstall task [419](#)
 Project task [312](#)
 switching to [22](#)
 task overview [91](#)
 tasks [22](#)
 testing the installer [112](#)
 tutorial [91](#)
 Uninstall task [420](#)
Advertise Variables [201](#)
advertising
 merge module installer variables [201](#)
Ant [523, 650](#)

build integration 650
setting source path variables 653
task reference 650

APIs 660
documentation 660
project automation 661

Automatically Set Classpath 449

B

background images 43
installer mode 42
bidi orientation 33
Billboards 43, 387
Browse for Folder dialog box 604
Build All 205
Build Configurations
about 180
adding to project build 187
assigning Tags to project elements 191
associating Tags with 192
benefits of using 180
building 203
building all 204
building via command line 204
copying 186
creating 179
creating for migrated project 184
creating new 184
creating new Tags 191
default for new and migrated projects 180
editing 179
Migrated Configuration 184
options 181
overview 179
removing 187
renaming 185
searching for Tags 193, 487
specifying locale settings 194
using Tags to customize 188
Build Configurations tab 181
Build Installer panel 450
Build Log 441
Build task 110, 178, 424
Build Configurations tab 181
Build Log subtab 441
building installers using Build Configurations 203
creating CD-ROM/DVD installers 199
creating installer for customized flavor of Unix 198
creating merge modules 200
creating Web installers 199

defining targets 194
overview 178
setting build distribution options 199
specifying locales 59
build tool 36
build.exe 36, 204
command line build 204
command-line arguments 644
buildproperties.properties file 636
compared to settings in Advanced Designer 636
BuildProperties.xml file 629
compared to settings in Advanced Designer 629

C

CD-ROM installers
creating 199
Change Product_Name Installation.exe 164
Check File/Folder Attributes rule 142, 585
Check for updates automatically 219
Check If File/Folder Exists rule 587
Check Platform rule 143, 587
examples 143
Check Running Mode rule 144, 153, 588
about 159
and Maintenance Mode 159
Check System Architecture Rule 584
Check User-Chosen Language Rule 584
Choose Alias, Link, Shortcut 545
Choose Features to Uninstall 545, 555
Choose File 545
Choose Folder 546
Choose Install Folder 546, 555
Choose Install Sets 549, 555
Choose Java VM 550, 555
Choose Java VM panel 55
result of including 54
result of not including 53
Choose Link Folder 555
Choose Main 448
Choose Remote System i (i5/OS) Install Folder 558
Choose Uninstall Type 546, 555
Classpath 575
classpath 20, 88
Clean Components 132
ClearCase 214
color settings 33
command 261, 641
command line 261, 641
command-line build 36, 204, 304
arguments 644

examples 649
 Installer and Uninstaller arguments 641
 Maintenance Mode options 643
 platform arguments 648
 reference 641
 command-line launcher
 arguments 650
 Comment 523
 Compare File Modification Timestamp Rule 584
 Compare InstallAnywhere Variable Numerically rule 147
 Compare InstallAnywhere Variables Numerically rule 592
 Compare InstallAnywhere Variables Rule 584
 Compare InstallAnywhere Variables Rule Numerically Rule 584
 components 40, 41, 128, 129, 132, 394
 adding 129
 adding files to 132
 assigning components to features 135
 best practices 129
 dependency 130, 131
 integrating with target 133
 key file 131
 organizing 128
 removing empty components 132
 shared 130
 standard 130
 types 130
 Config 339
 Config subtask 289
 console 261, 641
 Console actions 29, 555
 console installers 42, 44
 consoles 555
 Copy File 497, 523
 Copy Folder 497, 524
 Create Alias, Link, Shortcut 499
 Create Alias, Link, Shortcut to DIM File 500
 Create DIM Reference 501
 Create Folder 497, 503
 Create LaunchAnywhere for Java Application 505
 Create LaunchAnywhere for Java Application action 53
 Create LaunchAnywhere for Java Apps 497
 Create Uninstaller 497, 507
 Create VM Pack 467
 Custom Code 280, 286, 523, 555
 custom code
 importing InstallShield MultiPlatform APIs into
 InstallAnywhere 283
 Custom Code API 37, 38, 68
 and variables 38
 overview 37
 plug-ins 38
 Custom Code Panel 546
 customizers 31
 Properties tab 31
 Rollback tab 31
 Rules tab 31
 Tags tab 31
 Customizing a Check File/Folder Attributes rule
 example 142
 Customizing a Check If File/Folder Exists rule 143
 CVS 214

D

Debug 280, 289, 292, 299, 523
 Debugging 293, 294, 295, 296
 Delete File 497, 524
 Delete Folder 497, 530
 dependencies
 support for signed JARs 282
 Deploy WAR/EAR Archive 509
 developer installation manifests (DIMs) 214
 Digital certificates 287
 DIM references 39, 214
 adding 214
 creating a shortcut to a file in a DIM 218
 customizing 216
 removing 217
 Disable WOW64 Redirection 372
 Disk Space Check 546
 Display HTML 552
 Display Message 546, 555
 Distribution 435
 distribution options
 setting 199
 Do not limit instances 364
 Dock
 Mac OS X 296
 Download Additional VM Packs 496
 DVD installers 47
 creating 199
 dynamic text 59

E

editions 26
 Enable Instance Management 363
 Enable Instance Overwrite Warning 163, 366
 Enable Maintenance Mode Support 360
 Enable Rollback 287
 Enable Source Paths 212

Enable Update Notifications 497, 510
Enable WOW64 Redirection 372
Encryption 225
Enterprise Edition 293, 481
Enterprise edition 26, 43
Evaluate Custom Rule rule 146, 590
Execute Ant Script 523, 531
Execute Command 523, 532
Execute Custom Code 533
Execute Script/Batch File 523, 534
Execute Target File 523, 525
Execute Target File action 211
Execute Uninstaller 525
exit codes 304
Expand Archive 497, 514
External Resource Bundle 279, 280
external resource bundles 60
 file format 61
 naming conventions 61
 support for Merge Modules in Install and Uninstall phases 61

F

Features 393
features 40
 adding 134
 assigning files to 134
 assigning to components 135
 organizing 128
 uninstalling 69
files 622
 adding to a project 85, 121
 assigning to components 132
 assigning to features 134
 file formats 622
 reference 622
 updating location of 213
Find Component in Registry 133, 523, 535
Find File/Folder 547
FlexNet Connect 218
folders
 SpeedFolders 67
font settings 33
fonts 211
 installing 211

G

General Actions 523
General actions 29

Get Password 547, 555
Get Serial Number 547, 555
Get System i (i5/OS) Login Credentials 559
Get User Input 556
Get User Input - Advanced 547
Get User Input - Simple 547
Get User Input panels 32, 579
 Bidi orientation 33
 color settings 33
 component types 32
 defaults 35
 font settings 33
 input methods 32
 results variables 33
 text reading order 33
 VAR_BOOLEAN_X variable 36
Get Windows Registry Entry 523, 525
graphical user interface (GUI) installers 42
 background images 43
 billboards 43
 localization 43
 look and feel 43
 panel additions 43
 splash screen 43
gui 261

H

help
 Help Library conventions 16
 using 16
Help menu
 Download Additional VM Packs 496
hosts 39
 application server 39
 database server 39
 operating system 39

I

Identify Instance Using 365
Image Settings 576
Important Note 547
Install actions 29, 497
 adding 122
Install Archive 497, 517
Install Complete 547, 556
Install Failed 556
Install File 497, 517
Install from Manifest 497, 498
Install HP-UX Depot 497, 498

Install Linux RPM 497, 498
 Install Log File
 content 624
 format 624
 naming conventions 625
 Install Merge Module 497, 518
 Install Sets 40, 392
 Install Solaris Package 497, 498
 Install SpeedFolder 497, 498
 Install task 84, 98, 119, 122, 412
 adding a Trigger Rollback action 288
 adding LaunchAnywhere executables 123
 installing fonts 211
 install.exe 261, 641
 InstallAnywhere
 Advanced Designer 21
 Ant task reference 650
 authoring environments 19
 authoring modes 81
 build tool 36
 creating new project 23
 Custom Code API 37
 custom code APIs 660
 editions 26
 hosts 39
 installing updates 219
 opening a project 23
 opening existing project 24
 product registry 623
 Project Wizard 20
 projects 65
 rules 65
 setting default authoring environment 23
 source paths 66
 switching to a different project 25
 switching to Advanced Designer interface 22
 tutorials 81
 variables 38, 72, 73, 74, 593
 InstallAnywhere Launcher
 specifying 32-bit or 64-bit JVM 324
 InstallAnywhere Merge Module Import Assistant dialog box 474
 InstallAnywhere Variable
 checking value of 225
 installer modes 42
 console 42
 graphic user interface (GUI) 42
 silent 42
 Installer Panel Additions 576
 Installer UI task 373, 576
 Installer Valid VM List 51

installers
 assigning a rule to 137
 building 178
 console 44
 creating basic 115, 116
 defining rules 136
 graphical user interface (GUI) 42
 minimizing size of 210
 optimizing for Unix 210
 testing 205
 InstallShield MultiPlatform
 calling ISMP APIs in InstallAnywhere 283
 Instance Management 159
 Do not limit instances 161
 Identify Instance Using 162
 identifying new instance by version 367
 options on Project > Advanced subtask 362
 Restrict to a Single Instance 161
 Restrict to Number of Instances 161
 specifying options 160
 Version Field Level That Identifies New Instance 162
 Introduction 547

J

JARs
 support for signed JARs 282
 Java virtual machines. See JVMs.
 javadocs 660
 Jump to Target 523, 525
 JVM search instructions file 252
 JVM Settings subtask 337
 JVM Spec File 252
 JVMs 48, 49
 about Java VM selection criteria 56
 Create LaunchAnywhere for Java Application action 53
 criteria used in JVM search 52
 how a launcher selects a VM 51
 launcher's VM selection behavior at run-time 52
 No Specific VM option 55
 selecting 49
 selection by LaunchAnywhere launchers 49
 specifying 32-bit or 64-bit 324
 using first VM found matching search settings 54
 VM selected by the installer 53
 VM selected on the Choose Java VM panel 53
 VM Used by the Installer option 54
 when VM packs are installed 55
 when VM searches occur 50
 where VM searches occur 51

K

key file [131](#)

L

Label Settings [576](#)
language code [261, 641](#)
language codes [615](#)
Launch Default Browser [523, 526](#)
LaunchAnywhere [58, 74, 208, 289, 293, 593, 606](#)
 command line parameter [58](#)
 command-line arguments [650](#)
 overview [58](#)
 properties [606](#)
LaunchAnywhere executable
 adding [101](#)
LaunchAnywhere executables [293](#)
 adding [123](#)
Launcher JVM Selection [324](#)
launchers
 creating [208](#)
launchers for Java Applications
 creating [208](#)
LAX
 LaunchAnywhere executable [208, 293, 606](#)
 properties [606](#)
License Agreement [556](#)
License Agreement (Panel action) [548](#)
line [261, 641](#)
List of Installer Steps [576](#)
locales [43, 273, 274, 286, 299, 333](#)
 language codes [615](#)
 specifying in Build Configurations [194](#)
Locales subtab [59](#)
localization [43, 58, 59, 279, 280](#)
 basics of [59](#)
 best practices [60](#)
 common localizable elements [617](#)
 dynamic and static text [59](#)
 external resource bundles [60](#)
 language codes [615](#)
 locales folder location [59](#)
 localizable elements [617](#)
 overview [58](#)
 reference [615](#)
Log Settings subtask [356](#)

debugging [296](#)
installer [296](#)
troubleshooting [296](#)
Magic Folders [61, 73, 296, 611](#)
 and variables [73](#)
 overview [61](#)
 variables [611](#)
main class [20, 87](#)
Maintenance Mode
 action groups [153](#)
 Add Features [360](#)
 Add Features user experience [167](#)
 adding a Check Running Mode rule [144](#)
 command-line build options [643](#)
 configuring [150](#)
 creating new Tags on Project > Advanced subtask [191](#)
 Enable Maintenance Mode support [360](#)
 enabling [150, 151](#)
 end user experience [165](#)
 Instance Management options [159, 362](#)
 launcher [164](#)
 Maintenance Mode launcher compared to Uninstaller [164](#)
 Manage Instances dialog box [171](#)
 Manage Tags [369](#)
 options [151](#)
 options on Project > Advanced subtask [360](#)
 overview [149](#)
 Pre-Install task [154](#)
 Pre-Uninstall task [156](#)
 Remove Features [360](#)
 Remove Features user experience [168](#)
 Repair Installation [360](#)
 Repair Product user experience [169](#)
 Uninstall Product [360](#)
 Uninstall Product user experience [170](#)
 when multiple installed instances [171](#)
Manifest File Format [625](#)
Manifest files [625](#)
 format [625](#)
Match Regular Expression Rule [584](#)
merge modules
 advertising merge module installer variables [201](#)
 benefits of using [62](#)
 creating [200](#)
 External Resource Bundles
 support in Install and Uninstall phases [61](#)
 integrating into projects [64](#)
 methods to add to existing installer [63](#)
 overview [62](#)
 uninstalling [69](#)

M

Mac OS X [296](#)

using third party merge modules 63
 Modify Text File - In Archive 523, 537
 Modify Text File - Multiple Files 523, 538
 Modify Text File - Single File 523, 539
 Move File 497, 498
 Move Folder 497, 498

N

Native vs. Swing resources 604

O

Optimize registry entry 130, 396
 Organization task 129, 391
 Components subtask 131, 135
 Features subtask 134
 Output Debug Information 523, 527
 Output Text to Console 523, 527

P

Panel actions 29, 545
 settings 576
 Perforce 214
 Perform XSL Transform 523, 527
 Perform XSL Transform - In Archive 523, 528
 platform 297
 Platforms 320
 Plug-In Actions 571
 plug-ins 38
 Post-Install task 105, 126, 417
 customizing 126
 Post-Install vs. Post-Uninstall 71
 Post-Uninstall task 423
 Pre-Install actions 96
 Pre-Install Summary 556
 Pre-Install Summary Panel 553
 Pre-Install task 118, 411
 customizing 118
 Pre-Install vs. Pre-Uninstall 71
 Pre-Uninstall task 419
 product registry 623
 productVersion 264
 project 65
 adding files 85
 choosing main class 87
 creating new 23, 82
 defining Install task 84
 opening 23
 opening existing 24

switching to a different project 25
 project automation API 661
 project file 65
 Project task 201, 289, 312, 313, 339, 481
 Advanced subtask 359, 360, 362, 369, 370, 371
 Config subtask 289
 Description subtask 316
 Info subtask 201
 JVM Settings subtask 337
 Log Settings subtask 356
 Rules subtask 137, 335
 Variables subtask 353
 Project Wizard 20, 81, 449
 Add Files panel 85
 Build Installer frame 20
 Build Installer panel 89, 450
 building an installer 82, 89
 Choose Main panel 87
 choosing main class 87
 classpath 20
 Classpath panel 88
 creating a new project 82
 defining Install tasks 84
 Try Installer panel 90
 tutorial 82
 ProjectLocalizationInfo.txt 59
 projects 65
 adding components 129
 adding features 134
 adding files to 121
 creating 117
 defining Install task 119
 defining rules 136
 enabling installation rollback 287
 opening 117
 opening existing 117
 opening from Advanced Designer 118
 opening from Project Wizard 117
 organizing features and components 128
 setting the classpath 88
 testing the installer 90

Q

Query InstallShield Universal Software Information 528

R

Read/Modify XML File 528, 541
 Ready to Install 556
 Refresh Windows Environment 528

Register Windows Service [497](#), [528](#)

registry [623](#)

Remove Features [152](#), [361](#)

removing

empty components [132](#)

Repair Installation [153](#), [361](#)

resources

updating location of [213](#)

response file [261](#)

response files [45](#), [627](#), [641](#)

Restart Windows [523](#), [528](#)

Restrict to a Single Instance [364](#)

Restrict to Number of Instances [364](#)

rollback options [287](#), [370](#)

Enable Rollback [287](#)

Rollback Settings [287](#)

Rollback tab [288](#)

Trigger Rollback action [288](#)

Rollback Settings [370](#)

rules [65](#), [335](#)

applying multiple [139](#)

assigning to a group of actions [138](#)

assigning to an action [137](#)

assigning to installer [137](#)

Check File/Folder Attributes [142](#), [585](#)

Check If File/Folder Exists [587](#)

Check Platform [143](#), [587](#)

Check Running Mode [144](#), [159](#), [588](#)

Compare InstallAnywhere Variable Numerically [147](#)

Compare InstallAnywhere Variables Numerically [592](#)

Customizing a Check If File/Folder Exists [143](#)

customizing rules [141](#)

defining [136](#)

Evaluate Custom Rule [146](#), [590](#)

how rules are evaluated during an installation [148](#)

list of [583](#)

Rules subtask [335](#)

Run SQL Script [521](#)

S

SCM [214](#)

Scrolling Message [548](#)

self-extractors [47](#)

behavior on non-Windows platforms [47](#)

behavior on Windows [47](#)

Set Classpath [449](#)

Set InstallAnywhere Variable - Multiple Variables [523](#), [528](#)

Set InstallAnywhere Variable - Single Variable [523](#), [529](#)

Set System Environment Variable [497](#), [499](#)

Set Windows Registry - Multiple Entries [497](#), [529](#)

Set Windows Registry - Single Entry [497](#), [529](#)

setup.exe [261](#), [641](#)

Show Message Console 'Dialog' [556](#)

Show Message Dialog [523](#), [544](#)

signcode [287](#)

signing [287](#)

silent [261](#), [641](#)

silent installer mode [42](#)

silent installers [45](#)

response files [45](#)

source control management software [214](#)

Source Paths [66](#), [211](#), [212](#)

adding and removing [212](#)

enabling [212](#)

predefined [66](#)

substitution [66](#)

SpeedFolders [67](#), [497](#)

splash screen [43](#)

Standard edition [26](#), [43](#)

Start, Stop, Pause Windows Service [529](#)

static text [59](#)

stderr [208](#), [289](#), [293](#), [294](#), [575](#), [606](#)

stdout [208](#), [289](#), [293](#), [294](#), [575](#), [606](#)

Strict VM Selection [57](#)

Substitute Recursively [356](#)

Swing vs. Native resources [604](#)

System i (i5/OS) Actions [556](#)

System i (i5/OS) Command [560](#)

System i (i5/OS) Find Component in RAIR [561](#)

System i (i5/OS) Install File [562](#)

System i (i5/OS) Integrated File System (IFS) [563](#)

System i (i5/OS) Library [564](#)

System i (i5/OS) Licensed Program [565](#)

System i (i5/OS) Licensed Program Exists Condition Rule [584](#)

System i (i5/OS) Object [567](#)

System i (i5/OS) Primary Language Install Condition Rule [585](#)

System i (i5/OS) Process Remote Install (Merge Modules) [568](#)

System i (i5/OS) Program [569](#)

System i (i5/OS) Program Temporary Fix (PTF) [570](#)

System i (i5/OS) Program Temporary Fix (PTF) Condition Rule [585](#)

T

Tag Search Results dialog box [487](#)

Tags

assigning to project elements [191](#)

associating with Build Configurations [192](#)

creating 191
 Manage Tags options on Project > Advanced subtask 369
 searching for 193, 487
 using to customize Build Configurations 188

targets
 defining 194

task
 Billboards 387
 Build 424
 Build Log 441
 Build Targets 428
 Config 339
 Distribution 435
 File Settings 318
 Help 390
 Info 313
 Installer UI 373
 Locales 333
 Organization 391
 Platforms 320
 Project 312

team development 211

Templates 280

templates 67
 overview 67

testing 90, 112, 205
 installers for other platforms 205

text
 dynamic and static 59

Trigger Rollback 288, 521

Troubleshooting Installers 289

Try Installer 205, 451

Try Web Install 205

tutorials 81
 Advanced Designer 91
 Project Wizard 82

U

Uninstall AIX Entries 176
 Uninstall Categories 174
 Uninstall Category 176, 522
 Uninstall Complete 548, 556
 Uninstall DB Scripts 176
 Uninstall Files 176, 522
 Uninstall Folders 176, 522
 Uninstall InstallShield Universal Software 529
 Uninstall JEE Archives 176
 Uninstall LaunchAnywhere 176, 522
 Uninstall Merge Modules 523

Uninstall Product 153, 362
 Uninstall RAIR Entries 176
 Uninstall Registry Entries 176, 523
 Uninstall RPM Packages 176
 Uninstall Shortcuts/Links/Aliases 176, 523
 Uninstall task 420
 about customization 172
 adding Uninstall Categories and Actions 174
 Advanced Designer 420
 assigning a rule to Uninstall Category 177
 assigning actions to features and components 176
 benefits of customizing 172
 customizing 171
 overview 171, 173
 preventing an Uninstall Category from being uninstalled 177
 reordering Uninstall Categories 178

Uninstaller 69, 497, 576
 about 164
 custom code 68
 feature uninstall 69
 features 69
 InstallAnywhere variables 68
 merge modules 69
 multiple merge modules 69
 multiple products 69
 overview 68

Uninstaller Introduction 548, 556

Unix
 creating installer for customized flavor of Unix 198
 debugging 295
 optimizing installer size 210

updates
 checking automatically 219
 manually checking for 219
 update notifications 218, 219

user interface
 setting default 23

V

VAR_BOOLEAN_X variable 36
 variables 38, 72, 73, 223, 292, 593
 advertising merge module installer variables 201
 and Magic Folders 73
 comparing using Compare InstallAnywhere Variables Numerically rule 147
 evaluate at assignment 75
 evaluating 75
 IA_BROWSE_FOLDERS 604
 LAX properties 606

Magic Folders [611](#)
methods of setting [74](#)
notation [72](#)
setting [75](#)
standard InstallAnywhere [593](#)
Substitute Recursively option [356](#)
Variables subtask [353](#)
version
 setting at build time [263, 264](#)
Version Field Level That Identifies New Instance [365](#)
VM packs
 downloading [206](#)
 installing [207](#)
 overview [206](#)
VM Type [57](#)
VM Vendor [56](#)
VM Version [57](#)

W

Web installers [47](#)
 creating [199](#)
Win32 installer [294](#)
Windows Console [208](#)
Windows Execution Level [548](#)
Windows NT [294](#)
 troubleshooting [294](#)
Windows WOW64 Emulator Settings [371](#)
 InstallAnywhere Win32 Services [371](#)
 WOW64 redirection [371](#)
Wizard [448, 495](#)
WOW64
 emulator settings [371](#)