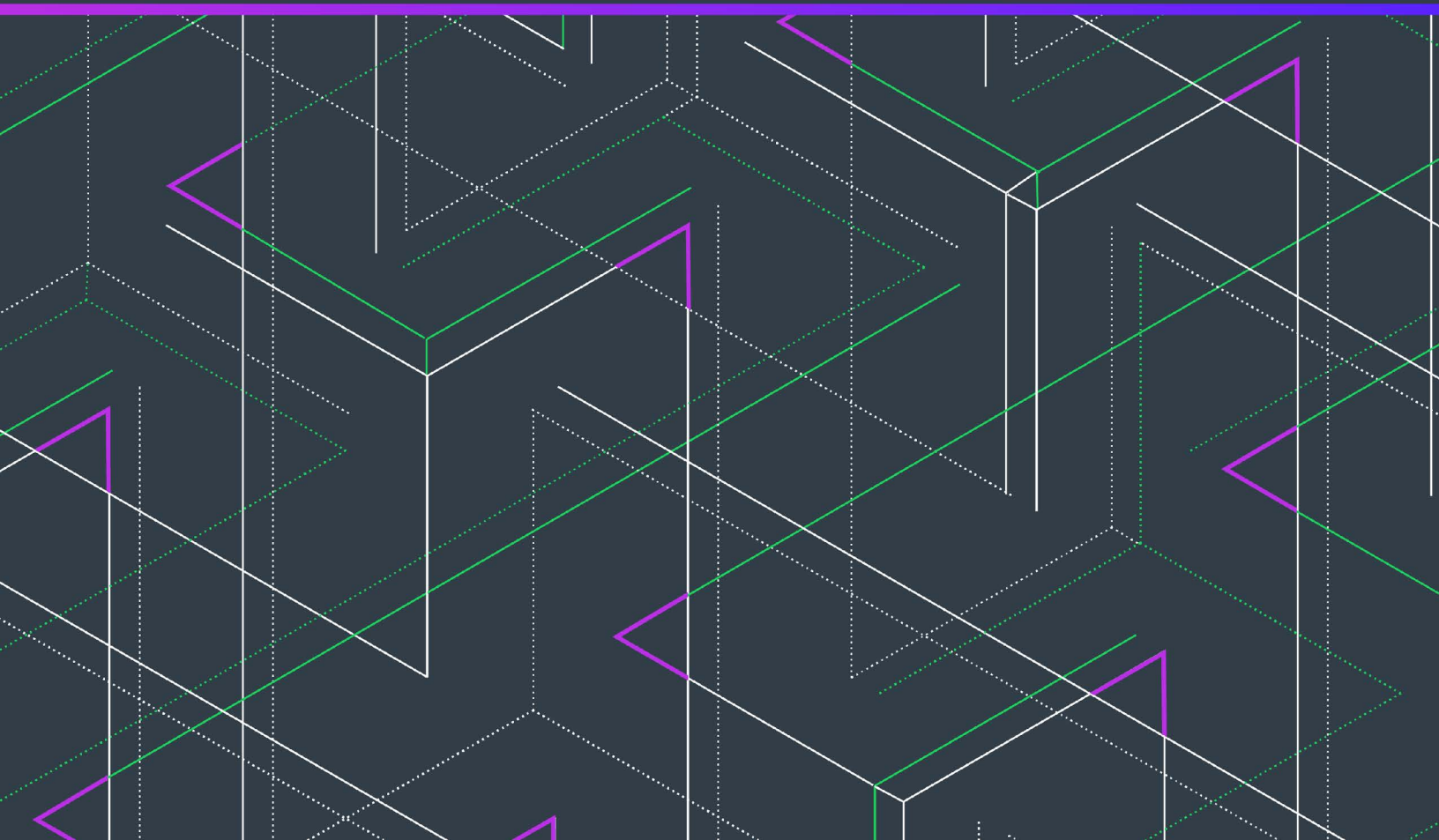
revenera™

InstallAnywhere 2026 R1

User Guide



Legal Information

Book Name: InstallAnywhere 2026 R1 User Guide

Part Number: IA-2026-UG00

Product Release Date: May 2026

Copyright Notice

Copyright © 2026 Flexera Software. All Rights Reserved.

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

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

Intellectual Property

For a list of trademarks and patents that are owned by Flexera Software, see https://www.revenera.com/legal/intellectual-property.html. All other brand and product names mentioned in Flexera Software products, product documentation, and marketing materials are the trademarks and registered trademarks of their respective owners.

Restricted Rights Legend

The Software is commercial computer software. If the user or licensee of the Software is an agency, department, or other entity of the United States Government, the use, duplication, reproduction, release, modification, disclosure, or transfer of the Software, or any related documentation of any kind, including technical data and manuals, is restricted by a license agreement or by the terms of this Agreement in accordance with Federal Acquisition Regulation 12.212 for civilian purposes and Defense Federal Acquisition Regulation Supplement 227.7202 for military purposes. The Software was developed fully at private expense. All other use is prohibited.

Contents

1 InstallAnywhere 2026 R1 Help Library 19

What Was New in Earlier Versions of InstallAnywhere 20

What's New in InstallAnywhere 2025 R1 21
New Features in InstallAnywhere 2024 R2 21
New Features in InstallAnywhere 2024 R1 21
What's New in InstallAnywhere 2023 R1 23
What's New in InstallAnywhere 2022 R2 24
What's New in InstallAnywhere 2022 28
What's New in InstallAnywhere 2021 SP2 30
What's New in InstallAnywhere 2021 SP1 32
What's New in InstallAnywhere 2021 32
What's New in InstallAnywhere 2020 SP2 33
What's New in InstallAnywhere 2020 SP1 35
What's New in InstallAnywhere 2020 36
What's New in InstallAnywhere 2018 SP1 37
What's New in InstallAnywhere 2018 40
What's New in InstallAnywhere 2017 SP1 55
What's New in InstallAnywhere 2017 60
What's New in InstallAnywhere 2015 SP1 63
What's New in InstallAnywhere 2015 64
What's New in InstallAnywhere 2014 SP1 68
What's New in InstallAnywhere 2014 69
What's New in InstallAnywhere 2013 73

Using Help 82

Contacting Us 83

2 Getting Started 85

Introduction to the InstallAnywhere Interface 85

About the InstallAnywhere Advanced Designer Authoring Environment 85

Working with InstallAnywhere Projects 86
Creating a New InstallAnywhere Project 86
Opening an Existing InstallAnywhere Project 88
Switching to a Different Project 88
Upgrading from Earlier InstallAnywhere Versions 89
Obtaining Updates for InstallAnywhere 96

3 Concepts of InstallAnywhere Installer Development 99
Actions 100
About Actions 101
About Action Groups 102
Action Customizers and the Action Execution Sequence 102
About Action Customizers 102
About the Action Execution Sequence 103
About Get User Input Panels 103
Input Methods/Component Types 104
Bidi Orientation and Text Reading Order 105
Font and Color Settings 105
Results Variables 105
Defaults 107
VAR_BOOLEAN_X Variable 108
Mandatory Field 108
Build Tools 109
Custom Code 110
Custom Code Basics 110
About Custom Code and Variables 111
About Plug-ins 112
Hosts 112
Support for Application Servers 112
Support for Database Servers 113
Install Sets, Features, and Components 113
Install Sets 114
Features 114
Components 114
Standard Components 115
Dependencies 115
Shared Components 115
Installer Modes 115
Graphical User Interface (GUI) Installers 116
GUI Localization 116
Customization of the User Interface 116
Console Installers 118
Silent Installers 119
About Response Files and Silent Installers 120
Installer Types 121

Java Virtual Machines **122**
About JVM Selection 122
How the Installer's Launcher Selects a JVM 125
About the Installer's JVM Search 125
About the Launcher's JVM Selection Behavior at Run-Time 126
About the Choose Java VM Panel 129
When Are VM Packs Installed? 129
Configuring Upgrade Installers that Use a Different JRE from the Previous Installer 130
About Java VM Selection Criteria 130
LaunchAnywhere **132**
Localization **132**
Dynamic and Static Text 132
Best Practices for Localizing 133
External Resource Bundles 133
Magic Folders **134**
Merge Modules **135**
Project File **137**
Rules **138**
Source Paths **138**
SpeedFolders **139**
Templates **139**
Uninstaller **139**
About Uninstallers and Custom Code 140
About Uninstallers and Variables 140
Feature Uninstallation 140
Uninstaller for Merge Modules and Multiple Products 140
Similarities Between the Sequences in an Installer and an Uninstaller 142
Variables **143**
Variable Notation 143
Magic Folders and Variables 143
Methods of Setting InstallAnywhere Variables 144
Evaluation of InstallAnywhere Variables 144
Searching for InstallAnywhere Texts 145
Selecting Variables from a List 147

4 InstallAnywhere Tutorial **149**
Creating a New Project **149**
Adding Pre-Install Actions **150**
Defining the Install Sequence **151**
Adding a LaunchAnywhere Executable to the Install Sequence **152**
Adding Post-Install Actions **153**
Building the Installer **155**
Testing the Installer **156**

5 Creating Basic Installers 157

Working with InstallAnywhere Projects 158
Creating a New Project 158
Opening an Existing Project 159

Specifying Installer Information 159
Configuring General Information 159
Specifying the Product Version 159
Configuring Run-Time Logging Preferences 160
Including a Software Identification Tag for Your Product 160

Organizing Files for Your Installer 162
Best Practices for Components 162
Adding Components 162
Adding Folders to a Project 163
Adding Files to a Project 163
Assigning Files to Components 164
Removing Empty Components 164
Integrating Components that Are Already Installed on Target Systems 164
Adding Features 165
Assigning Files to Features 165
Assigning Components to Features 165
Installing Fonts 166
Expanding Archive Files on the Target System 166
Working with Source Paths 168
Enabling Source Paths 168
Adding and Removing Source Paths 168
Updating the Location of Files and Resources 169

Customizing Sequences 170
Customizing the Pre-Install Sequence 170
Customizing the Install Sequence 170
Overview on Defining the Install Sequence 171
Adding Actions to the Install Sequence 172
Adding LaunchAnywhere Executable Files to the Install Sequence 172
Re-creating the InstallAnywhere Uninstaller and Associated Components 173
Customizing the Post-Install Sequence 174
Customizing the Uninstall Sequence 175
About Uninstaller Customization 175
Adding Uninstall Categories and Actions to the Uninstall Sequence 176
Assigning Actions in the Uninstall Sequence to Product Features and Components 177
Preventing the Actions in an Uninstall Category from Being Uninstalled 177
Reordering Uninstall Categories and Actions in the Uninstall Sequence 178

Defining Rules and Rule Expressions that Evaluate Conditions on Target Systems 178
Assigning a Rule to the Installer 179
Assigning a Rule to an Action 179
Assigning a Rule to a Group of Actions 180
Building Complex Rule Expressions 181

Using the Rules Manager to Create Complex Rule Expressions at the Project Level . . . 183
Configuring and Saving a New Rule Expression . . . 183
Associating a Rule Expression to a File Extension . . . 184
Loading a Rule Expression . . . 184
Deleting a Saved Rule Expression . . . 185
Customizing Built-in Rules . . . 185
Customizing a Check File/Folder Attributes Rule . . . 185
Customizing a Check If File/Folder Exists Rule . . . 186
Customizing a Check Platform Rule . . . 187
Customizing a Check Running Mode Rule . . . 188
Customizing Evaluate Custom Rule Rules . . . 189
Customizing a Compare InstallAnywhere Variable Numerically Rule . . . 190
How Rules Are Evaluated at Run Time . . . 190
Configuring Servers . . . 191
Managing Application Servers . . . 191
Adding an Application Host to Your Project . . . 192
Specifying Which Deployment Options to Support for Apache Tomcat Servers . . . 192
Specifying Which Deployment Options to Support for IBM WebSphere Servers . . . 194
Deploying Web Applications (WAR and EAR Archives) to Servers . . . 195
Enabling End Users to Specify Apache Tomcat Server Information . . . 195
Enabling End Users to Specify IBM WebSphere Server Information . . . 196
Managing Database Servers . . . 197
Adding a SQL Database Host to Your Project . . . 198
Including JDBC Drivers for Connecting to SQL Databases . . . 198
Obtaining DB2 Drivers . . . 200
Obtaining a MySQL Driver . . . 200
Obtaining an Oracle Driver . . . 201
Obtaining a Generic JDBC Driver . . . 201
Running a SQL Script . . . 202
Enabling End Users to Specify Database Connection Information . . . 202
Preparing Your Installer for Maintenance Mode . . . 203
Enabling and Configuring Maintenance Mode . . . 204
Enabling Maintenance Mode . . . 204
About Maintenance Mode Action Groups . . . 205
Customizing Maintenance Mode Text and Images . . . 207
About Check Running Mode Rules . . . 207
Specifying Instance Management Behavior . . . 208
About the Uninstaller/Maintenance Mode Launcher . . . 210
Run-Time Behavior for Maintenance Mode . . . 211
Configuring Installation Rollback Behavior . . . 214
Building and Testing Installers . . . 216
Working with Build Configurations . . . 216
Creating a New Build Configuration . . . 217
Renaming a Build Configuration . . . 217
Copying a Build Configuration . . . 218
Removing a Build Configuration . . . 218

Adding a Build Configuration to the Project Build . . . 219
Using Tags to Customize Build Configurations . . . 219
Determining Whether a Project Element Is Included in a Build Configuration . . . *220*
Creating New Tags . . . *220*
Assigning Tags to Project Elements . . . *221*
Associating Tags to Build Configurations . . . *222*
Searching for Tags . . . *222*
Importing Tags from Merge Modules . . . *223*
Specifying Locale Settings . . . 223
Defining Build Targets . . . 223
Specifying Build Distribution Options . . . 225
Creating Web Installers . . . 225
Creating CD-ROM/DVD Installers . . . 225
Creating Merge Modules . . . 227
Advertising Merge Module Installer Variables . . . *228*
Specifying How to Show the Progress of a Merge Module Installation . . . *229*
Building Installers Using Build Configurations . . . 229
Building a Specific Build Configuration . . . 229
Building the Build Configurations That Are Enabled for Project Build . . . 230
Building All of the Build Configurations in a Project . . . 230
Using build.exe to Build Installers from the Command Line . . . 230
Using an Ant Task to Build Installers from the Command Line . . . 233
Testing Installers . . . 237
Working with JRE VM Packs . . . 237
Downloading JRE VM Packs for Bundling in Installers . . . 238
Adding JRE VM Packs for Your Installers to Your Development and Build Machines . . . 239
Adding a VM Pack to Your Project . . . 239
Targeting 32-Bit and 64-Bit Windows-Based Systems . . . 239
Enabling or Disabling WOW64 Emulation on 64-Bit Windows-Based Target Systems . . . 241
Creating Pure 64-Bit Installers for 64-Bit Windows-Based Target Systems . . . 242
Creating Launchers for Java Applications . . . 243
Run-Time Behavior for LaunchAnywhere Files on OS and OS X–Based Systems . . . 243
Launching Additional Installers . . . 244
Working with Source Code Control Software . . . 244

6 Adding Advanced Functionality to Installers . . . 245

Working with Variables . . . 247
Setting Variables in the Advanced Designer . . . 247
Preventing the Substitution of Unknown Variables . . . 249
Checking the Value of a Variable . . . 250
Encrypting Variable Values . . . 250
Resolving Variables at Build Time . . . 250
Creating JRE VM Packs . . . 252
Using the Create JRE VM Pack Wizard . . . 252
Directory Name Requirement for Preparing a JRE VM Pack . . . 253

Running the Create JRE VM Pack Wizard as a Standalone Tool . . . 254
Creating a JRE VM Pack Manually . . . 255
JRE VM Pack Structure . . . 255
JRE VM Pack Properties . . . 257
Making a JRE VM Pack FIPS-Compliant . . . 258
Controlling the JVM That Your Installers Use . . . 259
Controlling the Installation of Bundled JVMs . . . 259
Preventing the Installation of the Bundled JVM . . . 260
Installing the Bundled JVM When the Installer Includes a Launcher . . . 260
Installing the Bundled JVM Only When No Valid JVM Exists . . . 260
Preventing the Uninstallation of the Bundled JVM . . . 261
Changing the Bundled JVM Install Folder . . . 261
Changing the Bundled JVM Install Subfolder . . . 262
Controlling the JVM that Your Launchers Use . . . 262
Customizing the VM Search Settings for Your Launchers . . . 262
Customizing the VM Search Paths and Patterns . . . 263
Customizing Windows Search Paths . . . 263
Customizing Java Executable Patterns for Windows-Based Installations . . . 265
Customizing UNIX Search Paths . . . 266
Customizing Java Executable Patterns for UNIX-Based Installers . . . 268
Customizing Individual Launcher Settings . . . 270
Using the Choose Java VM Panel . . . 271
JVM Spec Files . . . 271
Generating Response Files . . . 275
Getting User Input at Run Time . . . 276
Using the Get User Input - Simple Panel . . . 276
Using the Get User Input - Advanced Panel . . . 278
Using Command-Line Arguments with Installers and Uninstallers . . . 279
Setting Project Version at Build Time . . . 283
Checking Disk Space During Installation . . . 284
Preparing Your Installer for Update Notifications . . . 286
Adding an Enable Update Notifications Action . . . 286
Registering Your Product with the FlexNet Connect Publisher Site . . . 287
Removing an Enable Update Notifications Action . . . 287
Localizing Projects and Installers . . . 288
Generating Multilanguage Installers . . . 288
Localizing Resources . . . 288
Localizing Custom Installer Labels . . . 289
Localizing the Splash Screen . . . 289
Localizing the Get User Input Panels . . . 291
Adding an External Resource Bundle . . . 293
Referencing an External Resource Key . . . 294
Internationalizing Custom Code . . . 294
Default to Base Locale for French-Canadian and Portuguese-Brazilian . . . 295

Packaging and Executing Custom Code **295**
Support for Signed JARs as Dependencies 296
Calling InstallShield MultiPlatform APIs in InstallAnywhere **297**
Packaging Custom Code as a Plug-in **298**
Digitally Signing Windows-Based Installers **299**
Support for Digital Signature in Standard Signing 300
Support for Digital Signature in Custom Signing 302
Verifying that Your Output Files Are Digitally Signed on Windows–Based Target Systems 302
About Authentication and Code-Signing Support for OS or OS X–Based Installers **304**
Code-Signing Methods for OS or OS X–Based Installers 305
Requirements for Code-Signing Support for OS or OS X–Based Installers 306
Obtaining a Developer ID Application Certificate for Code Signing OS or OS X–Based Installers 306
Adding the Code-Signing Capability to Your InstallAnywhere Build Machines or Code-Signing Machines 307
Code Signing Your OS or OS X–Based Installers and Including Authentication Support 309
Verifying that Your Code-Signing Output Files Are Working as Expected on OS or OS X–Based Target Systems . . . 311
Troubleshooting Tips for Code-Signing and Authentication Support for OS or OS X–Based Target Systems 314
OS X Notarization **316**
Introduction 316
Notarizing your OS or OS X–Based Installers and Including Authentication Support 317
Determining Whether an Installation Was Successful **319**
Troubleshooting Issues with Installers **319**
Improving Installation Performance 319
Debugging During Installer Development 320
Directing Installer Debug Output 321
Using the Output Debug Information Action 322
Debugging Using Display Message Panel 322
Debugging LaunchAnywhere-Launched Executable Files 323
Debugging During Post-Development 323
Debugging a Windows-Based Installer 323
Debugging a UNIX/Linux or Pure Java Installer 324
Debugging OS or OS X–Based Installers 325
Reviewing Debug Information 325
Troubleshooting Issues with OS or OS X–Based Magic Folders 328
Troubleshooting EBCDIC Encoding Issues 329
Exit Codes 330
Build Exit Codes 330
Installer Exit Codes 335

7 Updating Applications **337**
Requirements for Upgrade Support **337**
Creating an Upgrade **338**
Configuring Upgrade Settings 338
Enabling Upgrade Support 338
Detecting Installed Earlier Versions that Need to Be Updated 339
Adding Upgrade-Specific Actions to Sequences 341

Special Considerations for Upgrade Support **342**

8 Creating Virtual Appliances **345**

About Virtual Appliances **345**
Components of a Virtual Appliance 345
Advantages of Using Virtual Appliances 346
Supported Hypervisors and Platforms for Virtual Appliances 347
Virtual Appliance Creation Workflow 348

Obtaining VM Templates for Virtual Appliances **349**
Downloading VM Templates 349
Creating Custom VM Templates 350
About Creating VM Templates 350
Creating VM Templates Using the Create InstallAnywhere VM Template Wizard 351
Creating VM Templates Using the Command Line 352

Creating a VMware vSphere 5 Virtual Appliance **355**

Creating an Amazon EC2 Virtual Appliance **359**

Working with Virtual Appliance Configurations **363**
Creating a New Appliance Configuration 363
Renaming an Appliance Configuration 364
Copying an Appliance Configuration 364
Removing an Appliance Configuration 365
Adding an Appliance Configuration to the Project Build 365
Specifying Connection Settings for Building Virtual Appliances 366
Configuring the Appropriate Appliance Output and Type for VMware-Based Virtual Appliances 367
Specifying the Appliance URL 368
Support for Elastic Block Store (EBS) for Amazon 368
About EC2 and S3 Regions of Deployment for Amazon 369

Working with VM Configurations for a Virtual Appliance **370**
Selecting a VM for a VMware Virtual Appliance 370
Specifying Hardware Requirements for a Virtual Appliance 371
Installing Operating System Packages for a Virtual Appliance 372
Specifying Which OS Package Repository to Use for Virtual Appliances 373
Adding Installers to Your Virtual Appliance 374
Using RPM/Debian Packages for Proprietary Operating Systems on Virtual Appliances 375
Using Custom Boot and Login Scripts for a Virtual Appliance 377
Customizing Product Properties 379

Identifying Different Virtual Appliances and VMs **380**

Managing Multiple Tiers for a Virtual Appliance **380**

Building a Virtual Appliance **381**
Building a Specific Appliance Configuration 381
Building the Appliance Configurations That Are Enabled for Project Build 381
Building All of the Appliance Configurations in a Project 382
Using build.exe to Build Virtual Appliances from the Command Line 382
Building Virtual Appliances Through Ant Task-Based Build 382
Build Output for VMware vSphere 5 Virtual Appliances 383

Build Output for Amazon EC2 Virtual Appliances . . . 383
Deploying a Virtual Appliance . . . 384
Deploying a VMware vSphere 5 Virtual Appliance on Demand . . . 385
Deploying an Amazon EC2 Virtual Appliance on Demand . . . 386
Deploying an Amazon Virtual Appliance Through the Amazon Web Services (AWS) Management Console . . . 386
Deploying an Amazon EC2 Virtual Appliance Through Command-Line Tools . . . 390
Using the IABookstore Sample for Creating Virtual Appliances . . . 392
Using the IABankingApplication Sample for Creating Multi-Tier Virtual Appliances . . . 392

9 Reference . . . 395
Advanced Designer Reference . . . 396
Project Page . . . 397
General Settings View . . . 398
Platforms View . . . 412
Locales View . . . 427
Manage Expressions View . . . 428
Installer Rules View . . . 432
JVM Settings View . . . 434
General Settings Tab in the JVM Settings View . . . 435
Installer Settings Tab in the JVM Settings View . . . 435
Search Panel Settings Tab in the JVM Settings View . . . 439
Variables View . . . 443
Advanced View . . . 445
Maintenance Mode Settings . . . 446
Windows WOW64 Emulator Settings . . . 448
Instance Management Settings . . . 448
Manage Tags Settings . . . 451
Rollback Settings . . . 451
Upgrades View . . . 452
Installer UI Page . . . 460
Look & Feel Settings View . . . 460
UI Panel Settings Area . . . 460
General UI Settings Area . . . 470
Maintenance Mode Runtime Panel Label Settings Area . . . 473
Installer Icon Area . . . 473
Billboards View . . . 474
Help View . . . 475
Organization Page . . . 476
Install Sets View . . . 476
Features View . . . 477
Components View . . . 477
Modules View . . . 480
Hosts View . . . 484
Application Server Host Settings . . . 485
Database Server Host Settings . . . 489
Sequence Page . . . 490

Pre-Install View . . . 491
Install View . . . 491
Post-Install View . . . 494
Pre-Uninstall View . . . 495
Uninstall View . . . 495
Post-Uninstall View . . . 497
Build Page . . . 497
Build Installers View . . . 497
Build Configurations Tab . . . *497*
Build Targets Subtab . . . 500
Distribution Subtab . . . 504
Tags Subtab . . . 506
Locales Subtab . . . 506
Build Log Tab . . . *507*
Build Appliances View . . . 508
Appliance Configuration Tab . . . *509*
VM Configuration Tab . . . *516*
Product Properties Subtab . . . 517
Hardware Subtab . . . 518
OS Packages Subtab . . . 519
Installers Subtab . . . 520
Script Info Subtab . . . 521
Repository Settings Subtab . . . 522
Manage VM Tiers Tab . . . *523*
Build Log Tab . . . *523*
Troubleshooting “Out of Memory” Error Upon InstallAnywhere Launch or Command Line Build . . . 524
Wizard Reference . . . 525
Choose a VM Wizard . . . 526
Select a Virtual Machine/Template Panel . . . 526
Summary of Selected VM Panel . . . 527
Create InstallAnywhere VM Template Wizard . . . 528
InstallAnywhere VM Template Information Panel . . . 529
VMware OS Information Panel . . . 529
Amazon OS Information Panel . . . 531
Review EULA for Creating the VM Template Panel . . . 531
VMware vCenter/vSphere 5 Information Panel . . . 532
Review Summary Information Panel . . . 532
Creating the Baseline VM Panel . . . 532
Finish Panel . . . 533
Menu Reference . . . 533
File . . . 533
Edit . . . 534
Tools . . . 536
Help . . . 536
Dialog Box Reference . . . 536
About InstallAnywhere Dialog Box . . . 538
Add Files to Project Dialog Box . . . 538
Add Installer/Edit Installer Dialog Box . . . 539

Add New/Manage Connection Dialog Box . . . 542
Advertise Variables Dialog Box . . . 544
Build Time Variables Dialog Box . . . 546
Configure Input Items Dialog Box . . . 546
Change Disk Space and Name Dialog Box . . . 546
Choose an Action Dialog Box . . . 547
Choose Icon Dialog Box . . . 549
Choose Label Settings Dialog Box . . . 549
Choose Variable Dialog Box . . . 550
Configure Search and Replace Strings Dialog Box . . . 551
Create New Project Dialog Box . . . 551
Create JRE VM Pack Dialog Box . . . 552
Custom Rules Dependencies Dialog Box . . . 552
Edit Advertised Variables Dialog Box . . . 553
Edit Build Time Variables Dialog Box . . . 554
Edit Installer Properties Table Dialog Box . . . 554
Filter File Dialog Box . . . 555
Install Step Label Settings Dialog Box . . . 556
InstallAnywhere Merge Module Import Assistant Dialog Box . . . 557
InstallAnywhere Preferences Dialog Box . . . 557
General Settings Tab . . . 558
Resources Tab . . . 559
Source Paths Tab . . . 560
Updates Tab . . . 561
Installer Steps Dialog Box . . . 562
LaunchAnywhere Properties Dialog Box . . . 564
Manage Instances Dialog Box . . . 564
Manage Upgrade Configurations Dialog Box . . . 564
Create/Open Project Dialog Box . . . 566
Create New Project Dialog Box . . . 566
Open Project File Dialog Box . . . 567
Prepare the Helper Tool Dialog Box . . . 568
Read/Modify XML File Dialog Box . . . 569
RPM Specification Settings Dialog Box . . . 569
Save New Project As Dialog Box . . . 571
Search Results Dialog Box . . . 572
SWVPD Registry Settings Dialog Box . . . 574
Uninstaller Properties Dialog Box . . . 574
Uninstaller Steps Dialog Box . . . 574
Uninstall Step Label Settings Dialog Box . . . 576
Actions . . . 579
Install Actions . . . 580
Create Alias, Link, Shortcut Action . . . 582
Create Folder Action . . . 583
Create LaunchAnywhere for Java Application Action . . . 584
Create Uninstaller Action . . . 587

Deploy WAR/EAR Archive Action . . . 589
Download File Action . . . 594
Enable Update Notifications Action . . . 595
Expand Archive Action . . . 599
Expand Archive (7-zip) Action . . . 600
Expand Archive (TAR) Action . . . 601
Install Archive Action . . . 602
Install File Action . . . 602
Install HP-UX Depot Action . . . 603
Install Merge Module Action . . . 604
Install Solaris Package Action . . . 606
Install SpeedFolder Action . . . 607
Set System Environment Variable Action . . . 609
Run SQL Script Action . . . 610
Trigger Rollback Action . . . 613
Uninstall Actions . . . 614
General Actions . . . 615
Delete Folder Action . . . 620
Execute Ant Script Action . . . 621
Execute Command Action . . . 623
Execute Custom Code Action . . . 624
Execute Script/Batch File Action . . . 625
Find Component in InstallAnywhere Registry Action . . . 626
Modify Text File - In Archive Action . . . 628
Modify Text File - Multiple Files Action . . . 631
Modify Text File - Single File Action . . . 634
Read/Modify XML File Action . . . 636
Show Message Dialog Action . . . 640
Associate File Extension/URL Prefix - Windows Action . . . 641
Panel Actions . . . 644
Choose Database Connection Panel Action . . . 648
Choose Install Sets Panel Action . . . 650
Choose Java VM Panel Action . . . 651
Custom Code Panel Action . . . 653
Display HTML Panel Action . . . 654
License Agreement Panel Action . . . 655
Minimal UI Panel Action . . . 657
Pre-Install Summary Panel Action . . . 658
Tomcat Runtime Deployment Panel Action . . . 659
WebSphere Runtime Deployment Panel Action . . . 661
Console Actions . . . 664
Choose Database Connection Console Action . . . 666
Get Password Console Action . . . 668
Tomcat Runtime Deployment Console Action . . . 669
WebSphere Runtime Deployment Console Action . . . 671
System i (i5/OS) Actions . . . 673

Choose Remote System i (i5/OS) Install Folder Action . . . 674
Get System i (i5/OS) Login Credentials Action . . . 675
System i (i5/OS) Command Action . . . 677
System i (i5/OS) Find Component in RAIR Action . . . 677
System i (i5/OS) Install File Action . . . 678
System i (i5/OS) Integrated File System (IFS) Action . . . 679
System i (i5/OS) Library Action . . . 680
System i (i5/OS) Licensed Program Action . . . 681
System i (i5/OS) Object Action . . . 683
System i (i5/OS) Process Remote Install (Merge Modules) Action . . . 684
System i (i5/OS) Program Action . . . 685
System i (i5/OS) Program Temporary Fix (PTF) Action . . . 686
Plug-In Actions . . . 687
ExecuteAsRoot Action . . . 688
ExtractToFile Action . . . 689
Properties File Reader Action . . . 690
Common Action Settings . . . 690
Common Customizer Settings . . . 691
Panel Action Settings . . . 691
Get User Input Panels . . . 694
Get User Input - Simple . . . *695*
Get User Input - Advanced . . . *697*
Get User Input - Console . . . *698*
Rules Reference . . . 699
Check File/Folder Attributes Rule . . . 701
Check If File/Folder Exists Rule . . . 702
Check Platform Rule . . . 703
Check Running Mode Rule . . . 704
Compare InstallAnywhere Variables Numerically Rule . . . 706
Compare Versions . . . 707
Evaluate Custom Rule Rule . . . 707
Variables . . . 709
Standard InstallAnywhere Variables . . . 710
$IA_BROWSE_FOLDERS$ Variable . . . 724
LAX Properties . . . 726
Magic Folders . . . 730
Localization Reference . . . 734
Language Codes . . . 734
Common Localizable Elements . . . 736
Files and File Formats . . . 740
Product Registry . . . 740
Install Log File Format . . . 741
Manifest Files . . . 742
Response Files . . . 743
BuildProperties.xml File . . . 746
Supported Properties in the BuildProperties.xml File . . . 747

buildproperties.properties File 751
Supported Properties in the buildproperties.properties File 752
Command-Line Reference 756
Build Command-Line Arguments 756
Installer and Uninstaller Command-Line Arguments 762
Maintenance Mode Command-Line Arguments. 765
Launcher Command-Line Arguments 766
InstallAnywhere Ant Task Reference 766
Custom Code APIs. 773
Project Automation API Code Samples 775
Project Automation APIs for Maintenance Mode and Instance Management 777

Index 781

1

InstallAnywhere 2026 R1 Help Library

InstallAnywhere is a multiplatform installation development solution for application producers who need to deliver a professional and consistent installation experience for physical, virtual, and cloud environments. From a single project file and build environment, InstallAnywhere creates reliable installations for on-premises platforms—Windows, Linux, macOS and OS X, Solaris, AIX, HP-UX, IBM iSeries—and enables you to deploy your products into cloud and virtual environments.

Information about using InstallAnywhere is presented in the following sections:

Table 1-1 ▪ InstallAnywhere Help Library

Section	Description
What Was New in Earlier Versions of InstallAnywhere	Informs you about changes that were made in earlier versions of in InstallAnywhere.
Using Help	Provides information about the InstallAnywhere documentation.
Getting Started	Describes the InstallAnywhere interface authoring environments and explains how to begin creating an installation project.
Concepts of InstallAnywhere Installer Development	Introduces you to the basic and advanced concepts of InstallAnywhere installation development.
InstallAnywhere Tutorial	Provides guided exercises to explore using InstallAnywhere.
Creating Basic Installers	Provides procedures and suggestions about common situations when using InstallAnywhere to build a basic installer.
Adding Advanced Functionality to Installers	Explains how to add more advanced procedures to an InstallAnywhere installer.
Updating Applications	Describes how to create update installers for applications.

Table 1-1 ▪ InstallAnywhere Help Library (cont.)

Section	Description
Creating Virtual Appliances	Explains how to use an existing InstallAnywhere project to build virtual appliances for cloud platforms such as Amazon EC2 and VMware vSphere.
Reference	Contains comprehensive reference information for the InstallAnywhere user interface; available actions that define operations that an installation performs; parameters for the command-line tools that enable you to perform tasks such as building or running an installation; the Ant task for building installations; custom code APIs for automating development and testing of installations; and more.

What Was New in Earlier Versions of InstallAnywhere

This section describes features and enhancements that were released in earlier versions of InstallAnywhere:

- What's New in InstallAnywhere 2025 R1
- New Features in InstallAnywhere 2024 R2
- New Features in InstallAnywhere 2024 R1
- What's New in InstallAnywhere 2023 R1
- What's New in InstallAnywhere 2022 R2
- What's New in InstallAnywhere 2022
- What's New in InstallAnywhere 2021 SP2
- What's New in InstallAnywhere 2021 SP1
- What's New in InstallAnywhere 2021
- What's New in InstallAnywhere 2020 SP2
- What's New in InstallAnywhere 2020 SP1
- What's New in InstallAnywhere 2020
- What's New in InstallAnywhere 2018 SP1
- What's New in InstallAnywhere 2018
- What's New in InstallAnywhere 2017 SP1
- What's New in InstallAnywhere 2017
- What's New in InstallAnywhere 2015 SP1

- What's New in InstallAnywhere 2015
- What's New in InstallAnywhere 2014 SP1
- What's New in InstallAnywhere 2014
- What's New in InstallAnywhere 2013

What's New in InstallAnywhere 2025 R1

InstallAnywhere 2025 R1 includes the following new feature:

- Support for macOS Universal Binaries

Support for macOS Universal Binaries

Previously, InstallAnywhere was compiled only for x64 Intel architectures. On macOS systems, powered by the Apple Silicon (Arm64/AArch64) architectures, InstallAnywhere relied on Rosetta—Apple's translation layer—to emulate x64 Intel environments. This emulation sometimes results in reduced performance, application hangs, and compatibility limitations on macOS platforms.

Starting with this release, InstallAnywhere is compiled as a Universal Binary. A Universal Binary includes executable code for both Intel x64 and Apple Silicon (Arm64/AArch64) architectures in a single package. This enhancement enables InstallAnywhere to operate natively on both Intel-based and Apple Silicon-based systems, ensuring optimal performance and full compatibility across all supported macOS platforms.

New Features in InstallAnywhere 2024 R2

InstallAnywhere 2024 R2 includes the following new feature:

- Introduced Support for ARM64

Introduced Support for ARM64

InstallAnywhere 2024 R2 introduces support for ARM64 (AARCH64) Linux distribution, which enables you to build projects on both Windows and Linux operating systems running the ARM64 (AARCH64) architecture.

New Features in InstallAnywhere 2024 R1

InstallAnywhere 2024 R1 includes the following new features:

- Ability to Configure Custom Signing Solution for Digital Signing
- Introduced New Component for Custom Code Panels
- Introduced Exit Code for Invalid Java Virtual Machine

Ability to Configure Custom Signing Solution for Digital Signing

InstallAnywhere 2024 R1 introduces new signing type settings that enable you to select and configure a custom signing solution to digitally sign your Windows-based installers (installer .exe file, installer launcher, and uninstaller launcher) during build time. If the user prefers to use the custom signing solution instead of the InstallAnywhere standard solution, these settings are essential.

Selecting the **Custom** signing type enables additional fields where a custom signing tool path and arguments can be configured. On the **Project > Platforms > Windows** area, under the **Digital Signing** setting, you can use following new settings to configure the custom signing solution:

- **Signing Method**—This setting enables you to specify the method to digitally sign your Windows-based installers during build time. The available options for this setting are:
  - **Standard**—Select this option to utilize the default InstallAnywhere sign tool to digitally sign your Windows-based installers during build time.
  - **Custom**—Select this option to use an external sign tool to digitally sign your Windows-based installers during build time. Selecting this option enables the **Custom Sign Tool Location** and **Sign Tool Arguments** settings.

By default, the **Signing Method** setting is set to the **Standard** option.

- **Custom Sign Tool Location**—Specify the location of the external sign tool that you want to use to digitally sign your Windows-based installers during build time. To specify sign tool's location, click the ellipsis button (...).
- **Sign Tool Arguments**—Specify the valid argument for an external sign tool's configuration. For instance, the following argument can be used for an external sign tool's configuration:

  ```
  sign /a /fd SHA256 /t http://timestamp.digicert.com
  ```

The following displays the enabled **Custom Sign Tool Location** and **Sign Tool Arguments** settings when the **Signing Method** setting is set to the **Custom** option:

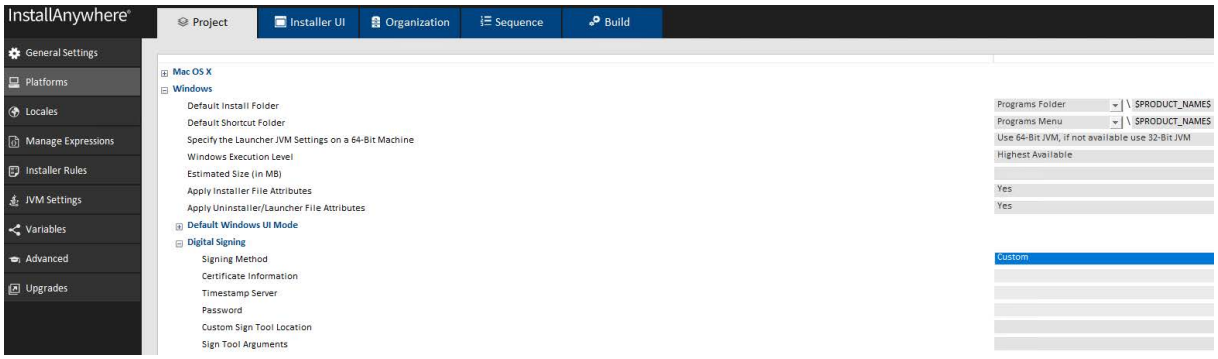


Figure 1-1: Enabled Settings for Custom Signing

For more information, see the following:

- Platforms View
- Digitally Signing Windows-Based Installers

Introduced New Component for Custom Code Panels

Previously, InstallAnywhere Custom Code panels were unable to support multiple-line contents and URLs.

InstallAnywhere 2024 R1 introduces a new component named, `IATextPane`, which enables the custom code panels to support multiple-line contents, URLs, and DPI scaling. This component is available in the `com.zerog.ia.api.pub.controls` package and can be used in Custom Code panels. The Javadocs for this component are available in the `\javadoc\com\zerog\ia\api\pub\controls` directory.

The following displays a Custom Code panel using the `IATextPane` component during installation:

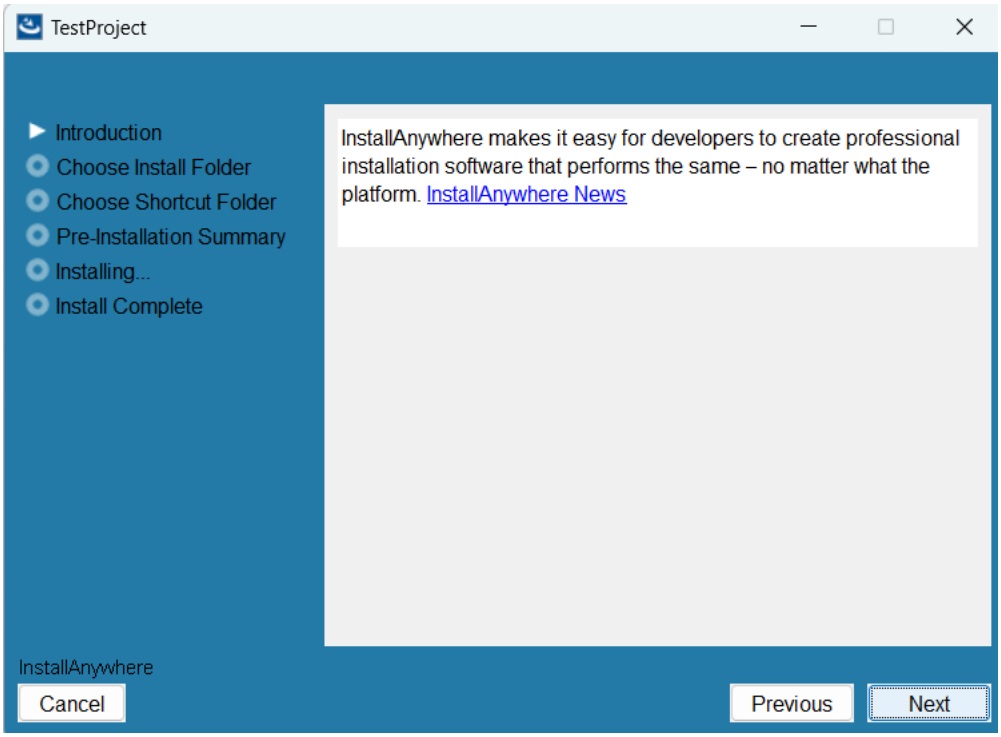


Figure 1-2: Custom Code Panel using `IATextPane`

Introduced Exit Code for Invalid Java Virtual Machine

Previously, when an InstallAnywhere installer failed to detect a valid Java Virtual Machine (JVM) during installation, the following error message appeared without an exit code:

```
Could not find a valid Java Virtual machine to load. You may need to reinstall a supported java virtual machine.
```

In InstallAnywhere 2024 R1, if an InstallAnywhere installer fails to detect a valid Java Virtual Machine (JVM) during installation, an exit code 106 will be returned and will be displayed on the console along with the following error message as:

```
Could not find a valid Java Virtual machine to load. You may need to reinstall a supported java virtual machine. exit code : 106.
```

What's New in InstallAnywhere 2023 R1

InstallAnywhere 2023 R1 includes the following new features:

- Support for LZMA2 Compression Using Expand 7-Zip Action
- Ability to Customize Image, Label, and Help Settings for Pre-Uninstall and Post-Uninstall Sequences

- Support for the Windows 11 Smart App Control
- Support for the macOS Sonoma

Support for LZMA2 Compression Using Expand 7-Zip Action

In previous releases, InstallAnywhere let you to use the **Expand Archive (7-zip)** action in the **Install** view on the **Sequence** page to expand a 7-Zip archive (.7z or .xz) with the LZMA compression method only.

This release includes the LZMA2 compression method to expand a 7-Zip archive (.7z or .xz). As a consequence, the **Expand Archive (7-Zip)** action supports both the LZMA and LZMA2 compression methods to expand a 7-Zip archive (.7z or .xz) on a target system.

Ability to Customize Image, Label, and Help Settings for Pre-Uninstall and Post-Uninstall Sequences

In this release, you can enable and customize the following tabs on the **Properties Customizer** of panels in the **Pre-Uninstall** and **Post-Uninstall** views on the **Sequence** page to configure the uninstaller steps that are displayed on the navigation panel for the uninstaller:

- **Image settings**—On the **Installer UI > Look & Feel Settings > Customer UI Designer** dialog box, under the **Installer Steps** area, selecting **Images** from the **Installer Steps Type** drop-down menu enables you to customize the **Image Settings** tab on the **Properties Customizer**. By default, the **Image Settings** tab is disabled.
- **Label Settings**—On the **Installer UI > Look & Feel Settings > Customer UI Designer** dialog box, under the **Installer Steps** area, setting **Enable Uninstall Label Settings** to **Yes** enables you to customize the **Label Settings** tab on the **Properties Customizer**. By default, the **Label Settings** tab is disabled.
- **Help Settings**— On the **Installer UI > Help** view, selecting the **Enable installer help** and **Use different help text for each panel** options enables you to to customize the **Help Settings** tab on the **Properties Customizer.** By default, the **Help Settings** tab is disabled.



Note ▪ *To configure the uninstaller step labels, the* ***Enable Uninstall Label Settings****,* ***Configure Uninstall Labels*** *and* ***Uninstall Progress Label*** *settings have been introduced in the* ***Installer Steps*** *section on the* ***Customer UI Designer*** *dialog box in the* ***Look & Feel Settings*** *view on the* ***Installer UI*** *page.*

Support for the Windows 11 Smart App Control

InstallAnywhere now supports the Smart App Control feature of the Windows 11 operating system.

Support for the macOS Sonoma

The Setups created with InstallAnywhere 2023 R1 can now run on the macOS Sonoma version.

What's New in InstallAnywhere 2022 R2

InstallAnywhere 2022 R2 includes the following new features:

- New Project Properties to Manage the Progress Bar
- New Build Property to Manage the JRE

- New Setting to Upgrade the App Notarization
- Support for macOS Ventura

New Project Properties to Manage the Progress Bar

In previous releases, InstallAnywhere installers displayed a single progress bar that showed the progress of the entire installer using a percentage from 0% to 100%. The following is an example of a standard progress bar:

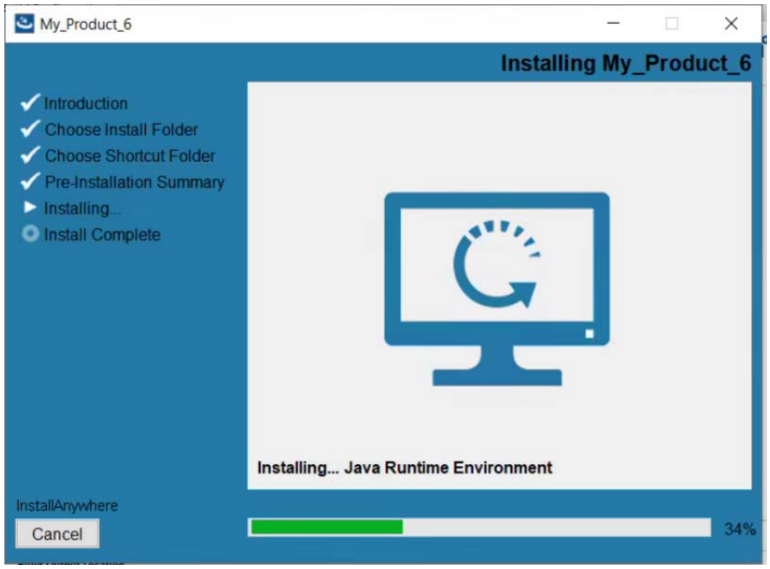


Figure 1-3: Standard Progress Bar

In InstallAnywhere 2022 R2, you have the option of displaying an “indeterminate progress bar” that indicates that installer is running but does not show the percentage of progress. The following is an example of an indeterminate progress bar:

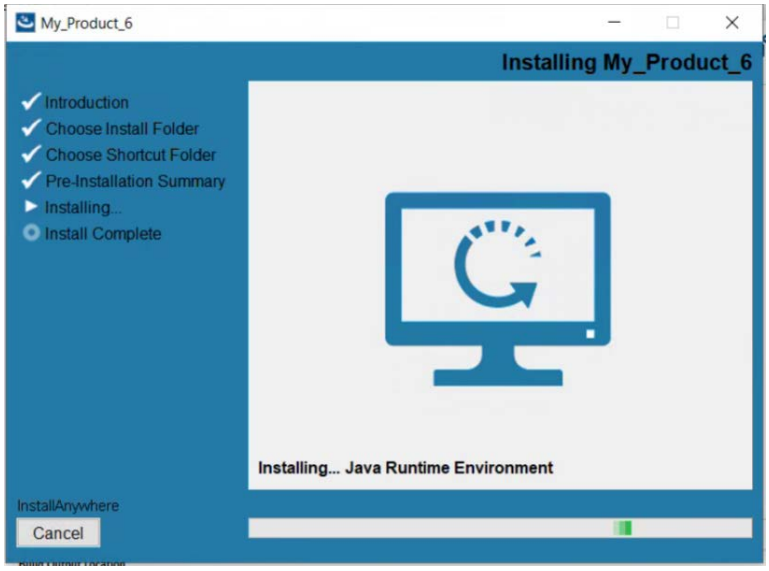


Figure 1-4: Indeterminate Progress Bar

On the **Look & Feel Settings > Installer UI > Customer UI Designer** dialog box, under **Inner Install Frame**, there are two new options to enable you to customize the progress bar displayed.

The new progress bar options are:

- **Show Progress Bar**—Set this option to **Yes** to display the standard progress bar that shows the progress of the entire installer by percentage complete, from 0% to 100%. If this option is set to **No**, the **Show Indeterminate Progress Bar** will be enabled. By default this option is set to **Yes**.
- **Show Indeterminate Progress Bar**—Set this option to **Yes** to display a bar to indicates that the installer is running but does not show the percentage of progress. If **Show Progress Bar** is set to **Yes**, this option is disabled.

If both progress bar options are set to **No**, then progress of installer will not be displayed at all.The following is an example when both progress bar options are disabled:

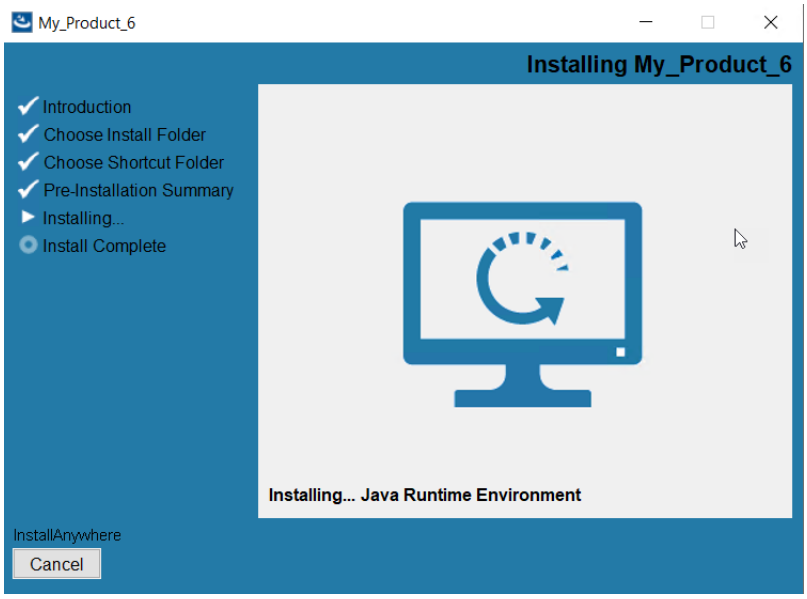


Figure 1-5: Both Progress Bars Disabled

Note ▪ *This change was tracked in issue IOJ-2254111.*

New Build Property to Manage the JRE

On the **Build> Build Installers > Build Configuration > Distribution** subtab, under **Web Installer Options,** there is new option **Bundle JRE as Directory** to enable you to manage the JRE file.

In previous releases, by default, InstallAnywhere Installers for the macOS operating system contained the JRE bundled as `jre.zip`.

In InstallAnywhere 2022 R2, you can select the new **Bundle JRE as Directory** option to make InstallAnywhere Installers bundle the JRE as a directory inside the macOS installer. The following shows the **Bundle JRE as Directory** option selected:

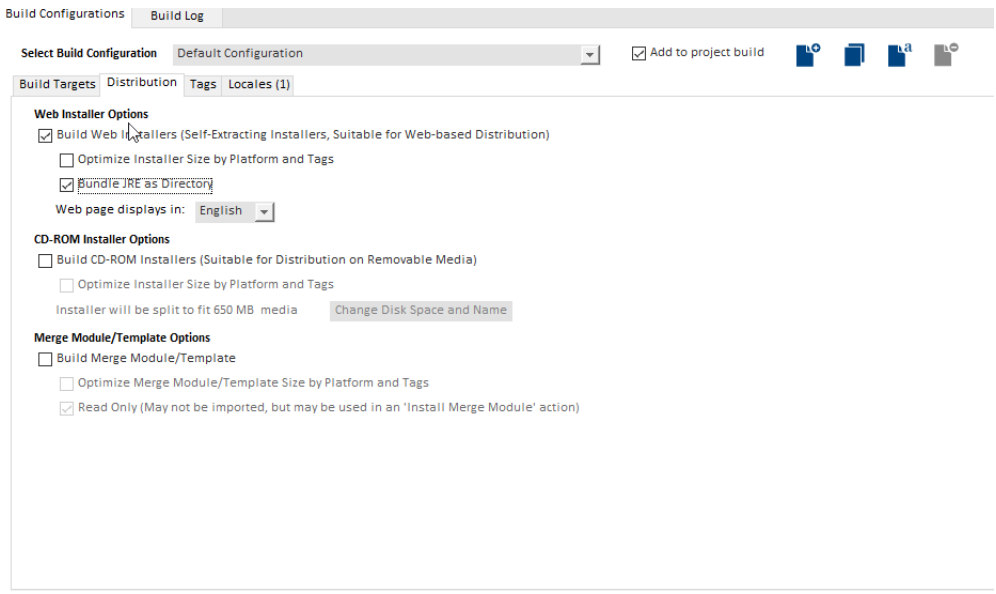


Figure 1-6: Bundle JRE as Directory Option

If the **Bundle JRE as Directory** option is unchecked, the JRE in the macOS installer is bundled as `jre.zip`. By default, this option is unchecked.

Note ▪ *This new feature is applicable to DMG distribution with VM installers to macOS operating system.*

Note ▪ *This change was tracked in issue IOJ-2227434.*

New Setting to Upgrade the App Notarization

On the **Platform > Project** page, under **Mac OS X > App Notarization**, InstallAnywhere 2022 R2 now includes a new field named **Team Identifier** to support the notarization of OS X-based installers.

In the **Team Identifier** field, you can define the team identifier information which is accessible on the Apple Developer certificate. If the **Notarize the Generated Installer** check box under **App Notarization** is selected, the **Team Identifier** field is enabled. The following shows the **Team Identifier** field enabled:

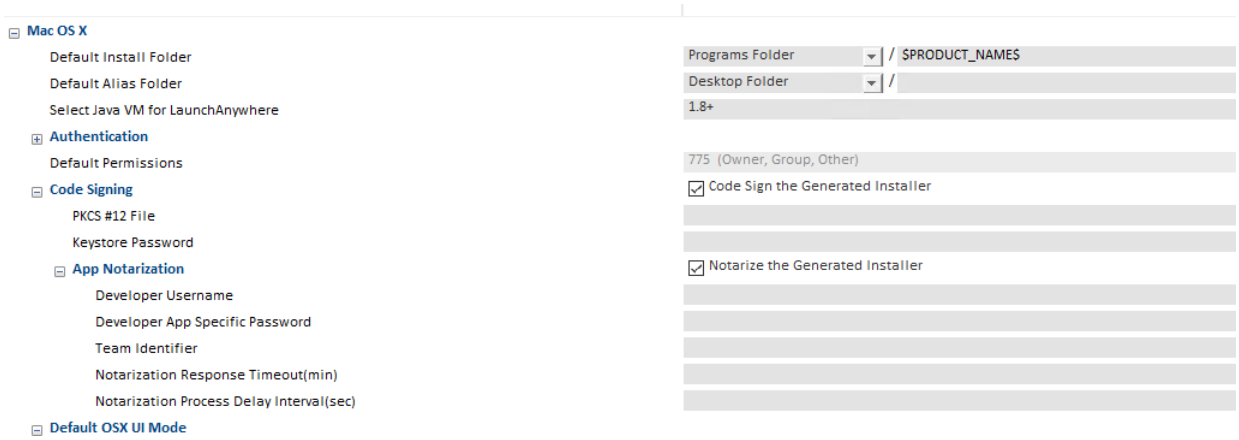


Figure 1-7: Team Identifier Enabled

In the **Team Identifier** field, enter the team identifier information which is accessible on the Apple Developer certificate.

Note ▪ *This change was tracked in issue IOK-930701.*

Support for macOS Ventura

The Setups created with InstallAnywhere 2022 R2 can now run on macOS Ventura version.

Note ▪ *This change was tracked in issue IOK-883482.*

What's New in InstallAnywhere 2022

InstallAnywhere 2022 includes the following new features:

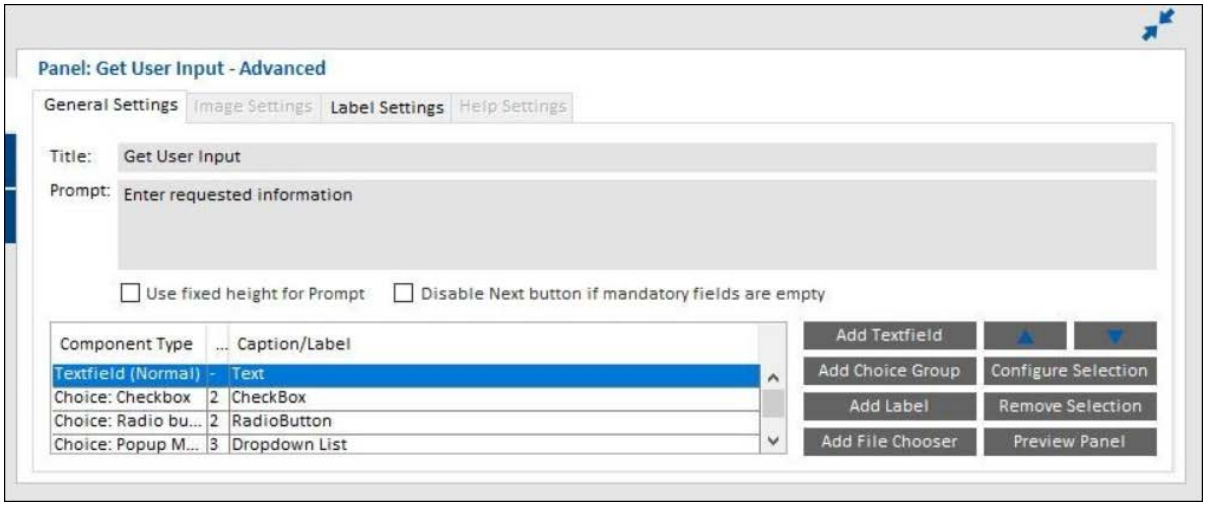
- Enhanced Get User Input Panel - Advanced
- New Project Property for Status Message Control
- Support to Register URL Prefix with Application
- New Option for Log Format
- New Property for Defining the Permission for Log file
- Upgrade of InstallAnywhere 2022 with 64-bit JDK

Enhanced Get User Input Panel - Advanced

Added a checkbox "Disable Next button if mandatory fields are empty" option in the Get User Input panel to enable/disable the Next button for notifying unfilled mandatory fields. Either you can select or not select this checkbox based on the requirement.

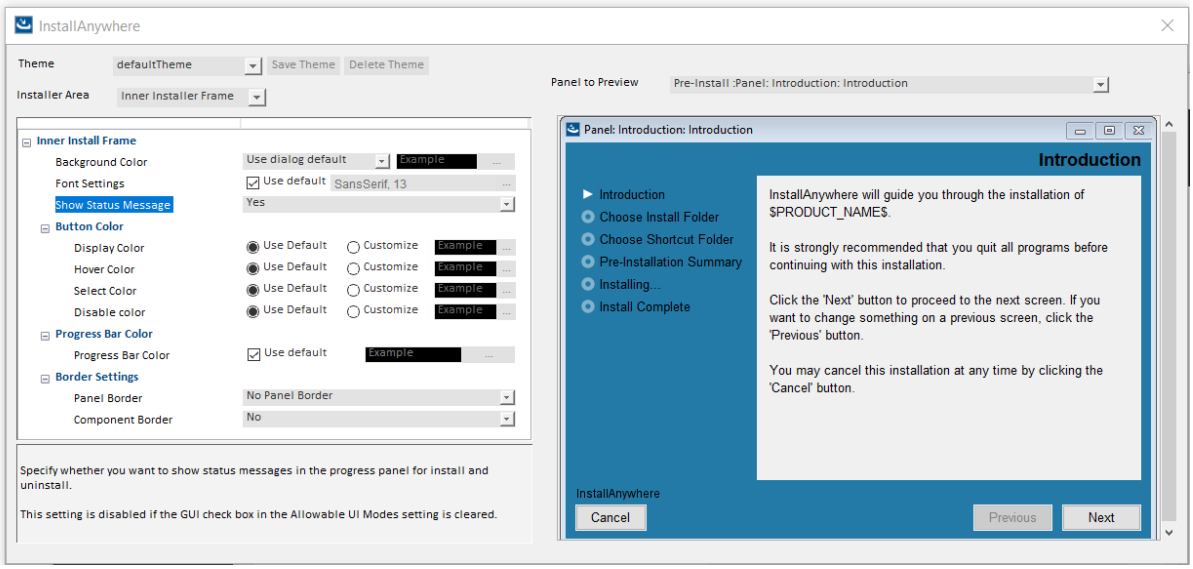
- When checked, the Next button in the Get User Input panel would be disabled until user enters a value in the mandatory text field.
- When unchecked, then a pop-up dialog appears prompting the user to provide values in the mandatory fields.

By default, this checkbox is unchecked.



New Project Property for Status Message Control

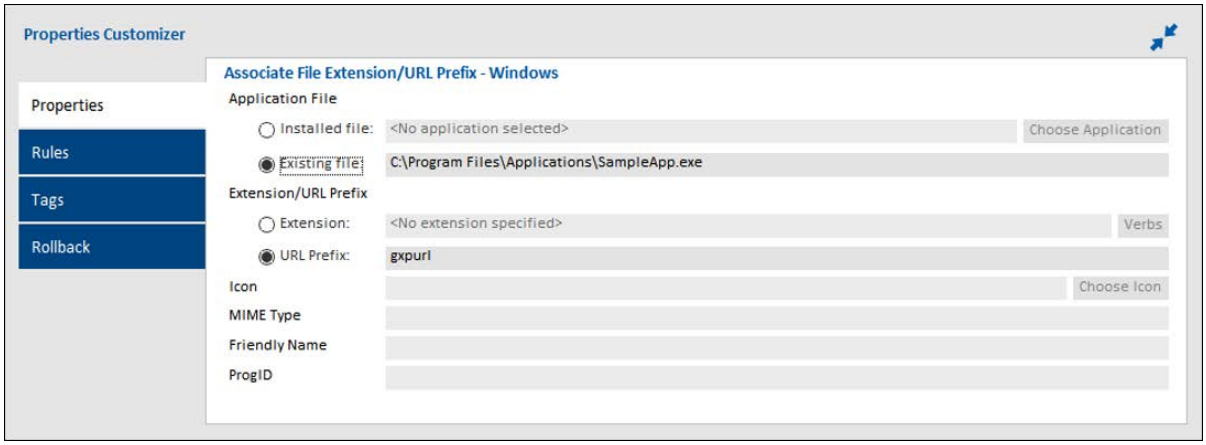
Enhanced InstallAnywhere to configure whether actions should display status messages in Install & Uninstall progress panels or not. A new project property "Show Status Message" has been added in Installer UI > Custom UI Designer > Inner Install Frame for controlling status message. By default, this option is set to "Yes". If this option is set to "No", status messages will not be displayed above the progress bar on the Progress Panel during Install & Uninstall.



Support to Register URL Prefix with Application

Enhanced "Associate File Extension - Windows" action to add support for registering URL Prefix with application. The action has been renamed to "Associate File Extension/URL Prefix - Windows". New property "URL Prefix' has been added in this action available in Install and Post-Install phases.

The URL Prefix can now be associated to an Existing or Installed application.



New Option for Log Format

Added a new option “Show 24 hour date format in logs” in the Log Settings to display the log in 12-hour or 24-hour format. When “Show 24 hour date format in logs” is set, the logs are displayed in 24-hour log format. Else, the logs are displayed in 12-hour log format.

New Property for Defining the Permission for Log file

Added a new property “Unix Permission for Logs” under Project > General Settings > Log Settings to define the permissions of the log file when installed in Linux operating system.

Upgrade of InstallAnywhere 2022 with 64-bit JDK

Upgraded InstallAnywhere2022 with 64-bit JDK 1.8 for the automation jar to work with any Node-lock license checkout.

What’s New in InstallAnywhere 2021 SP2

InstallAnywhere 2021 SP2 includes the following new features:

- Subscription Based Licensing

Subscription Based Licensing

InstallAnywhere 2021 SP2 and later offers subscription licensing with two different licensing models: a node-locked subscription license and a concurrent subscription license. A concurrent subscription license can be configured with FlexNet Licensing Server.

- **Node-Locked subscription License:** Node-locked subscription license is to be used by one user on one machine or virtual image, and may not be installed on shared computers. InstallAnywhere will be licensed for the subscription duration to which the node-locked license is entitled.
- **Concurrent subscription License:** If your organization has purchased concurrent licenses of InstallAnywhere or the Standalone Build, a FlexNet Licensing Server that you set up in your environment manages how many instances of the product can be run simultaneously. Each user who wants to launch one of these products needs to establish a connection with the licensing server to check out and check in licenses when needed.InstallAnywhere will be licensed for the subscription duration to which the concurrent license is entitled.

InstallAnywhere updates the subscription details in the About dialog box for all kind of subscription licensing. On the Help menu, click About InstallAnywhere. The About InstallAnywhere dialog box opens and shows the Subscription details after the activation. The About InstallAnywhere dialog box shows the subscription end date and the remaining days left for the current subscription.

The About InstallAnywhere dialog box for Node-Locked subscription license:



The About InstallAnywhere dialog box for Concurrent subscription license:



What's New in InstallAnywhere 2021 SP1

InstallAnywhere 2021 SP1 includes the following new features and enhancements:

- Support for Windows Server 2022
- Codesign with the "Developer ID Application:" String

Support for Windows Server 2022

Now, the Installer runs on Windows Server 2022 machine as well with all Java version.

Codesign with the "Developer ID Application:" String

Now, the macOS installer supports codesigning with "Developer ID Application:" string for the certificate name.

What's New in InstallAnywhere 2021

InstallAnywhere 2021 includes the following new features and enhancements:

- Support of File Associations in Windows
- Support for 8.3 File Naming

Support of File Associations in Windows

In InstallAnywhere 2021, new Action has been added to handle File Associations on Windows platforms. This Action is available for all project types. New Action “Associate File Extension - Windows” is added to Install Phase & Post- Install Phase. It is available at:

- **Sequence** > **Install** > **Add Action** > **General**
- **Sequence** > **Post-Install** > **Add Action** > **General**

Support for 8.3 File Naming

You can now enable 8.3 file naming for all volumes with elevated privileges for Windows target machines. Added new action “Enable 8.3 File Naming Action” to enable 8.3 naming for all volumes and added a new variable “$IA_8DOT3_FILENAMECREATION_STATE$” to identify whether the 8.3 File creation is enabled or not for all volumes.

What’s New in InstallAnywhere 2020 SP2

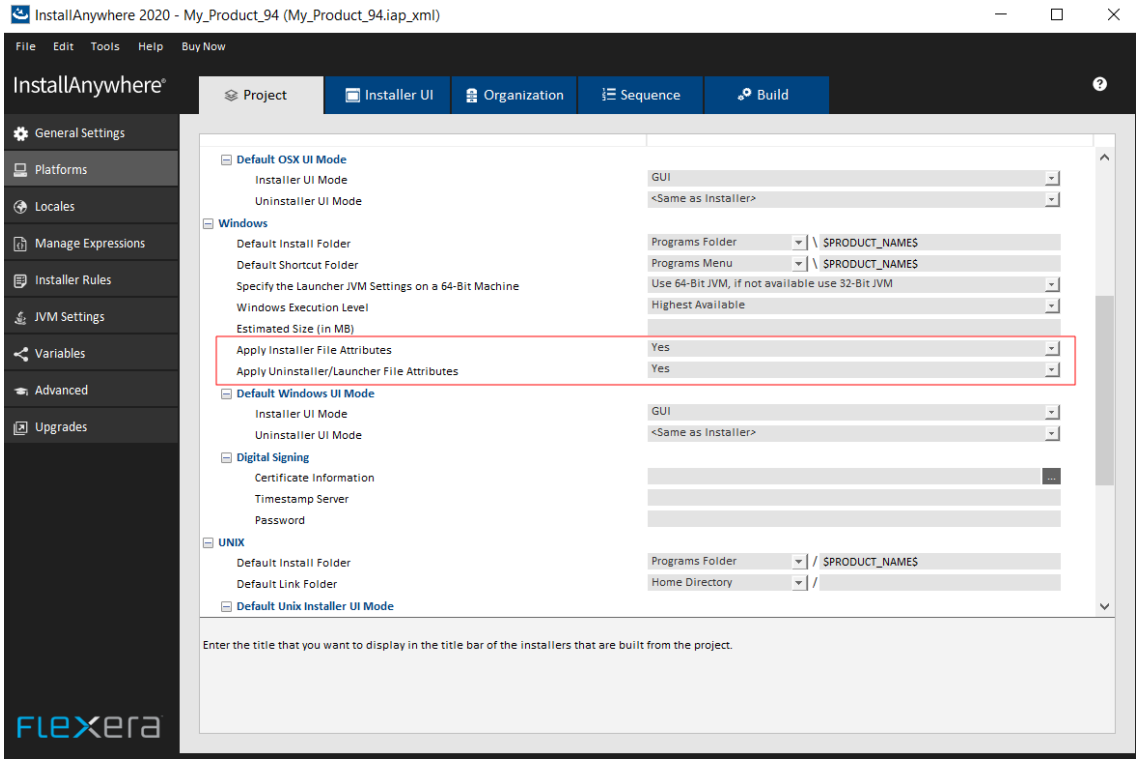
InstallAnywhere 2020 SP2 includes the following change:

- Option to Turn-Off the Application of File Attributes
- Enable Silent and Console Mode by Default for Basic Template
- Running InstallAnywhere SAB within a Docker container
- Easier Merges of Project Files
- Configure the Panel: Choose File
- Addition of a New Variable Containing Invocation User Details

Option to Turn-Off the Application of File Attributes

Now in InstallAnywhere 2020 SP2, the options has been added to handle File Attributes changes for Installer, Uninstaller, and Launchers. The following fields are added:

- **Apply Installer File Attributes** - This field is added under **Project > Platforms > Windows** section for all project types. By default, the value is set to `Yes`. The value can be set to `No` to stop File Attributes changes for windows Installer.
- **Apply Uninstaller/Launcher File Attributes** - This field is added under **Project > Platforms > Windows** section for all project types. By default, the value is set to `Yes`. The value can be set to `No` to stop File Attributes changes for Uninstaller and Launchers.



Enable Silent and Console Mode by Default for Basic Template

Previously, the silent and console mode were not selected by default when a new project was created. Now, the silent and console mode are enabled by default while creating a new project.

Note ▪ *This change was tracked in issue IOJ-2109343.*

Running InstallAnywhere SAB within a Docker container

Added instruction to install SAB build as well as register the license within a Docker container. For more detail, see Knowledge Base article.

Note ▪ *This change was tracked in issue IOJ-2092582.*

Easier Merges of Project Files

Modified the format and design of the project files to allow the easier merges of project files from the different branches in source control.

Note ▪ *This change was tracked in issues IOJ-1995786 and IOJ-1560334.*

Configure the Panel: Choose File

Previously, all the files were displayed in the Choose File panel. Now, you can configure the Panel to filter by file type and display only a specific file type.



***Note** ▪ This change was tracked in issue IOJ-1925434.*

Addition of a New Variable Containing Invocation User Details

Added a new variable “ia.mac.invoked.user” to store the user name of the invoked user who launched the authenticated installer, before the installer is elevated to run as the root user.

This change was tracked in issue IOJ-1739950.

What’s New in InstallAnywhere 2020 SP1

InstallAnywhere 2020 SP1 includes the following new features and enhancements:

- Advanced JRE Handling for Version Upgrades
- Horizontal/Vertical Scrolling Available on Custom Panels

Advanced JRE Handling for Version Upgrades

If you are creating an upgrade installer that contains a JRE with a different processor type or version than the previous version of the installer, problems could occur during installation of that JRE.

In InstallAnywhere 2020 SP1, a new option has been added to prevent those problems from occurring: the **Advanced JRE handling for Version Upgrades** option on the **Project > Installer Settings** tab of the **JVM Settings** view.

Selecting the **Advanced JRE handling for Version Upgrades** option will enable Maintenance Mode and Instance Management Panels to display the existing instances of the same product across both 32-bits and 64-bits. If this option is not selected, the panels will display only instances which correspond to the bits of the upgrade installer.

The following are scenarios when you should select this option:

- **New installer has a JRE with a different processor type than previous**—For example, you are currently building App A V2 with a 64-bit JRE version X, but App A V1 had a 32-bit JRE version X.
- **New installer has a JRE with a different version than previous**—For example, you are currently building App A V2 with a 64/32-bit JRE version X, but App A V1 had a 64/32-bit JRE version Y.

Horizontal/Vertical Scrolling Available on Custom Panels

Using InstallAnywhere’s custom code API enables you to create custom panels where necessary. In InstallAnywhere 2020 SP1, you can now enable vertical or horizontal scrolling on custom panels.

To specify vertical or horizontal scrolling on a custom panel, select the **Enable Vertical Scroll** or **Enable Horizontal Scroll** option in the **Properties Customizer** for that panel on the **Pre-Install** view of the **Sequence** page.

What's New in InstallAnywhere 2020

InstallAnywhere 2020 includes the following new features and enhancements:

- Support for Java 13
- Mac OS X Notarization
- Support for Digital Signature using Windows Store
- JVM Specification Files for Java 10 and 11
- HTTPS Protocol Option in Download File Action
- Warning for an Invalid Java VM
- Support for InstallAnywhere Registration and License

Support for Java 13

InstallAnywhere 2020 now supports the latest version of Java, Version 13.

Using InstallAnywhere 2020, you can:

- Create Java 13 VM packs
- Create installers which uses bundled Java 13 VM pack
- Create installers that can detect a Java 13 on the host and run automatically

Mac OS X Notarization

Starting in macOS 10.14.5, all new or updated kernel extensions and all products from developers new to dispersing with Developer ID must be notarized so as to run. Starting in macOS 10.15, by default, the notarization is required for all products.

Notarization likewise ensures your users if your Developer ID signing key is uncovered. The notary service keeps up a review trail of the product appropriated utilizing your signing key.

In InstallAnywhere 2020, you can automatically notarize your application with ease. InstallAnywhere now supports notarizing macOS or OS X–based installers with a Developer ID Application certificate during the build time.

The process of notarizing your authentication wrappers, your installers, and your uninstallers varies, depending on whether you are performing the notarizing step at build time on the InstallAnywhere build machine or on a separate designated notarization machine.

You can specify whether and how you want InstallAnywhere to notarize your OS X–based installer at build time. If you notarize the installer, end users can download your installer from outside the Mac App Store and install the product without being blocked by the Gatekeeper. The App NOTARIZATION settings in this area are:

- **Notarize the Generated Installer**
- **Developer Username**
- **Developer App Specific Password**
- **Notarization Response Timeout(min)**

- **Notarization Process Delay Interval(sec)**

Support for Digital Signature using Windows Store

InstallAnywhere 2020 provides the Certificate Selection dialog box to specify which certificate you want to use to sign your files. InstallAnywhere lets you choose between the following options:

- You can specify the .pfx certificate file on your machine that you want to use for signing.
- You can reference a certificate store that contains the certificate that you want to use for signing.

JVM Specification Files for Java 10 and 11

Now in InstallAnywhere 2020, you have JVM specification files available for Java 10 and 11 in the default folder (IA_HOME/resource/jvms).

HTTPS Protocol Option in Download File Action

You can now use the Download File action to download a file during installation using HTTPS protocol. You specify information regarding this action on the Download File Action customizer.

Warning for an Invalid Java VM

Previously, there was no warning/caution message displayed while selecting a Java VM which was not valid for the InstallAnywhere installation.

In this release, there will be an error message when you select a Java VM that is not on the valid VM list from the Choose Java VM panel.

Support for InstallAnywhere Registration and License

Previously, there was no message whether or not InstallAnywhere was registered or licensed when building a project.

Now in InstallAnywhere, you have a command output that will display that determines whether your copy of InstallAnywhere is registered and licensed while building the project.

What's New in InstallAnywhere 2018 SP1

For information about new features and enhancements added in InstallAnywhere 2018 SP1, refer to the following sections:

- Configure Installer Updates
- Support for Java 11
- Improved Designer Search & Expanded Search Functionality
- Ability to Provide a Simple Way to get a Feature's Long Name from its Short Name
- Ability to Hold the Path for Global InstallAnywhere Registry Location
- Allow Alignment of Input Fields for Get User Input Panel
- Ability to Support tar.gz and tar.z

- Ability to Read REG_MULTI_SZ Values from Get Windows Registry Entry Action
- Option to Set Size field via InstallAnywhere Project for Add or Remove Programs Applet on Windows
- Ability to Set Installer Title Image on a Linux and MAC Machine for Windows Build Target

Configure Installer Updates

In InstallAnywhere 2018 SP1, you can now enable and configure a software downloadable update. The end user can either download a newer version or skip and proceed with the current installation process. A new Installer Updates option has been added to the Upgrades view where you can specify these downloadable update settings.

Enable Downloadable Update Support

You can now enable updates for the end users to see the downloadable update support. The installation has the ability to check for downloadable updates.

To enable updates, select the Enable Updates option on the Installer Updates option of the Upgrades view.

Specify Download URL

You can now enter the download URL in the Download URL field on the Enable Updates option on the Installer Updates option of the Upgrades view.

The Download URL is a web hosted location where iaupdate.xml and downloadable updates are hosted.

Enable Force Install

You can now use this option for the end users to forcibly download and install the updates (if available).

Configure Prompt Message

You can now customize the message and provide a title for the message that is displayed during the installation to prompt the end users.

Specify Prompt Message Title

You can now specify the title of the Upgrade Prompt message for the end users.

Specify Prompt Message

You can enter the customized message in the Message field on the Upgrade Prompt to the Installer Updates option of the Upgrades view. You can either provide a new message or use default localized string.

Support for Java 11

InstallAnywhere 2018 SP1 now supports the latest version of Java, Version 11. Open JDK 11 VM packs that are bundled for Windows, Linux and MAC target platforms.

Using InstallAnywhere 2018 SP1, you can create installers that support Java 11, including:

- Creating Java 11 VM packs
- Creating installers which uses bundled Java 11 VM pack

Creating installers that can detect a Java 11 on the host and run automatically

Improved Designer Search & Expanded Search Functionality

Stronger searching finds what you need fast - search your entire project for text or variables and navigate automatically to the referenced location.

Now, InstallAnywhere supports expand search functionality for String Value and variable. If the partial search is unchecked, the string search will search for the whole text. If the partial search is checked, then the parts of strings & variables will also be searched.

The “Variable Search” which is a new checkbox has been added in InstallAnywhere.

- If the Variable Search check box is checked, it will enclose with a dollar sign symbol ($) before and after the variable to search for a variable.
- If the Variable Search check box is unchecked, it will not be enclosed with a dollar sign symbol ($) before and after the text to search for a string and variable.
- The results of the search shows in a tree structure with the total number of strings. The Items that are associated with the string are listed below, and grouped by category:
    - Pre-Install
    - Install
    - Post-Install
    - Pre-Uninstall
    - Post-Uninstall
- The Search result node displayed navigates to the corresponding action which is then displayed in a tree structure.

In InstallAnywhere 2018, you used to replace all instances of a variable with the replacement variable.

Now in InstallAnywhere 2018 SP1, you can replace the strings with all instances. Search for a string and select ‘Replace All’. Enter the string which you want to replace with.

Ability to Provide a Simple Way to get a Feature's Long Name from its Short Name

A new read only variable $CHOSEN_INSTALL_FEATURE_LIST_LONG$ is added under the InstallAnywhere Variables list which contains the long names of the installed Features. This variable can be used to get the long name of the feature from the feature short name.

Ability to Hold the Path for Global InstallAnywhere Registry Location

A new read only variable $IA_GLOBAL_REG_LOCATION$ is added under InstallAnywhere Variables list which holds the path for global InstallAnywhere registry location. This variable can be used to determine where the global registry is present on the target machine.

Allow Alignment of Input Fields for Get User Input Panel

The Selection box for the ‘Control Alignment’ has been added in Get User Input Panel Advanced-> Configure Selection of Text Field. By default, the Alignment selection will be ‘Left’ and the alignment changes applies to Text Fields.

There are two possible values for the text alignment controls that is ‘Left and ‘Right’.

Ability to Support tar.gz and tar.z

The Expand Tar action now also supports expanding of tar.gz and tar.z files.You can use the Expand Archive (TAR) action to expand a TAR, .gz, .Z archive file on the target system.

Ability to Read REG_MULTI_SZ Values from Get Windows Registry Entry Action

There is an ability to read multi string REG_MULTI_SZ values from the Windows Registry and store the value in an InstallAnywhere Variable(s).

Option to Set Size field via InstallAnywhere Project for Add or Remove Programs Applet on Windows

This new field 'Estimated size(in MB)' is added under Projects-> Platforms->Windows section for all project types. By default 'Estimated size(in MB)' field value will be blank. You can enter an estimated size of any value > 0 (Zero) and <= 4194303(4TB).



Note ▪ *Cannot enter floating/decimal value for an estimated size.*

Ability to Set Installer Title Image on a Linux and MAC Machine for Windows Build Target

The Installer Title image setting is now enabled for Linux and MAC platform so that it can be used to build windows target from both Linux and Mac.

What's New in InstallAnywhere 2018

For information about new features and enhancements added in InstallAnywhere 2018, refer to the following sections:

- Perform Open Source Risk Assessment and Vulnerability Scan with FlexNet Code Aware
- All InstallAnywhere Features Now in a Single Edition
- Redesigned InstallAnywhere Advanced Designer User Interface
- Enhanced Upgrade Installers Now Accommodate Maintenance Mode and Instance Management
- Support for Java 9
- Ability to Customize the Windows File Properties of an Installer Executable File for a Windows Build Target
- Ability to Customize the Get Info Property Value for Copyright Field for an OS X / macOS Build Target
- Ability to Add Checkbox to Install Complete Panel to Launch Target Application or README File
- Option to Disable Logging Upon User Cancellation During Pre-Install Sequence
- Execute Script/Batch File Action Run in Console Mode Now Writes Action Type and Identifier to Console
- In Console Mode, User Can Now Move Forward Without Scrolling Through the Entire License Agreement
- Ability to Set Compiler Flags, Such as SafeSEH, for Executables Installed on Windows
- ANT Version Associated with Execute ANT Script Action Updated to ANT 1.9.9

- New Compare Versions Rule
- Expand/Collapse Buttons on Sequence Page Views
- Get Password Action Console Panel Now Available in Pre-Uninstall and Post-Uninstall Phases
- New User-Defined Range of Installer Exit Codes
- New Variable to Store Response File Location
- Launching Build from User Interface Now Leaves Current Project Unchanged

Perform Open Source Risk Assessment and Vulnerability Scan with FlexNet Code Aware

There is an increasing need to understand the licensing obligations of all open source software components being bundled with a software application. At the same time, users are getting increasingly worried about the vulnerabilities that these open source components bring in. With InstallAnywhere's new FlexNet Code Aware scanning feature, users will get the insight into the licensing obligations and vulnerabilities associated with all open source components.

InstallAnywhere now includes full integration with FlexNet Code Aware, an automated open source risk assessment and package discovery solution that enables you to quickly scan your products for security and intellectual property (IP) compliance risk.

- Supported File Types
- Running FlexNet Code Aware from Within InstallAnywhere
- Running FlexNet Code Aware as a Standalone Product
- Reading the FlexNet Code Aware Report
- More Information



Note ▪ *FlexNet Code Aware is automatically activated when you activate InstallAnywhere. However, the FlexNet Code Aware license expires one year from the date that it is first launched.*



Important ▪ *FlexNet Code Aware is not supported on 32-bit machines. A 64-bit operating system is required.*

Supported File Types

FlexNet Code Aware supports analysis of the following files:

- Java Packages
- Node Packages
- Nuget Packages
- RPM Packages
- Ruby Packages
- EXE & DLL Files

Security vulnerabilities are looked up against the National Vulnerability Database (NVD).

Running FlexNet Code Aware from Within InstallAnywhere

To run FlexNet Code Aware from within InstallAnywhere, click Run **FlexNet Code Aware Analysis** on the InstallAnywhere **Tools** menu.

When FlexNet Code Aware completes the scan of your project, the Results Summary view opens, displaying the number of files scanned, and the number of open-source packages and vulnerabilities found.

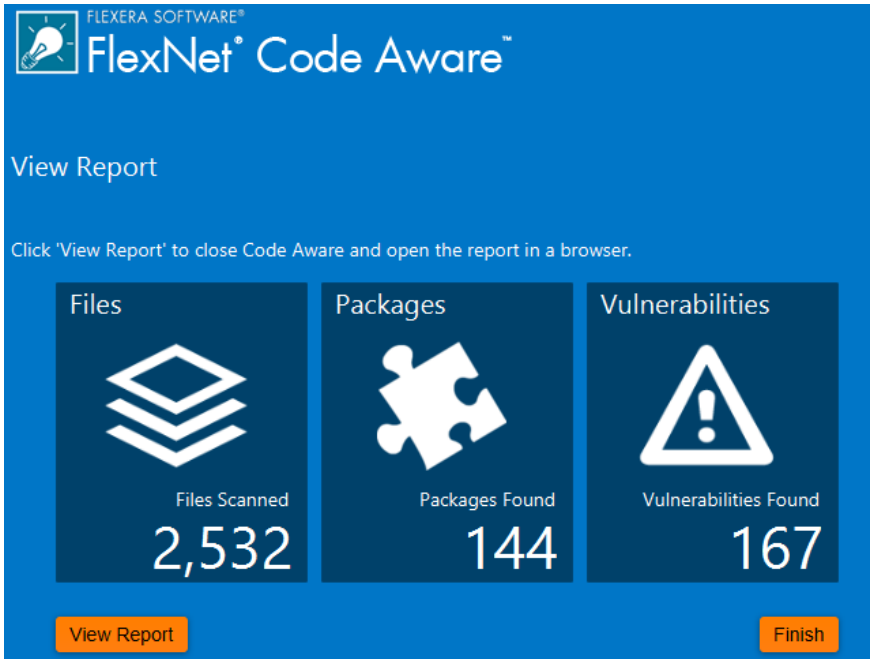


Figure 1-8: FlexNet Code Aware Results Summary

When you click the **View Report** button, a full report is displayed.

Running FlexNet Code Aware as a Standalone Product

Rather than scanning the files in the open InstallAnywhere project, you can also launch FlexNet Code Aware as a standalone product and select a directory of files to scan.

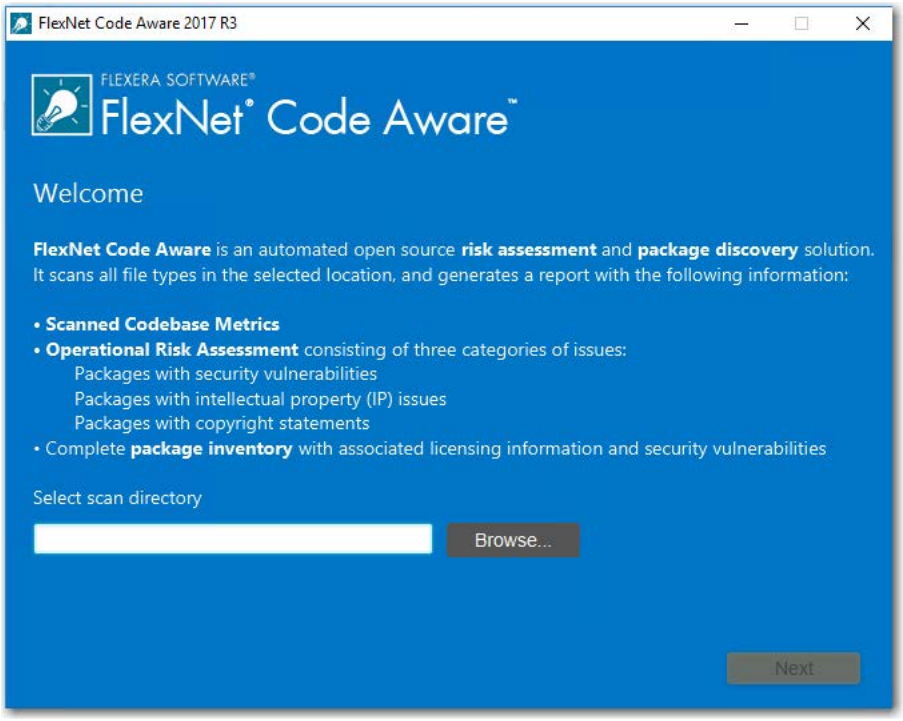


Figure 1-9: FlexNet Code Aware Welcome Panel

Just click **Browse**, select the directory that you want to scan, and then analysis will begin.

Reading the FlexNet Code Aware Report

When you click **View Report** on the Results Summary screen, the full FlexNet Code Aware report opens, consisting of an **Initial Summary** view and a **Package Inventory** view.

Initial Summary View

The **Initial Summary** view presents the user with a scan summary, and assessments of operational risk, security vulnerability exposure, and license exposure.

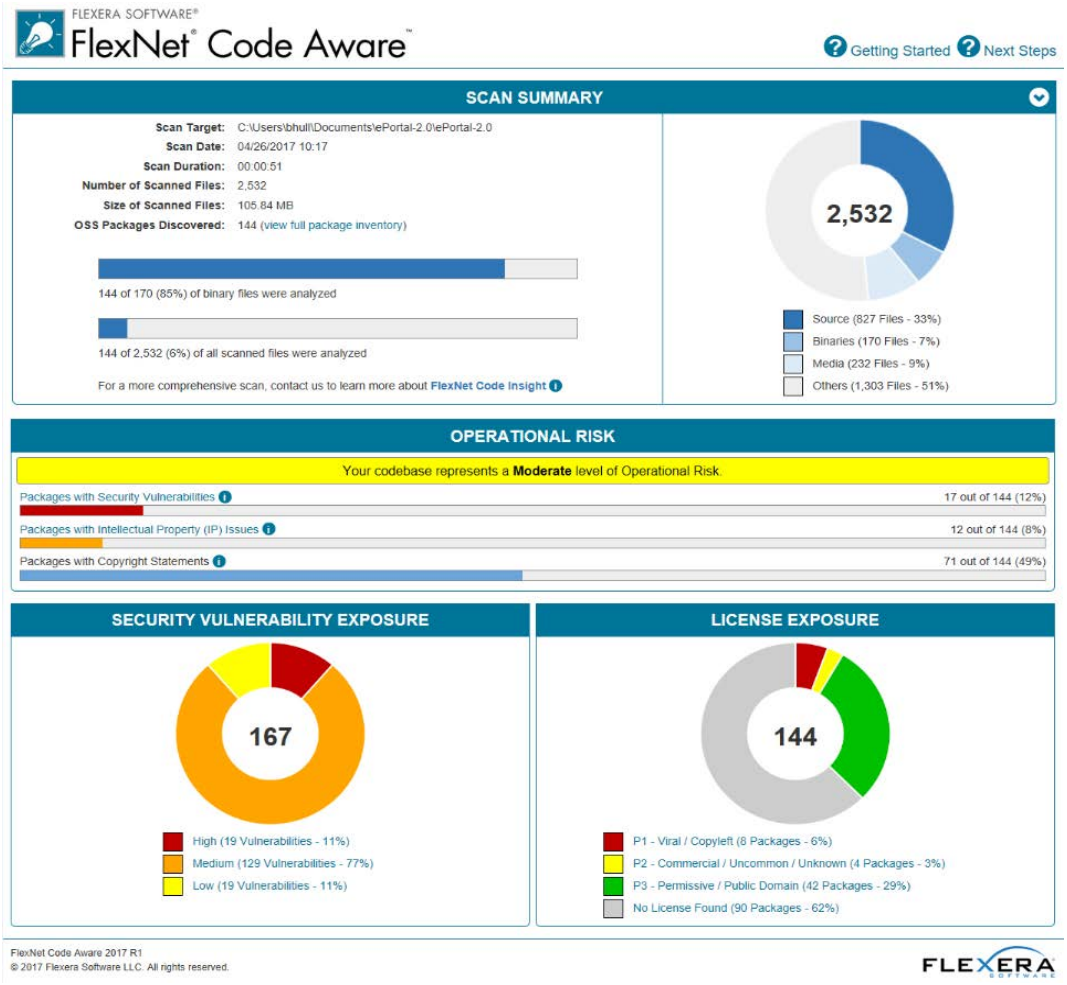


Figure 1-10: FlexNet Code Aware Initial Summary View

The FlexNet Code Aware Initial Summary View displays the following information:

- **Scan Summary**—This section provides details regarding the codebase that was scanned, including a breakdown of file types, percent of files analyzed, and number of findings.
- **Operational Risk**—This section provides a composite risk rating based on the combination of packages with Intellectual Property (IP) issues and packages with Security Vulnerabilities.
- **Security Vulnerability Exposure and License Exposure**—These sections provide a breakdown of the types and categories of identified issues.

Package Inventory View

The **Package Inventory** view, available by clicking **View full package inventory** in the **Scan Summary** section, provides a complete list of discovered open source and third-party packages with associated licenses, security vulnerabilities, dependencies, and detected copyright statements.

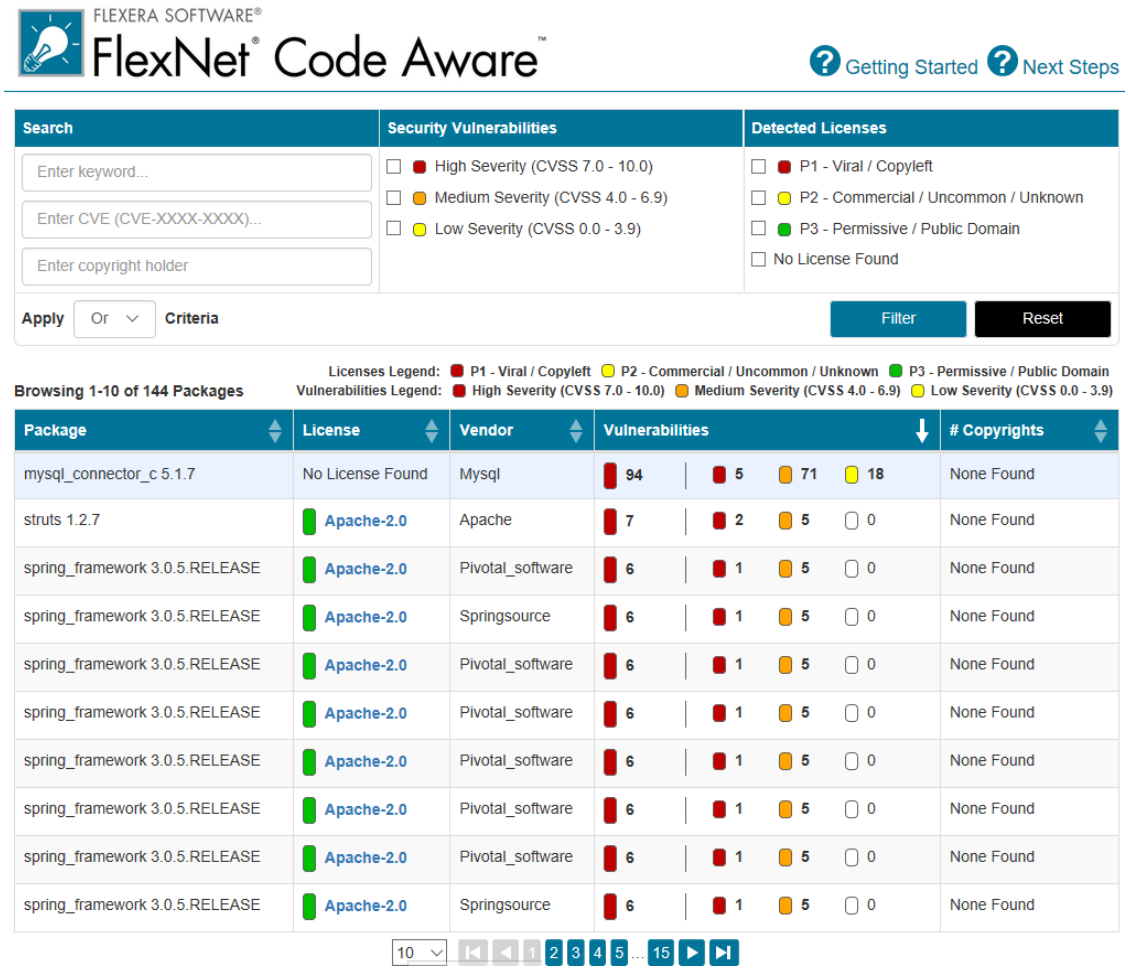


Figure 1-11: FlexNet Code Aware Package Inventory View

The **Package Inventory** view provides filters that you can use to execute targeted queries to refine the list to various package types of interest.

To view additional package details, click a vulnerability count listed in the **Vulnerabilities** column of the package you want to review:

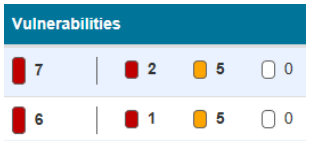


Figure 1-12: Vulnerabilities Column

The **Vulnerabilities Detail** page opens (covering a portion of the Package Inventory view), and displays detailed information on the selected package.

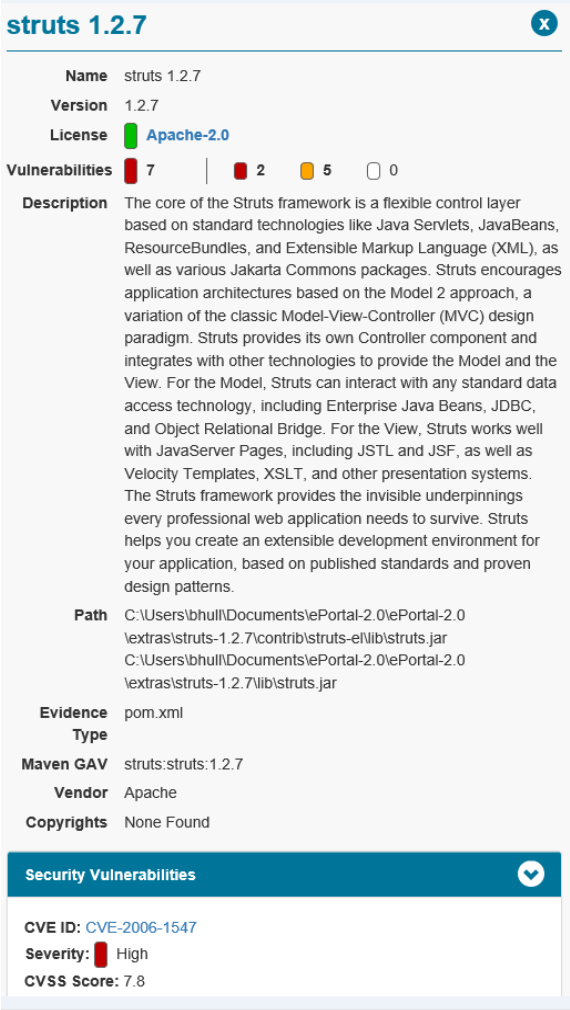


Figure 1-13: Vulnerabilities Detail

More Information

For detailed information on using FlexNet Code Aware, see *Using FlexNet Code Aware to Perform Open Source Risk Assessment* in the InstallAnywhere Help Library.

All InstallAnywhere Features Now in a Single Edition

Previous versions of InstallAnywhere were available in two different editions, Professional and Premier, with additional features included in add-on Virtualization and Cloud packs. Starting with InstallAnywhere 2018, all features are included in a single edition.

After upgrading to InstallAnywhere 2018, you will have access to all features, including these features that were previously only available in Premier Edition and with the add-on packs:

- **Extensive run-time language support**—InstallAnywhere includes default run-time strings in 32 supported languages, and gives you the ability to include custom localized splash screens in your installations.
- **Built-in support for creating upgrades**—You can create upgrades that uninstall earlier versions of the product if present before installing the new version.

- **Instance management support**—You can create installers that let end users install multiple instances of a product on the same machine.
- **Predefined and custom source path variables**—You can use predefined source path variables in your installer, such as IA_HOME, $IA_PROJECT_DIR$, and $USER_HOME$. You can also use custom source path variables.
- **Build time variables**—You can configure your project to use build-time variables, variables that have their values set at build time.
- **Flexible build configuration management**—You can define tags to bundle related sets of actions, panels, features, and components, and assign the tags to the appropriate items in your project. Then you can associate the tags with specific build configurations to include or exclude items from builds.
- **Flexible build target management**—You can create different build targets for each platform that your product supports.
- **Ability to add custom GUI panels**—You can design your own custom panels that guide end users through the selection or completion of various UI elements such as text boxes, check boxes, and lists.
- **Merge module support**—You can include merge modules in your installer project.
- **Project automation APIs**—InstallAnywhere includes project automation APIs that let you design, modify, build, and test an installation from an InstallAnywhere project via Java code.
- **Docker support**—You can use InstallAnywhere to configure and build Docker images, and deploy Web applications to the cloud as Docker images that are run as Docker containers.
- **Virtual appliance support**—You can build enterprise-ready and cloud-ready multi-tier virtual appliances for VMWare ESX based on your existing InstallAnywhere projects.
- **Ability to share components with other products**—You can optionally identify a component in your project as shared. At run time, if this type of component is not already present on a target system, the installer adds it. If it is already present, the installer registers it as shared.

All existing customers will be upgraded to the full-featured, single edition.

Redesigned InstallAnywhere Advanced Designer User Interface

The user interface of InstallAnywhere 2018's Advanced Designer has been redesigned and updated to provide a cleaner, more modern look, with a new black / gray / blue color scheme.

A user-friendly tabbed interface improves the ease of switching between views. Buttons on the **Sequence** tab have been repositioned to a more logical location above the Action lists.

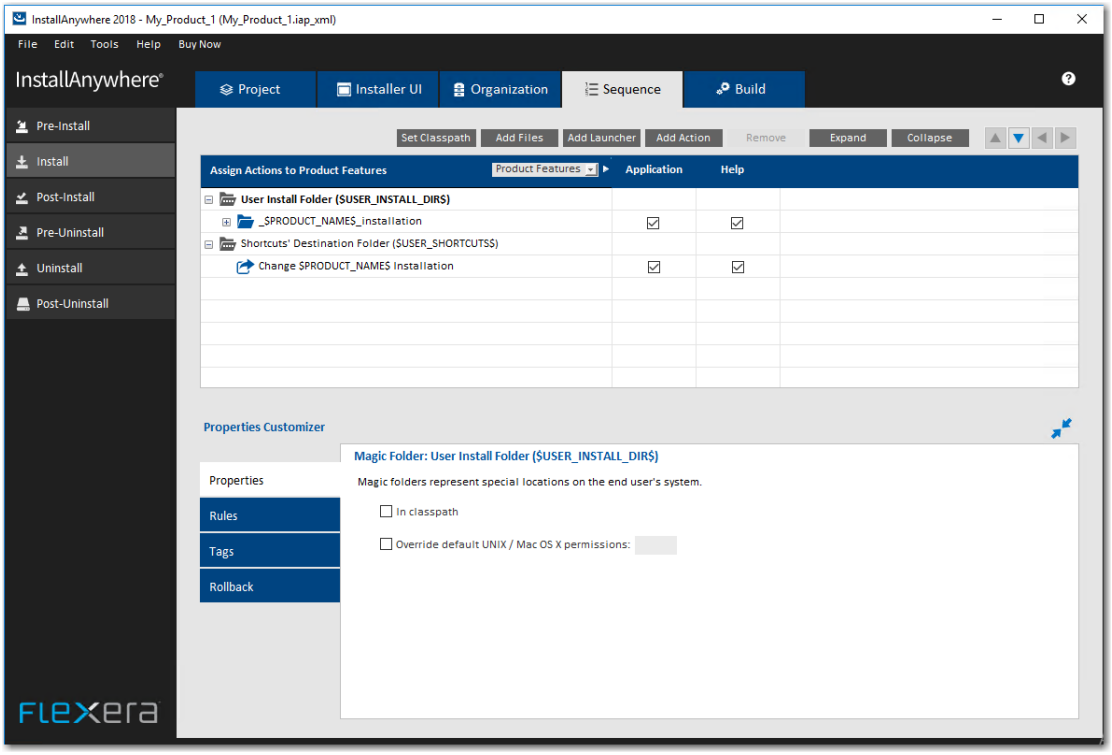


Figure 1-14: Redesigned Advanced Designer User Interface of InstallAnywhere

The **Create/Open Project** dialog box that opens when you launch InstallAnywhere has also been redesigned to be more user friendly.

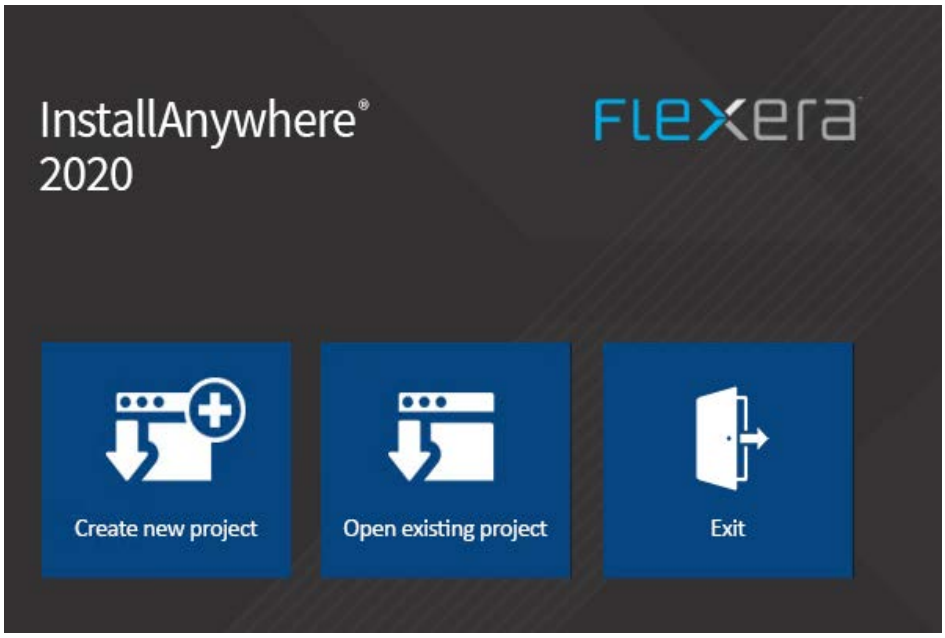


Figure 1-15: Redesigned Create/Open Project Dialog Box

When you click **Create new project** on the **Create/Open Project** dialog box, the redesigned **Create New Project** dialog box opens, prompting you to select the desired template to use, enter a name for the project, and select a location for the project.

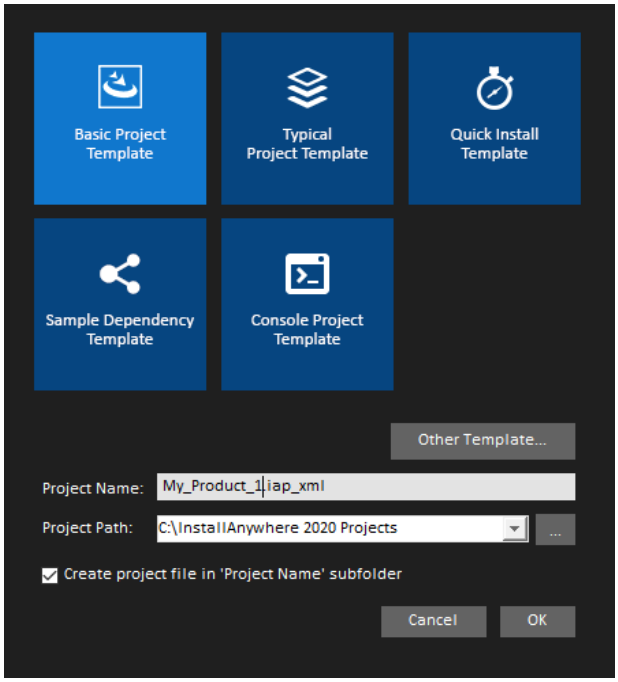


Figure 1-16: Redesigned Create a New Project Dialog Box

Enhanced Upgrade Installers Now Accommodate Maintenance Mode and Instance Management

In InstallAnywhere 2018, the behavior of upgrade installers has been enhanced to accommodate maintenance mode and instance management.

Previously, upgrade installers did not support maintenance mode or instance management. Upgrade installers could only upgrade an earlier version of an application (by uninstalling the previous version and installing the new version), or, if an earlier version does not exist, just install the new version.

In InstallAnywhere 2018, you can use an upgrade installer to add or remove features of a previously installed version as well as repair broken installations. You can also use an upgrade installer to specify whether multiple instances of a product can be installed on the same machine. At runtime, the user can choose which instance of the product to upgrade or perform maintenance on.

Note ▪ *This feature is not supported when performing a silent upgrade.*

Support for Java 9

InstallAnywhere 2018 now supports the latest version of Java, Version 9. Using InstallAnywhere 2018, you can create installers that support Java 9, including:

- Creating Java 9 VM packs
- Creating installers that bundle a Java 9 VM pack

- Creating installers that can detect a Java 9 VM on the host and run automatically

Ability to Customize the Windows File Properties of an Installer Executable File for a Windows Build Target

In previous releases, when you viewed the **Details** tab of the Windows **Properties** dialog box for an InstallAnywhere installer `.exe` file, information specific to the version of InstallAnywhere used to create the installer was displayed instead of information specific to the product being installed:

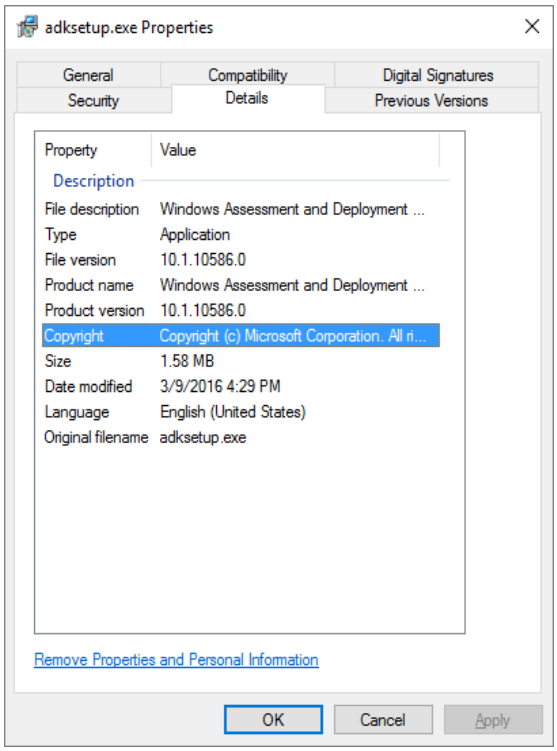


Figure 1-17: Details Tab of Windows Properties Dialog Box

The following hard-coded values were displayed for all Windows `.exe` installers created using InstallAnywhere:

- **File description**—Displayed `InstallAnywhere Self-Extractor`.
- **File version**—Displayed the version of InstallAnywhere that was used to create the installer.
- **Product version**—Displayed the version of InstallAnywhere that was used to create the installer.
- **Copyright**—Displayed InstallAnywhere's copyright information.

For Windows installer `.exe` files created using InstallAnywhere 2018, the project-specific values entered on the on the **Project > General Settings** page of the InstallAnywhere project file are now displayed on the **Details** tab of the Windows **Properties** dialog box, instead of the previously hard-coded values.

Ability to Customize the Get Info Property Value for Copyright Field for an OS X / macOS Build Target

In previous releases, when you viewed the **Copyright** field on the **Get Info** dialog box for a Mac OS X build target for an installer targeting a OS X / macOS operating system, the copyright information specific to the version of InstallAnywhere used to create the installer was displayed instead of information specific to the product being installed:

In InstallAnywhere 2018, you can now customize the **Get Info** property value for the **Copyright** field for a Mac OS X build target.

The **Copyright** and **Version** fields can be edited on the InstallAnywhere **Project > General Settings** view under **Product Information**. The values entered here will be displayed on the dialog box that opens in OS X / macOS operating systems when you select the **Get Info** option.

Ability to Add Checkbox to Install Complete Panel to Launch Target Application or README File

In InstallAnywhere 2018, a new option has been added to configure the **Install Complete** panel to display a checkbox to prompt the user to launch a README file and/or launch a target application. The application could be either the main target application of the product that was just installed, or a utility installed alongside the product.

Option to Disable Logging Upon User Cancellation During Pre-Install Sequence

A new option has been added to enable you to disable logging if the user cancels an installation during the Pre-Install sequence.

To disable logging, open the **Project > General Settings** view of the Advanced Designer, scroll down to the **Log Settings** group, and set the **Skip Logging in Pre-Install** option to **Yes**.

Execute Script/Batch File Action Run in Console Mode Now Writes Action Type and Identifier to Console

When run in console mode, an InstallAnywhere installer displays a **Please Wait** message during installation for each **Execute Script/Batch File** action and **Execute Command** action included in the **Install Sequence** or **Post-Install Sequence**.

In InstallAnywhere 2018, an installer running in console mode now identifies each action that displays a **Please Wait** message by also displaying the type of action (**Execute Script/Batch File** or **Execute Command**) and an action title/name.

In Console Mode, User Can Now Move Forward Without Scrolling Through the Entire License Agreement

In InstallAnywhere 2018, a new option has been added to enable users using a console mode installer to proceed to the next step without scrolling through the entire license agreement.

When you add a License Agreement console screen to an installer on the **Sequence > Pre-Install** page, you can now select the **Do not force user to scroll through license agreement** option.

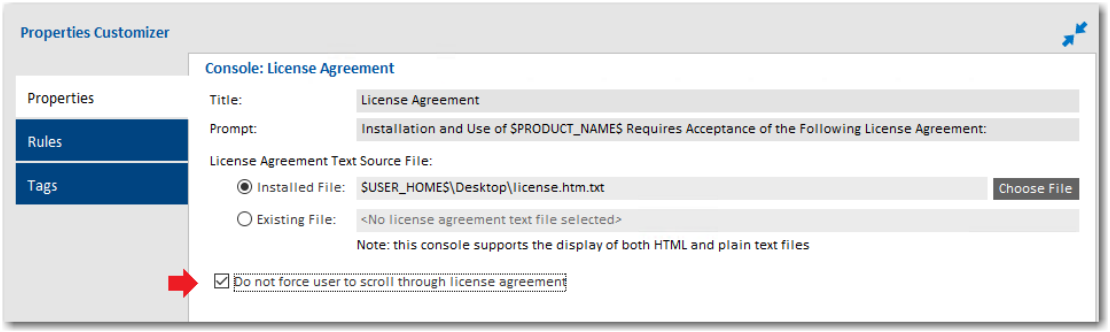


Figure 1-18: Properties Tab of Console: License Agreement Console Screen

When this option is selected, users using a console mode installer will be able to accept or reject the license agreement without scrolling through all of the text of the license agreement. The following text will be displayed:

```
PRESS ANY KEY TO CONTINUE TO READ LICENSE AGREEMENT OR PRESS '0' to ADVANCE TO END
```

Note ▪ *To control whether a user using a **panel** mode installer can proceed to the next step without scrolling through the entire license agreement, use the **Force user to scroll through license agreement** option on the **Properties** tab of the **Panel: License Agreement** > **Properties** tab.*

Ability to Set Compiler Flags, Such as SafeSEH, for Executables Installed on Windows

In previous releases, executables installed on Windows did not meet security requirements because compiler flags, such as SafeSEH, were not set. In InstallAnywhere 2018, compiler flags are now set properly for executables installed on Windows. These executables now pass the SafeSEH check.

Support has been added to LaunchAnywhere (32-bit and 64-bit) and SelfExtractor (32-bit and 64-bit) projects for the following flags:

- **/SAFESEH**—This is applicable only for 32-bit target and not applicable for 64-bit targets. Therefore, for all 64-bit build configurations, we have set this flag to NO.
- **/HIGHENTROPYVA**—This is applicable only for 64-bit executable images. This is not applicable for 32-bit executable images.

ANT Version Associated with Execute ANT Script Action Updated to ANT 1.9.9

In InstallAnywhere 2018, the ANT version associated with the **Execute ANT Script** action has been updated to ANT 1.9.9 for both InstallAnywhere development and InstallAnywhere runtime.

ANT 1.9.9 supports Java 1.5 and above. For more information, see:

http://ant.apache.org/manual/install.html
https://ant.apache.org/faq.html

New Compare Versions Rule

InstallAnywhere 2018 now includes a new **Compare Versions** rule that enables you to specifically compare two version numbers during an installation. This new rule is displayed on the **Choose a Rule/Expression** dialog box that opens when you click Add Rule on the **Organization** or **Sequence** page.

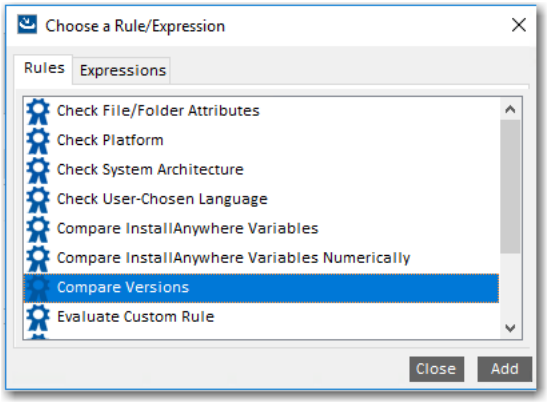


Figure 1-19: New Compare Versions Rule

When you add a **Compare Versions** rule, you are prompted to enter two operands and an operator in the Compare Versions **Properties Customizer**. Both operands may be expressed as either an InstallAnywhere variable being resolved (such as $VARIABLE1$) or as a literal version number string (such as 1.0.0.0).

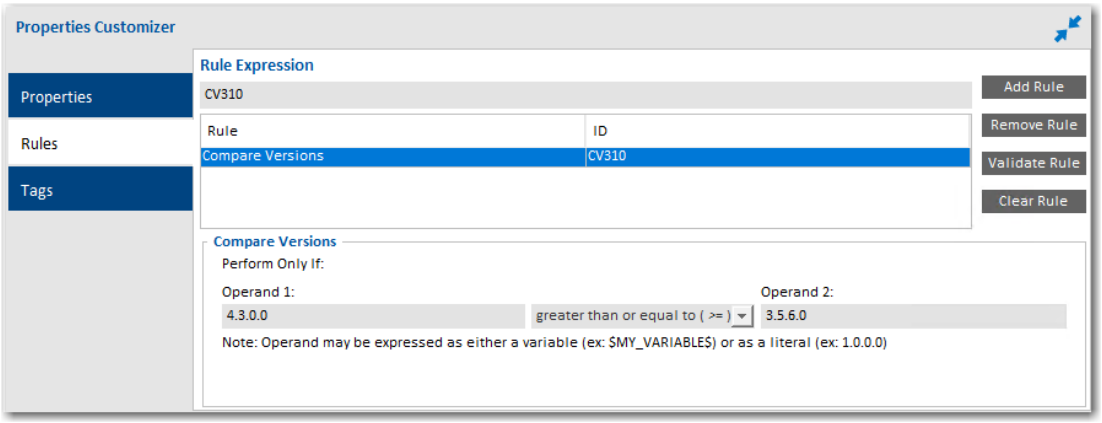


Figure 1-20: Compare Versions Rule Properties Customizer

Expand/Collapse Buttons on Sequence Page Views

In InstallAnywhere 2018, an **Expand** and a **Collapse** button have been added to views on the **Sequence** page to quickly collapse or expand the items in the Action List tree.

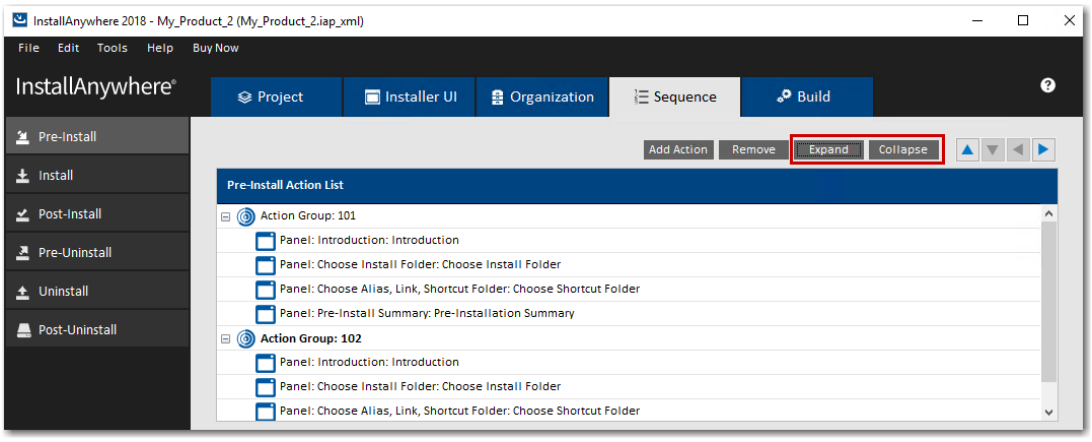


Figure 1-21: New Expand and Collapse Buttons on the Sequence Page

When you click the **Collapse** button, the items in the tree are collapsed into their primary item:

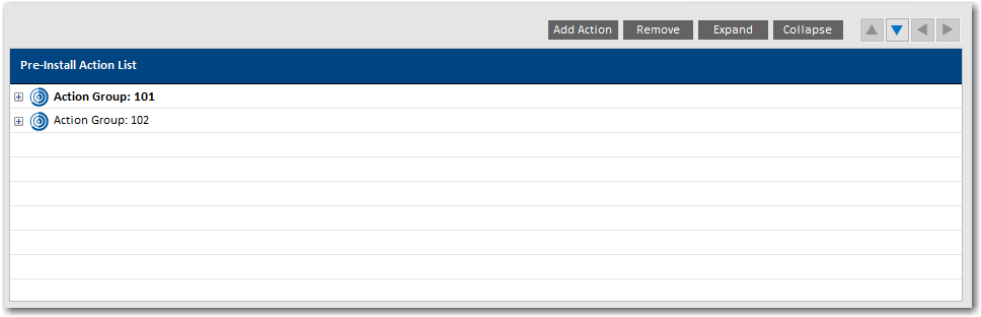


Figure 1-22: Collapsed Items in the Tree

Get Password Action Console Panel Now Available in Pre-Uninstall and Post-Uninstall Phases

Previously, you could not add a **Console: Get Password** action during **Pre-Uninstall** and **Post-Uninstall** tasks for console mode installers.

In InstallAnywhere 2018, you can now add a **Console: Get Password** action during **Pre-Uninstall** and **Post-Uninstall** tasks for console mode installers. You can also add a **Console: Get Password** action during **Pre-Uninstall** and **Post-Uninstall** tasks after adding a merge module project, which previously was not possible.

New User-Defined Range of Installer Exit Codes

InstallAnywhere 2018 now includes a range of user-defined installer exit codes. These can be used to act upon scripts that are included in your installers. The following 20 exit codes were added for customer use:

Table 1-2 ▪

Category	Exit Codes
Linux Exit Codes	43 to 62
Other Exit Codes	10795 to 10814

New Variable to Store Response File Location

A new InstallAnywhere variable named `$IA_RESPONSEFILE_PATH$` has been added to store the complete location of a response file being used by the installer.

Launching Build from User Interface Now Leaves Current Project Unchanged

In previous releases, when you initiated a build from the InstallAnywhere user interface, InstallAnywhere would first save the current project before performing the build. However, if you initiated a build from the command line, a new build copy of the current project would be made (projectBuild.iap_xml) before the build was performed, leaving the original project unchanged.

In InstallAnywhere 2018, the behavior when you initiate a build from the user interface matches the behavior when launching a build from the command line. In both cases, a build copy of the project file is made prior to the build, leaving the original project file unchanged.

What's New in InstallAnywhere 2017 SP1

Refer to the following sections for new features and changes in InstallAnywhere 2017 SP1:

- Integration with FlexNet Code Aware
- Enhancement to Existing GUI Automation Fixture

Integration with FlexNet Code Aware



***Note** ▪ For complete information about FlexNet Code Aware, refer to the FlexNet Code Aware user documentation.*

InstallAnywhere now includes integration with FlexNet Code Aware, an automated open source risk assessment and package discovery solution that enables you to quickly scan your products for security and intellectual property (IP) compliance risk.

The current release of FlexNet Code Aware supports analysis of the following files:

- Java Packages
- Node Packages
- Nuget Packages
- RPM Packages
- Ruby Packages
- EXE & DLL Files

Security vulnerabilities are looked up against the National Vulnerability Database (NVD).

Running FlexNet Code Aware

FlexNet Code requires a separate license from InstallAnywhere. There is also trial/evaluation version. For more information, refer to the FlexNet Code Aware product page of the Revenera website.

To run FlexNet Code Aware from within InstallAnywhere, click **Run FlexNet Code Aware Analysis** from the InstallAnywhere **Tools** menu. This menu option is disabled out if you are not currently in an open InstallAnywhere project.

When FlexNet Code Aware completes the scan of your project, a summary displays showing the number of files scanned, and the number of open-source packages and vulnerabilities found. A **View report** button is provided if you have a fully licensed version of FlexNet Code Aware. For more information about the details provided in this report, refer to Reading the FlexNet Code Aware Report.



Reading the FlexNet Code Aware Report

Note ▪ *The FlexNet Code Aware Report is not available in trial/evaluation mode. A fully licensed version of FlexNet Code Aware is required.*

To view the FlexNet Code Aware Report, click **View report** on the summary dialog that appears after FlexNet Code Aware has scanned your project.

The FlexNet Code Aware report consists of several sections:

- The initial Summary View presents the user with a **Scan Summary**, **Operational Risk** assessment, **Security Vulnerability Exposure,** and **License Exposure**.
  - The **Scan Summary** section provides details regarding the codebase that was scanned, including a breakdown of file types, percent of files analyzed, and number of findings.
  - The **Operational Risk** section provides a composite risk rating based on the combination of packages with Intellectual Property (IP) issues and packages with Security Vulnerabilities.
  - The **Security Vulnerability Exposure** and **License Exposure** sections provide a breakdown of the types and categories of identified issues.

- The Package Inventory View, available by clicking **view full package inventory** in the **Scan Summary** section, provides a complete list of discovered open source and third-party packages with associated licenses, security vulnerabilities, dependencies, and detected copyright statements.

  The Package Inventory View provides filters that you can use to execute targeted queries to refine the list to various package types of interest.

The following figures show the initial Summary View of a sample FlexNet Code Aware Report.

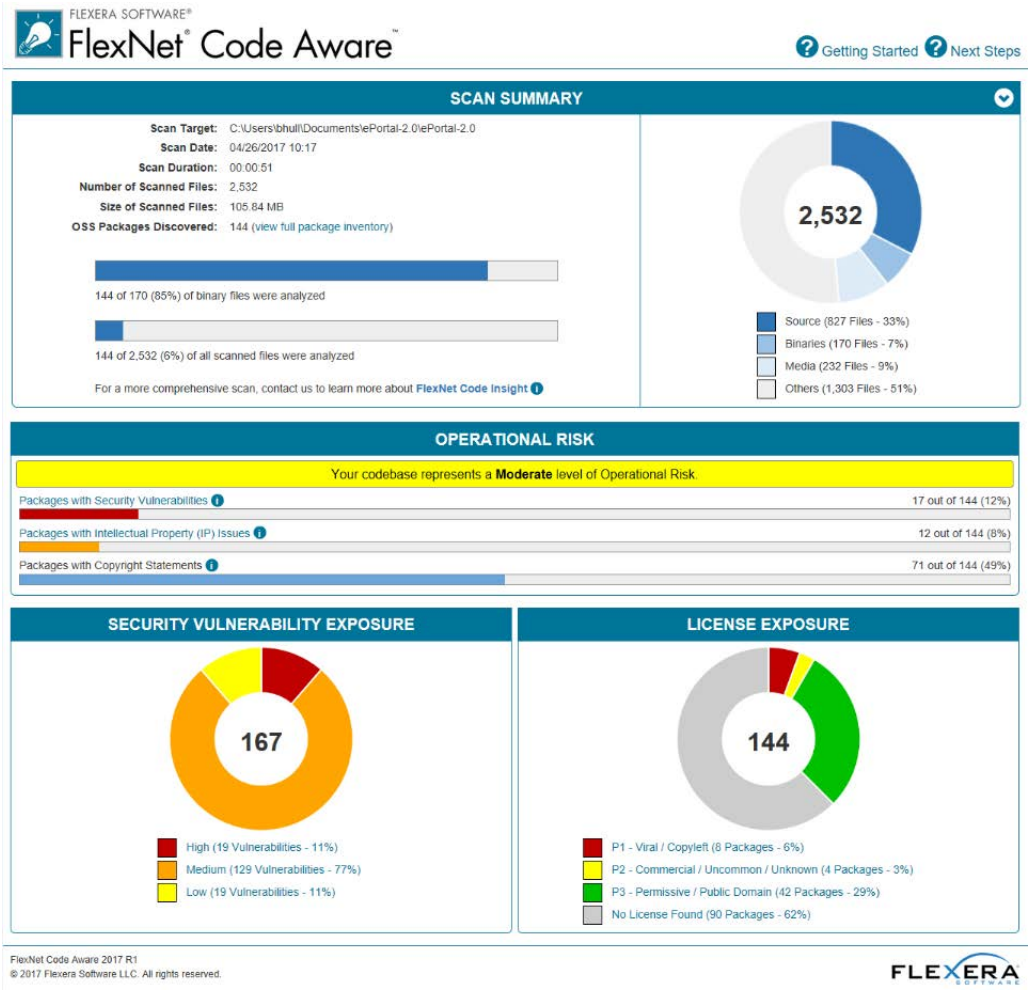


Figure 1-23: FlexNet Code Aware Initial Summary View

The following figures show the Package Inventory View of a sample FlexNet Code Aware Report.

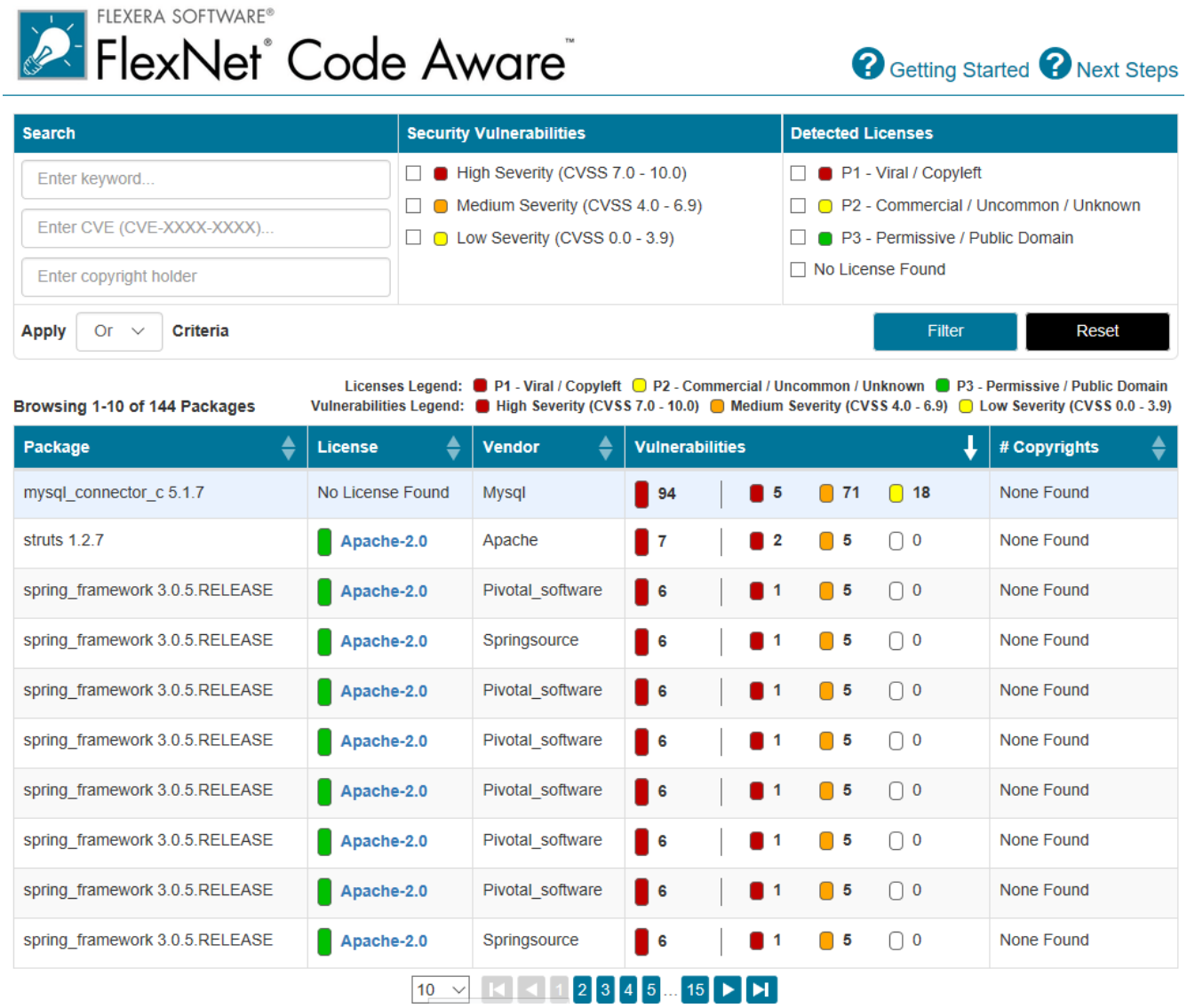
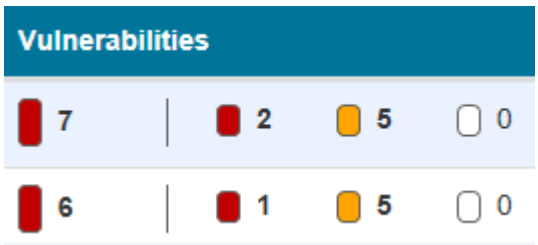


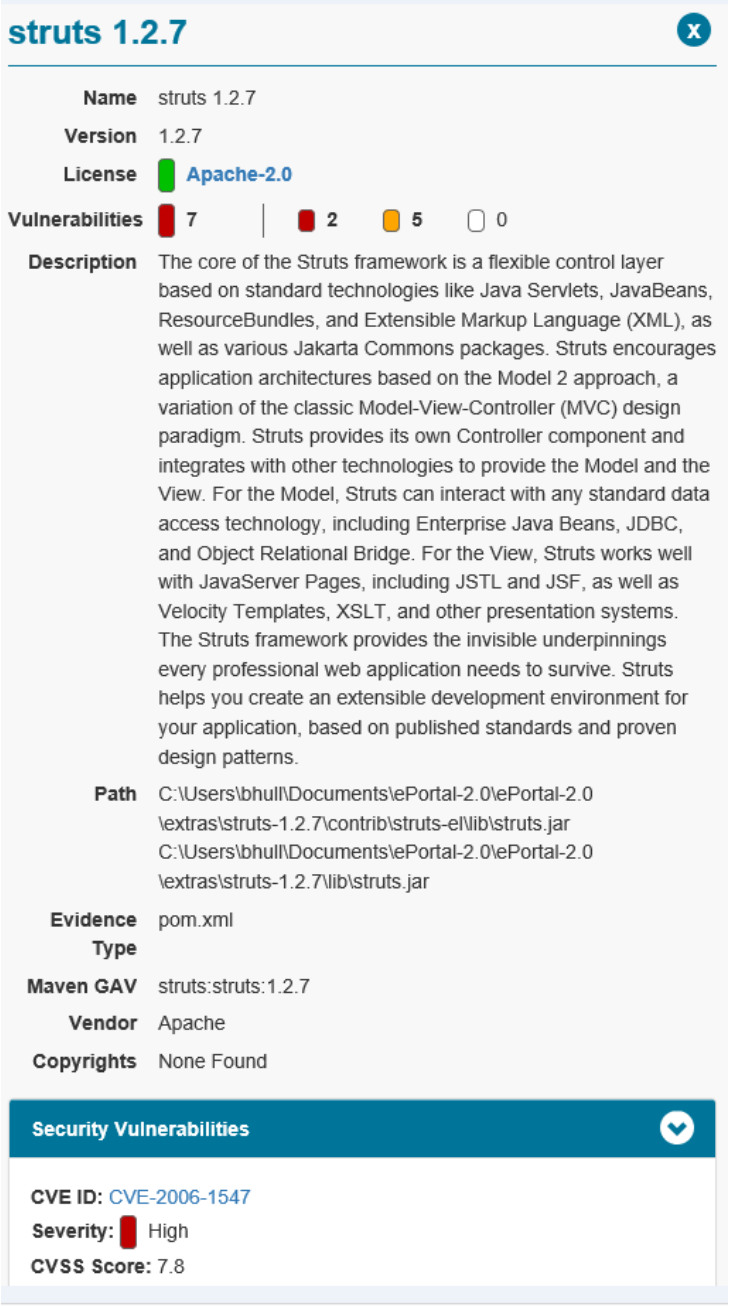
Figure 1-24: FlexNet Code Aware Package Inventory View

Viewing Package Details

Click a vulnerability count listed in the Vulnerabilities column of the Package Inventory report page for each package you want to review:



The **Vulnerabilities detail** page appears, covering a portion of the Package Inventory report:



Enhancement to Existing GUI Automation Fixture

InstallAnywhere now has improved GUI Automation Fixture by adding new APIs for launching binaries; that is, you can now launch Windows/Linux built-in installers in addition to the pure Java installer. The new APIs have ability to interact with various UI components and get UI attributes of the components like labels, text fields, checkboxes, radio buttons, combo boxes, list boxes and buttons.

Parameters that the APIs need can be obtained as an optional tool tip by running the installer with a special command line.

```
install.exe -Diashowtooltip=true
```

For complete information about each of the APIs, refer to the docs available at `IA_HOME/gui-test-auto/javadocs`. For a sample GUI Automation class, refer to your InstallAnywhere build directory at `IA_HOME/gui-test-auto/example/source/com/installanywhere/ia/automation/gui/samples/basic/BasicProjectAutomation.java`.

Supporting platforms: Windows and Linux.

What's New in InstallAnywhere 2017

InstallAnywhere 2017 includes the following new features:

- New Rules Manager Allows Creation of Complex Rule Expressions at the Project Level
- Customizable Advanced Runtime UI Themes
- Ability to Install RPMs and DEBs from Install Linux Package Action
- Support for the Latest Platforms
- Enhancements

New Rules Manager Allows Creation of Complex Rule Expressions at the Project Level

InstallAnywhere now includes a rule expression manager which allows installer authors to:

- Create complex rule expressions at the project level
- Combine multiple rules and save them to a single rule expression
- Reuse a rule or a set of rules across the project
- Associate a rule expression to a file extension, including the option to automatically associate the rule expression to a file whenever a file containing that file extension is added to the project
- Remove associated rules from custom file extensions

See the following help topics in the InstallAnywhere Help library for procedures about various functionality that you can do with project level rule expressions:

- Configuring and Saving a New Rule Expression
- Associating a Rule Expression to a File Extension
- Loading a Rule Expression
- Deleting a Saved Rule Expression

Customizable Advanced Runtime UI Themes

InstallAnywhere has improved the runtime user experience of the installer by adding customizable advanced runtime UI themes. A theme is a set of runtime UI settings that are used to customize the installer panels and frames. For example, you can specify fonts, font attributes, and even import your own fonts. You can customize the colors of windows and buttons, choose to use frameless windows, display your company's own logo in the installer steps panels, and many other settings.

For complete information about each of the advanced runtime UI theme settings available and for information about how to customize advanced runtime UI themes, see the UI Panel Settings Area topic.

Ability to Install RPMs and DEBs from Install Linux Package Action

Edition ▪ *These features are available in the following editions:*

- *InstallAnywhere Premier Edition with Virtualization and Cloud*
- *InstallAnywhere Premier Edition*

In InstallAnywhere 2017, the **Install Linux RPM** install action has been updated to **Install Linux Package** and now allows you to install and uninstall Linux RPM Package Manager (.rpm) files, Debian (.deb) packages, or files from a default or a custom repository. The package files can either be bundled with the installer or can be pre-existing on the system.

If you choose to install from a repository, there are two options:

- Install from Repository—Install from the default distribution repository on the machine you are targeting
- Install from Repository / Custom Repository—Install from a specified location of a custom repository. When you click the Custom Repository check box, the Choose Repository button is enabled, allowing you to search for the custom repository location, choosing either a *.list file for Debian file distributions or *.repo for Linux.

If the RPM or DEB is relocatable, and the **Relocatable** check box is selected in the action customizer, the file is installed to its location in the file tree.

Additionally, the RPM or DEB can be set to **Ignore Dependencies** (similar to the `--nodeps` option for the command-line RPM tool) and to Force Installation (`--force`).

You can also choose **Do Not Uninstall**. For custom repositories, the only check box that is available is to **Do Not Uninstall**.

The Install Linux Package action allows you to add only one package at a time. If you want to add additional packages, click **Add Action** again.

For complete information about install actions, refer to the Install Actions topic.

Support for the Latest Platforms

InstallAnywhere now supports the following platforms for running the installer run-time environment, as well as for the InstallAnywhere authoring environment:

- Windows 10 Anniversary Update (x86 and x64)
- Windows 2016 Server (x64)
- macOS Sierra (10.12)
- Red Hat Enterprise Linux 7.2 (x64)
- OpenSUSE Linux 13.2 (x64)
- SUSE Linux Enterprise 12 (SP1, x64)
- Fedora 24 (desktop editions; x64)

InstallAnywhere now supports the following platforms for running the installer run-time environment:

- CentOS 7 (x86, x64)
- CentOS 6.8 (x86, x64)

- Red Hat Enterprise Linux 7.2 for PowerPC (little endian)
- Ubuntu 16.04 LTS (x64)

Enhancements

This section lists enhancements that were included in InstallAnywhere 2017:

- New Command-Line Build-Related Argument to Generate Encrypted Variable Value
- New Require Root Setting Available Under Project Page in the Platforms View
- Modify Text File Action Enhancement
- Support for Croatian Local Added

New Command-Line Build-Related Argument to Generate Encrypted Variable Value

When building using the command line, you can use a new build-related argument to allow you to generate an encrypted hexadecimal value that can be directly copied and pasted into a response file as an encrypted variable value.

```
build -encrypt <product-code> <text-to-encrypt>
```

The resulting encrypted hexadecimal value can then be used later. For example, you could update the parent response file with an encrypted value generated from a password without needing to first run the entire installation to only generate this value.

This enhancement resolves issue IOJ-1758930.

New Require Root Setting Available Under Project Page in the Platforms View

The following new settings are available in the Unix area of the Project Page under the Platforms view.

Table 1-3 ▪ UNIX Settings

Setting	Description
Require Root	Specify whether the installer requires root account permissions. A root account is the user name or account that by default has access to all commands and files on a Linux or other Unix-like operating system. It is also referred to as the root account, root user, or the superuser. If you select **Yes** in this setting, the **Error Message** setting under **Require Root** is enabled.
Error Message	Specify an error message to appear if the installer is run with a non-root account. This field is enabled when **Yes** is selected from the **Require Root** drop-down list.

Modify Text File Action Enhancement

The Modify Text File actions have been enhanced to allow you specify both the source file that is being opened and read as well as the destination file that is being saved after it has been modified. To account for this change, the previous File Encoding field has been replaced with the following two fields:

- Source File Encoding

- Destination File Encoding

In previous versions of InstallAnywhere, there were instances where leaving the File Encoding field blank resulted in a loss of characters when saving the destination file as a result of Java not being able to easily detect what kind of encoding the file was actually using. Therefore, with this enhancement, you can now explicitly specify the source file and destination file encoding to ensure no loss of characters are experienced in the process. Now, the encoding that is applied is dependent solely upon the destination file encoding provided irrespective of the source file encoding. The following table provides example of the file encoding that is applied in cases where the destination file coding is left blank, is invalid, or is unsupported.

Table 1-4 ▪ File Encoding

Source File Encoding	Destination File Encoding	Encoding Applied
empty	empty	UTF-8
invalid (eg. Hello)	empty	UTF-8
unsupported (cp1047)	empty	UTF-8
empty	invalid	Platform default
empty	unsupported	Platform default

This enhancements applies to the following actions:

- Modify Text File - In Archive Action
- Modify Text File - Multiple Files Action
- Modify Text File - Single File Action

Support for Croatian Local Added



Edition ▪ *The Premier edition of InstallAnywhere includes default run-time strings in 31 supported languages. TheProfessional edition includes default run-time strings in 9 languages.*

InstallAnywhere has added support the Croatian locale. You can now specify Croatian as a locale you want your installation to support. For information about specifying a locale to your installer, refer to the Generating Multilanguage Installers topic.

What's New in InstallAnywhere 2015 SP1

Expanded Platform Support

InstallAnywhere now supports the following platform for the installer run-time environment, as well as for the InstallAnywhere authoring environment:

- OS X El Capitan (10.11) with Oracle Java 7 or 8

What's New in InstallAnywhere 2015

New Features

InstallAnywhere includes the following new features.

Ability to Build Docker Images from InstallAnywhere Projects

InstallAnywhere Premier Edition with Virtualization and Cloud enables you use your existing InstallAnywhere projects to configure and build Docker images along with your traditional multiplatform installers. You can use the Docker support to prepare your on-premises Web applications for deployment to the cloud and to data centers.

Docker is a platform that enables developers and system administrators to build, distribute, and run applications on physical machines, data center virtual machines, or the cloud; it enables you to separate your applications from your infrastructure. A Docker image is a read-only template that InstallAnywhere can build from your project. A Docker container is launched from a Docker image. A Docker container consists of a complete file system that includes everything that is needed to run—code, system tools, system libraries, and more; it is the run component of the Docker platform.

To enable, configure, and build Docker images, use the new Build Containers view on the Build page in the InstallAnywhere Advanced Designer.

The Build Containers view also enables you to search public and private Docker registries for base images—such as those for Ubuntu, CentOS, Java, and MySQL—and pull them into your Docker images. You can also specify installers that you want to be run on your Docker images.

To customize or further fine-tune the creation of Docker images, advanced users have the ability to edit the Dockerfile—the set of commands that tells the Docker engine how to generate a Docker image—that InstallAnywhere creates as you configure Docker settings in your project; this capability is in the Build Containers view.

A new `-dockerContainerMode` command-line build parameter is available for building Docker containers.

To learn more, see:

- Creating Docker Containers
- Requirements for Docker Support
- About Docker
- Components of the Docker Platform
- Advantages of Using Docker Containers
- Enabling Docker Support
- Working with Docker Container Configurations
- Configuring Docker Engine Settings
- Selecting the Base Image for Your Docker Container
- Selecting the Installers that You Want to Include in Your Docker Container
- Editing the Dockerfile Directly

- Building a Docker Container
- Troubleshooting “Out of Memory” Error Upon InstallAnywhere Launch or Command Line Build
- Build Command-Line Arguments

Support for Deploying Web Applications to Local or Remote IBM WebSphere Servers

InstallAnywhere Premier Edition offers improvements for deploying Web applications (.ear or .war) to IBM WebSphere servers.

Enabling End Users to Specify Connection Information for WebSphere Servers

Now you can add to your projects run-time panels or consoles that enable end users to specify settings for WebSphere servers before deploying Web applications to these servers. For example, you can let end users specify information such as the local or remote WebSphere server host where they want to deploy the Web application.

To add the new run-time panels to a project, add the WebSphere Runtime Deployment panel or console actions. These actions are available for the Pre-Install view on the Sequence page in the InstallAnywhere Advanced Designer.

By default, these new run-time panels and consoles use variables such as $WEBSPHERE_DEPLOYMENT_OPTION$ and $WEBSPHERE_HOSTNAME$ for storing the values that end users enter.

To learn more, see:

- Specifying Which Deployment Options to Support for IBM WebSphere Servers
- Deploying Web Applications (WAR and EAR Archives) to Servers
- Enabling End Users to Specify IBM WebSphere Server Information
- Deploy WAR/EAR Archive Action
- WebSphere Runtime Deployment Panel Action
- WebSphere Runtime Deployment Console Action

IBM WebSphere 7, 8, and 8.5 Support

InstallAnywhere now has support for deploying Web applications to WebSphere 7, 8, and 8.5 servers.

Expanded Support for Connecting to IBM DB2, Microsoft SQL Server, and PostgreSQL Databases and Running SQL Scripts

InstallAnywhere Premier Edition offers several improvements for connecting to IBM DB2, Microsoft SQL Server, and PostgreSQL databases and running SQL scripts.

Enabling End Users to Specify Connection Information for Local and Remote Servers

Now you can add to your projects run-time panels or consoles that enable end users to specify connection information such as the name of the local or remote server, as well as the credentials that should be used to connect to the server through server authentication. In addition, you can optionally enable end users to test the connection information that they entered.

To add the new run-time panels and consoles to a project, add the Choose Database Connection panel or console actions. These actions are available for the Pre-Install view on the Sequence page in the InstallAnywhere Advanced Designer.

By default, these new run-time panels and consoles use variables such as *$DB_NAME_VARIABLE$* and *$DB_SERVERHOST_VARIABLE$* for storing the values that end users enter.

To learn more, see:

- Enabling End Users to Specify Database Connection Information
- Choose Database Connection Panel Action
- Choose Database Connection Console Action

Support for Creating Databases on Target Systems

The Run SQL Script action now optionally supports the creation of Microsoft SQL Server and PostgreSQL databases on target systems. The Properties Customizer that is displayed when you select this action in the Install view on the Sequence page has a Create Database check box that you can use to indicate whether to create the database at run time.

For more information, see the following:

- Running a SQL Script
- Run SQL Script Action
- Hosts View
- Database Server Host Settings

IBM DB2 10.1 and 10.5 Support

InstallAnywhere now has support for managing IBM DB2 10.1 and 10.5 databases, as well as IBM DB2 9.0 and 9.7 databases.

Microsoft SQL Server 2008 R2, 2012, and 2014 Support

InstallAnywhere now has support for managing Microsoft SQL Server 2008 R2, 2012, and 2014 databases.

Support for SHA-2 Digital Certificates

InstallAnywhere now enables you to use digital certificates that use the SHA-2 hashing algorithm for signing your Windows-based installers (the installer .exe file, as well as the installer launcher and the uninstaller launcher) at build time.

SHA-256 is favored over SHA-1, which is being deprecated because of the potential for security vulnerabilities. Microsoft announced that Windows will stop trusting items that were signed and timestamped with SHA-1 certificates after January 1, 2016. In addition, certification authorities-the organizations that issue certificates-are phasing out the creation of SHA-1 certificates. Thus, it is recommended that you replace any SHA-1 certificates in your InstallAnywhere projects with SHA-2 certificates. For the latest information and more specific details, check with your certification authority.

If your project is configured to sign with a SHA-2 certificate, InstallAnywhere uses a SHA-2 hash in the signature of the files that it signs at build time. If your project is configured to sign with a SHA-1 certificate, InstallAnywhere generates a build error: either 813 (a SHA-1 certificate is configured in the project) or 814 (a SHA-1 certificate is configured in the project and a timestamp server is not being used for signing).

To learn more, see Digitally Signing Windows-Based Installers.

Support for the Latest Platforms

InstallAnywhere now supports the following platforms for running installer run-time environment, as well as for the InstallAnywhere the authoring environment:

- Windows 10 (x86 and x64)
- Red Hat Enterprise Linux 7.1 (desktop edition; x64)
- Red Hat Enterprise Linux 7 (zSeries)
- OpenSUSE Linux 13.2 (desktop edition; x64)
- SUSE Linux Enterprise 12 (server edition; x64)
- Ubuntu 15.04 (desktop and server editions; x64)
- Ubuntu 14.10 (desktop and server editions; x64)
- Fedora 20 (desktop edition; x64)

Enhancements

Support for Transparent Backgrounds on UI Panels

InstallAnywhere now enables you to make the main body area of each panel in the user interface of your installer transparent. Previously, this area behind the main UI text was always white.

To specify that you want the main body area to be transparent, select the new No Color option in the Inner Panel Background Color setting. To access this setting, on the Installer UI page, click the Look & Feel Settings view. Then, in the General UI Settings area, under the Installer Frame setting, the Inner Panel Background Color setting is available with the new option.

For more information, see Look & Feel Settings View.

Ability to Change the Text Color of the Installer Steps That the Installer UI Shows for Progress

InstallAnywhere now enables you to customize the font color that is used to display the text of the installer steps that is included on the left side of the installer user interface.

To specify the color, use the Look & Feel Settings view on the Installer UI page. In the List of Labels for Installer Steps setting, click the **Installer Step Label settings** button; the Installer Steps dialog box opens. On this dialog box, click the Icons and Fonts button. The Choose Label Settings dialog box opens. To change the font color of the text for the installer steps, click the Font Color button.

To learn more, see the following:

- Choose Label Settings Dialog Box
- Look & Feel Settings View

Enhanced Show Message Dialog Action

The Show Message Dialog action has been enhanced. One of the new settings that you can configure for this Pre-Install action is a check box called Cancel and Exit on ESC or X. If you want the installer to cancel and exit the installation if the end user presses the Escape button or clicks the Close button (X) on the message dialog, select this check box.

This check box is cleared by default.

For more information, see Show Message Dialog Action.

What's New in InstallAnywhere 2014 SP1

InstallAnywhere offers expanded platform support. It also resolves a number of issues.

Expanded Platform Support

InstallAnywhere now supports the following platforms for the installer run-time environment, as well as for the InstallAnywhere authoring environment:

- Red Hat Enterprise Linux 7
- OpenSUSE Linux 13.1 (x86 and x64)
- SUSE Linux Enterprise 11 SP3 (x64)
- OS X Yosemite (10.10)

Changes to Authentication and Code-Signing Support for OS X–Based Installers

Improvements have been made to the authentication and code-signing support for OS X–based installers. End users can now successfully run code-signed installers (as well as uninstallers) with authentication support on systems that have OS X 10.8 or later.

Flexible Methods for Code Signing

InstallAnywhere now supports two different methods of code signing:

InstallAnywhere performs the code-signing step at build time on the InstallAnywhere build machine—The InstallAnywhere user has access to the Developer ID Application certificate and, in the InstallAnywhere project, configures the Code Signing settings (the certificate location and the keystore password). At build time, InstallAnywhere code signs the appropriate output files.

The InstallAnywhere build output is code signed on a designated code-signing machine—The InstallAnywhere user builds an unsigned installer on the build machine. The build output is then code signed on a secure code-signing machine that has the Developer ID Application certificate. Note: A limitation of this method is that it is not possible to sign the uninstaller; therefore, the resulting uninstaller cannot be used to uninstall the product. The only way to remove a product whose installer was code signed using this method is to drag it to the Trash.

New Tool for Code Signing the Helper Tool

When you configure your project to include authentication support, InstallAnywhere adds a helper tool to your installer. The helper tool is the file that is used to launch the installer or uninstaller with elevated privileges. The helper tool must be code signed with the same Developer ID Application certificate that you will be using to sign your installer.

A new Prepare the Helper Tool utility is available when InstallAnywhere is installed on an OS X–based system. You can use this utility on your build machine to sign the helper tool, or you can copy the utility to your designated code-signing machine, where you can then sign the helper tool.

Once you have signed the helper tool, you can copy the signed helper tool to all of your OS X–based build machines.

Support for Version 2 Signatures

InstallAnywhere now has support for creating version 2 signatures for OS X–based systems. All code signing must be done on systems that are running OS X 10.9 or later. Version 1 signatures, which are created by earlier versions of OS X, are not recognized by Gatekeeper on systems with OS X 10.9 and later and are considered obsolete. Files that are signed with version 2 signatures will work on OS X 10.8 and later.

Requirements for Code Signing

The following requirements must be met for code signing OS X–based installers:

- The OS X–based installer must be built on an OS X–based system.
- All code signing must be done on systems that are running OS X 10.9 or later.
- A Developer ID Application certificate must be used to sign the files. The certificate should be added to the login keychain—not the system keychain—on the machine that is going to be used for code signing, and the same user account that was used to add the certificate to the login keychain should be used to sign files.
- If you plan on performing builds through the command-line console, ensure that the certificate has been granted access to be used by all applications.
- Ensure that the latest Xcode IDE and all of its default SDKs are installed on the machine that is going to be used for code signing.

For additional details about the authentication and code-signing support for OS X–based systems, see the following:

- About Authentication and Code-Signing Support for OS or OS X–Based Installers
- Code-Signing Methods for OS or OS X–Based Installers
- Requirements for Code-Signing Support for OS or OS X–Based Installers
- Adding the Code-Signing Capability to Your InstallAnywhere Build Machines or Code-Signing Machines
- Code Signing Your OS or OS X–Based Installers and Including Authentication Support
- Verifying that Your Code-Signing Output Files Are Working as Expected on OS or OS X–Based Target Systems
- Troubleshooting Tips for Code-Signing and Authentication Support for OS or OS X–Based Target Systems
- Prepare the Helper Tool Dialog Box

What's New in InstallAnywhere 2014

New Features

InstallAnywhere includes the following new features.

Ability to Create Upgrades that Uninstall the Earlier Version If Present Before Installing the New Version

InstallAnywhere includes support for creating upgrades. At run time, if an earlier version of the product is present on target systems, it is uninstalled, and then the new version is installed. If an earlier version is not present, the installer installs the new version.

To configure settings for your upgrade, use the new Upgrades view, which is available from the Project page.

Note that a requirement for upgrade support is that the base version of the product must have been installed with silent mode enabled. That is, the uninstaller must be capable of silently uninstalling the earlier version.

This feature is available in the Premier edition of InstallAnywhere.

To learn more, see:

- Updating Applications
- Requirements for Upgrade Support
- Creating an Upgrade
- Enabling Upgrade Support
- Configuring Upgrade Settings
- Detecting Installed Earlier Versions that Need to Be Updated
- Adding Upgrade-Specific Actions to Sequences
- Special Considerations for Upgrade Support
- Upgrades View
- Manage Upgrade Configurations Dialog Box

Updated Authentication Support for Apple OS X

If you want your OS X–based installers and uninstallers to install files to and remove files from locations where write permissions are restricted for standard users, you can configure your project to require authentication. This support has been updated for OS X 10.8 and later systems. When authentication is required and standard users who are not root users or administrative users with adequate privileges try to launch your installer or uninstaller, they are prompted to enter an administrator name and password in order to proceed.

To indicate whether you want to require authentication for your OS X–based installers and uninstallers, use the settings in the Authentication category in the Mac OS X area (Project page > Platforms view).

This feature is available in the Premier edition of InstallAnywhere.

To learn more, see:

- Requiring Elevated Privileges for OS or OS X–Based Installers and UninstallersIf you want your OS or OS X–based installers and uninstallers to install files to and remove files from locations where write permissions are restricted for standard users, you can configure your project to require authentication. When authentication is required and standard users who are not root users or administrative users with adequate privileges try to launch your installer or uninstaller, they are prompted to enter an administrator name and password in order to proceed.
- Platforms View

Expanded Support for Connecting to MySQL Servers and Running SQL Scripts

InstallAnywhere Premier Edition offers several improvements for connecting to MySQL servers and running SQL scripts.

Enabling End Users to Specify Connection Information for Local and Remote Servers

Now you can add to your projects run-time panels or consoles that enable end users to specify connection information such as the name of the local or remote server, as well as the credentials that should be used to connect to the server through server authentication. In addition, you can optionally enable end users to test the connection information that they entered.

To add the new run-time panels and consoles to a project, add the Choose Database Connection panel or console actions. These actions are available for the Pre-Install view on the Sequence page in the InstallAnywhere Advanced Designer.

By default, these new run-time panels and consoles use variables such as *$DB_NAME_VARIABLE$* and *$DB_SERVERHOST_VARIABLE$* for storing the values that end users enter.

To learn more, see:

- Enabling End Users to Specify Database Connection Information
- Choose Database Connection Panel Action
- Choose Database Connection Console Action

Support for Creating Databases on Target Systems

The Run SQL Script action now optionally supports the creation of databases on target systems. The Properties Customizer that is displayed when you select this action in the Install view on the Sequence page has a new Create MySQL Database check box that you can use to indicate whether to create the database at run time.

Note that some of the connection-related settings (such as Server Host and Server Port) that were previously displayed when the database host was selected in the Hosts view on the Organization page have been moved to a new tab at the bottom of the view for the Run SQL Script action.

For more information, see the following:

- Running a SQL Script
- Run SQL Script Action
- Hosts View
- Database Server Host Settings

MySQL 5.5 and 5.6 Support

InstallAnywhere now has support for managing MySQL 5.5 and 5.6 databases.

Expanded Documentation on Managing SQL Servers

The documentation on managing SQL servers has been expanded. It now explains how to include the appropriate drivers for connecting to SQL databases and offers additional how-to information. For details, see:

- Managing Database Servers
- Adding a SQL Database Host to Your Project

- Including JDBC Drivers for Connecting to SQL Databases
- Obtaining DB2 Drivers
- Obtaining a MySQL Driver
- Obtaining an Oracle Driver
- Obtaining a Generic JDBC Driver

Support for Deploying Web Applications to Local or Remote Apache Tomcat Servers

InstallAnywhere Premier Edition offers improvements for deploying Web applications to Tomcat servers.

Enabling End Users to Specify Connection Information for Tomcat Servers

Now you can add to your projects run-time panels or consoles that enable end users to specify settings for Apache Tomcat servers before deploying Web applications to these servers. For example, you can enable end users to choose between deploying applications to local or remote Tomcat servers. You can also let end users specify information such as the path on the local Tomcat server where they want to deploy the Web application or the name of the remote Tomcat server.

To add the new run-time panels to a project, add the Tomcat Runtime Deployment panel or console actions. These actions are available for the Pre-Install view on the Sequence page in the InstallAnywhere Advanced Designer.

By default, these new run-time panels and consoles use variables such as *$TOMCAT_DEPLOYMENT_OPTION$* and *$TOMCAT_SERVER_PATH$* for storing the values that end users enter.

To learn more, see:

- Specifying Which Deployment Options to Support for Apache Tomcat Servers
- Deploying Web Applications (WAR and EAR Archives) to Servers
- Enabling End Users to Specify Apache Tomcat Server Information
- Deploy WAR/EAR Archive Action
- Tomcat Runtime Deployment Panel Action
- Tomcat Runtime Deployment Console Action

Apache Tomcat 7 and 8 Support

InstallAnywhere now has support for deploying Web applications to Tomcat 7.0.x and 8.0 servers.

Red Hat Enterprise Linux 6.5 and Ubuntu 14.04 Support

InstallAnywhere now supports the following platforms for the installer run-time environment, as well as for the InstallAnywhere authoring environment:

- Red Hat Enterprise Linux 6.5 (desktop and server editions; x64)
- Ubuntu 14.04 (x64)

Enhancements

New Variable for the Browse for CD Button on Some Apple OS X–Based Systems

The Change Media run-time panel has a Browse button that should enable end users to browse to the next installation media that they want to use. Apple OS X–based systems that use Oracle Java 7 with an update earlier than update 45 do not support built-in resources for the button on that panel, so the browse button does not work as expected. If you want to use Swing resources instead of built-in resources for that panel and avoid problems with the browse button, you can set the new ia.mac.filechooser.substituteSwingInsteadOfNativeForOlderJRE7 variable to true for installers that target OS X–based systems that use Oracle Java 7 with an update earlier than update 45. The default value for this property is false.

To set ia.mac.filechooser.substituteSwingInsteadOfNativeForOlderJRE7 to true, specify the following command-line parameter in your project (Project page > JVM Settings view > Installer Settings tab > Optional Installer Arguments):

```
-Dia.mac.filechooser.substituteSwingInsteadOfNativeForOlderJRE7=true
```

To learn more, see $IA_BROWSE_FOLDERS$ Variable.

Enhanced Support for Registering Windows Services

The Register Windows Service action has a new Service Description setting that lets you specify the description of the Windows service that your installer is registering on Windows-based target systems. The description is displayed is displayed in the service control manager's Description column. It is also displayed in the Description box on the General tab of the service's Properties dialog box.

What's New in InstallAnywhere 2013

New Features

InstallAnywhere includes the following new features.

Redesigned User Interface for the Advanced Designer

The Advanced Designer in InstallAnywhere has been overhauled to enhance usability, increase productivity, and update the look and feel for the design-time environment.

Following are highlights for the redesign.

Revamped, Streamlined Navigation

The Advanced Designer has two main navigational elements:

- The primary navigational element is the horizontal row of buttons near the top of the InstallAnywhere interface, directly below the menu bar. You can click these buttons to open various pages in the Advanced Designer. This navigational element has buttons for Project, Installer UI, Organization, Sequence, and Build.
- The secondary navigational element is the vertical column of buttons on the left side of the InstallAnywhere interface; these buttons are links to the various views in the Advanced Designer. The list of buttons in this area changes, depending on which page in the Advanced Designer is currently open. For example, when the Organization page is open, the vertical navigation area on the left contains buttons that point to views such

as Install Sets, Features, and Components. When the Build page is open, the buttons on the left point to the Build Installers and Build Appliances views.

Previously, both levels of the navigation were positioned along the left side of the Advanced Designer, in two separate vertical columns. The primary navigational element was the first vertical column in the Advanced Designer, and this level of navigation represented a UI element that was previously called *tasks*. The secondary navigational element was the second column of links, which were referred to as *subtasks*.

Also as part of the navigation changes, the primary navigation element has been streamlined to make it easier to navigate the Advanced Designer. The Pre-Install, Install, Post-Install, Pre-Uninstall, Uninstall, and Post-Uninstall areas (which enable you to define and schedule files to be installed, actions to be launched, and panels to be displayed) are now available as individual views on the new Sequence page. Previously, these areas were part of the primary level of navigation.

The Help button is now a blue button with a question mark on it; it has been moved from the lower-left corner of the Advanced Designer to the top-right corner. To launch the InstallAnywhere Help Library in your machine's default Web browser, click this button.

To learn more, see:

- Advanced Designer Reference
- Project Page
- Installer UI Page
- Organization Page
- Sequence Page
- Build Page

New Default Start Mode: Advanced Designer Instead of Project Wizard

Although you can use the Project Wizard in InstallAnywhere to quickly create a basic installer, the Advanced Designer—with its support for advanced installer configuration—is the area in InstallAnywhere where most installation developers spend the majority of their time. Therefore, when you start InstallAnywhere, it now launches the Advanced Designer by default. Previously, it opened the Project Wizard by default.

The Advanced Designer has a setting that lets you specify which mode you want to use when starting InstallAnywhere. You can change the preferred startup mode at any time: On the Edit menu, click Preferences. The InstallAnywhere Preferences dialog box opens. On the General tab of this dialog box, use the InstallAnywhere Startup setting to specify whether you want InstallAnywhere to start in Project Wizard mode or Advance Designer mode.

To switch from the Advanced Designer to the Project Wizard: On the File menu, click Project Wizard. This functionality was previously available from a Start Wizard command on the Wizard menu in the Advanced Designer.

To switch from the Project Wizard to the Advanced Designer, click the Advanced Designer button on any panel of the Project Wizard.

New Grid-Based Format for Settings in Some Views of the Advanced Designer

Some of the views on the Project page and on the Installer UI page in the Advanced Designer have been consolidated and reorganized into two-column grids to improve productivity and usability. The new grid format shows the names of the settings in the left column and their corresponding values in the right column. The grid in

the new General Settings view on the Project page contains the settings that were previously displayed on the Info, Description, File Settings, Log Settings, and Software Tag subtasks. The Platforms Settings view on the Project page contains the settings that were previously displayed in different areas of the Platforms subtask. In addition, the Look & Feel view on the Installer UI page contains the setting that were previously displayed in different areas of the Look & Feel subtask.

The settings in the new grids are organized into different categories. You can expand or collapse the category rows to show or hide each part of the grid that contains the settings within a particular category.

For more information, see:

- General Settings View
- Platforms View
- Look & Feel Settings View

New Inline Help Panes in Some Views of the Advanced Designer

The new grid-based interface that is displayed for some of the views on the Project page and the Installer UI page includes an inline help pane. The content in this help pane changes, depending on what setting in the grid is selected. For example, if you select the Repair Mode Log setting in the General Settings view, the help pane at the bottom of the view shows help text that pertains to the Repair Mode Log setting. If you select a different setting in this view, the help pane displays help text about that particular setting.

The inline help is available in the General Settings view and the Platform Settings view on the Project page of the Advanced Designer. It is also available in the Look & Feel view on the Installer UI page of the Advanced Designer.

Usability Improvements for Sequencing Actions, Panels, and Other Items

InstallAnywhere now makes it easier to sequence and configure actions, panels, and other items in the Pre-Install, Install, Post-Install, and other views on the Sequence page in the Advanced Designer. If you are working with a long list of actions in the visual tree area that is displayed at the top of one of these views, you can temporarily hide the settings in the bottom of the view; this enables you to see more of the visual tree at once on the screen, without requiring you to scroll vertically. When you are done scheduling the actions and other items, you can have InstallAnywhere revert back to displaying the settings in the bottom of the view; this enables you to configure the settings for whichever action or other item is selected in the visual tree.

To hide the settings, click the new Collapse button; this button is on the right side of the screen, under the visual tree. When you click this button, InstallAnywhere hides the Properties Customizer box at the bottom of the screen, and changes the Collapse button into an Expand button. To make it possible to see and configure the action settings, click this Expand button.

For more details, see Sequence Page.

Usability Improvements for Managing Build-Related Settings for Installers and Virtual Appliances

The Build Installers view in the Advanced Designer includes some enhancements that make it easier to manage build configurations for installers. The Build Appliances view includes some similar enhancements for managing appliance configurations for virtual appliances.

The layout for the build configuration controls at the top of the Build Configurations tab in the Build Installers view has been revised to leave more vertical space for other areas of this page. As a result, it is now easier to use this Configurations tab to configure the settings for build targets; less vertical scrolling is required. To make this possible, the view has four new small buttons: Add Build Configuration, Copy Build Configuration, Rename Build Configuration, and Remove Build Configuration. These new buttons have icons, and they are positioned in the

same row as the Select Build Configuration list and the **Add to project build** check box. The new buttons replace the row of old buttons that was previously located below the Select Build Configuration list and the **Add to project build** check box.

Similarly, the appliance configuration controls at the top of the Appliance Configuration tab in the Build Appliances view has four new small buttons: Add Appliance Configuration, Copy Appliance Configuration, Rename Appliance Configuration, and Remove Appliance Configuration. These new buttons replace the row of old buttons.

The Build Installers view and the Build Appliances view are available on the Build page in the Advanced Designer.

To learn more, see:

- Build Page
- Build Installers View
- Build Appliances View

Support for the Latest Platforms

InstallAnywhere supports the following platforms for running InstallAnywhere (the authoring environment):

- Windows 8.1
- Windows Server 2012 R2
- Red Hat Enterprise Linux 6.4 (desktop and server editions; x86 and x64)
- SUSE Linux 10 (x86)
- OpenSUSE Linux 11.2, 11.3, 11.4, 12.1, 12.2, and 12.3 (x86 and x64)
- SUSE Linux Enterprise 11 SP2 (x64)
- Ubuntu 13.04 (desktop and server editions; x86 and x64)
- Fedora 18 and 19 (desktop and server editions; x86 and x64)

Ability to Create Virtual Appliances for Multi-tier Applications

InstallAnywhere includes support for creating multi-tier virtual appliances for VMware vCenter servers. You can create this sort of virtual appliance to enable your enterprise customers to quickly get started using your multi-tier application in their own virtualization environments, without requiring them to spend a lot of time working on complex configuration tasks.

The VM Configuration tab in the Build Appliances view on the Build page is where you define and configure one or more tiers for your virtual appliance. Each InstallAnywhere project contains a single VM tier by default. To perform other tasks such as add an additional tier to your project, use the new buttons (Add VM Tier, Copy VM Tier, Rename VM Tier, and Delete VM Tier) near the top of the VM Configuration tab. To indicate which tier in your project you want to configure, select the appropriate tier in the new Select VM Tier list on this tab.

Previously, InstallAnywhere included support for only single-tier virtual appliances.

The project automation APIs have been expanded to include support for multi-tier virtual appliances. A VirtualMachineTier class and a VirtualMachineCollectionEntity interface are now available. Each virtual machine is now added to a virtual appliance through virtual machine tiers. Previously, each virtual machine was added directly to the virtual appliance.

A new project automation API sample (MultiTierVirtualAppliance_Test.java) is installed with InstallAnywhere (*InstallAnywhere Installation Directory*\project-auto\sample\src\com\ia\projauto\samples). This Java file demonstrates how to create a multi-tier virtual appliance through project automation APIs.

`IABankingApplication.iap_xml` is a sample InstallAnywhere project that you can use to explore InstallAnywhere's support for multi-tier virtual appliances. This project file, along with application files, are installed with InstallAnywhere (*InstallAnywhere Installation Directory*\IABankingApplication). The sample project is configured to create a three-tier virtual appliance.

This feature is available in InstallAnywhere Premier Edition with Virtualization and Cloud.

For more information, see:

- Managing Multiple Tiers for a Virtual Appliance
- Manage VM Tiers Tab
- VM Configuration Tab
- Project Automation API Code Samples
- Using the IABankingApplication Sample for Creating Multi-Tier Virtual Appliances

Expanded Operating System Support for Creating VMware vSphere/vCenter Virtual Appliances

InstallAnywhere now enables you to create VMware vSphere/vCenter virtual appliances for an expanded list of operating systems:

- CentOS 6.2 and 6.3 (32-bit and 64-bit)
- Red Hat Enterprise Linux 6.4 (32-bit and 64-bit)
- Red Hat Enterprise Linux 6.3 (64-bit)
- Ubuntu 13.04 (32-bit and 64-bit)
- Ubuntu 12.10 (32-bit and 64-bit)
- Ubuntu 12.04 (32-bit and 64-bit)
- Ubuntu 11.10 (32-bit and 64-bit)

Previously, only 32-bit versions of CentOS 6.2, CentOS 6.3, Ubuntu 11.10, and Ubuntu 12.04 were supported.

This feature is available in InstallAnywhere Premier Edition with Virtualization and Cloud.

For a complete list of supported hypervisors and operating systems for virtual appliances, see Supported Hypervisors and Platforms for Virtual Appliances.

Ability to Create Virtual Appliances from Existing Virtual Machines

InstallAnywhere now has support for creating a virtual appliance from an existing VMware vSphere 5 server, without requiring you to use a predefined or custom virtual machine (VM) template (a preinstalled virtual machine disk that contains the operating system and any required operating system packages). This capability makes it easier to quickly get started building virtual appliances for VMware vSphere servers.

To specify an existing VM that you want to use for your vSphere appliance, specify the connection information in the Connection Settings area of the Appliance Configuration tab, which is available in the Build Appliances view on the Build page. Once you have configured connection information, switch to the VM Configuration tab, and click the new Choose a VM button. The Choose a VM button is next to the new Chosen VM list.

When you click the new Choose a VM button, the Choose a VM Wizard opens, enabling you to specify whether you want to create your virtual appliance based on a specific VM or a specific VM snapshot that is running on an VMware vSphere/vCenter 5 server. Optionally, you can also use this wizard to select a specific VM template that you want to use.

Previously, in order to create a virtual appliance for VMware vSphere 5 server, it was necessary to first specify the VM template that you wanted to use for your virtual appliance. You could use one of the ones that is available for InstallAnywhere, or you could create your own custom VM template in InstallAnywhere.

This feature is available in InstallAnywhere Premier Edition with Virtualization and Cloud.

To learn more, see:

- Selecting a VM for a VMware Virtual Appliance
- Choose a VM Wizard

Ability to Build Pure 64-Bit Installers for 64-Bit Windows-Based Systems

Windows Server Core supports disabling 32-bit Windows-on-Windows (WOW64) support. As this configuration becomes more popular, you may want to ensure that your 64-bit applications can install without any reliance on 32-bit functionality. To make this possible, InstallAnywhere now enables you to build pure 64-bit installers for Windows-based target systems; these installers can be run on 64-bit Windows-based systems that do not have WOW64 functionality. They contain 64-bit self-extractors, LaunchAnywheres, and VM packs, as needed. They can install to 64-bit file and registry locations and install 64-bit Windows services as required.

To create a 64-bit Windows-based installer, use the new Windows_Pure_64_Bit build target that is available on the Build Targets tab in the Build Installers view. You can use this build target to configure options such as whether you want to bundle a VM pack in your installer. If you specify that you want to include a VM pack, only 64-bit VM bundles are available for selection.

This feature is available in InstallAnywhere Premier Edition.

To learn more, see:

- Targeting 32-Bit and 64-Bit Windows-Based Systems
- Creating Pure 64-Bit Installers for 64-Bit Windows-Based Target Systems

New build.exe Command-Line Parameters

If you use the command-line tool `build.exe` to build installers, you can use the following platform-related arguments for adding or removing pure 64-bit Windows-based targets:

- w64 or W64—Pure 64-bit Windows-based installer without VM
- w64v or W64V—Pure 64-bit Windows-based installer with VM
- w64c or W64C—Pure 64-bit Windows-based installer without VM and with console launcher option
- w64g or W64G—Pure 64-bit Windows-based installer without VM and with graphical launcher option
- w64vc or W64VC—Pure 64-bit Windows-based installer with VM and with console launcher option
- w64vg or W64VG—Pure 64-bit Windows-based installer with VM and with graphical launcher option

To add one of the aforementioned build targets, precede the argument with a plus sign (+); to remove one of the aforementioned build targets, precede the argument with a minus sign (-). For example, to build a pure 64-bit Windows-based installer with VM and with the graphical launcher option, pass either of the following to `build.exe`:

```
+w64vg
+W64VG
```

For a list of all of the supported `build.exe` command-line parameters, see Build Command-Line Arguments.

New Platform Attributes for the Build Properties XML File

If you use a build properties XML file to pass build properties to `build.exe`, you can use the following new platform attributes to define a pure 64-bit Windows-based build target:

- BuildWindows64WithVM="<true/false>"
- BuildWindows64WithoutVM="<true/false>"
- Windows64VMpackLocation="<path_to_file>"
- Windows64DefaultUI="<Silent/GUI/Console>"

For a list of all of the supported properties for the build properties XML file, see Supported Properties in the BuildProperties.xml File.

New Properties for the .properties File

If you use a .properties file to pass build properties to `build.exe`, you can set a new property to define the default UI mode for pure 64-bit Windows-based build targets. The new property is default.ui.mode.windows64 property; you can set it equal to Silent, Console, or GUI. For example:

```
default.ui.mode.windows64=GUI
```

The following True/False properties are also new:

- config.1.com.zerog.ia.build.platform.windows64.novm
- config.1.com.zerog.ia.build.platform.windows64.vm
- config.1.com.zerog.ia.build.vmpack.windows64.path

For a list of all of the supported properties for the .properties file, see Supported Properties in the buildproperties.properties File.

New Attribute Values for the InstallAnywhere Ant Task

If you use Ant to build installers, you can use the following new attributes in your InstallAnywhere Ant task for defining pure 64-bit Windows-based build targets.

- BuildWindows64WithVM—Use this attribute with True or False values to indicate whether you are building a pure 64-bit Windows-based installer with VM.
- BuildWindows64WithoutVM—Use this attribute with True or False values to indicate whether you are building a pure 64-bit Windows-based installer without VM.
- Windows64DefaultUI—Use one of the following values to specify the default UI mode: Silent, Console, or GUI.
- Windows64VMPackLocation—Use this attribute to specify the path of the VM pack for pure 64-bit Windows-based build targets.

For a list of supported parameters for the InstallAnywhere Ant task, see InstallAnywhere Ant Task Reference.

Support for Digitally Signing Installers for Windows-Based Build Targets

InstallAnywhere includes support for digitally signing your Windows-based installers (the installer .exe file, as well as the installer launcher and the uninstaller launcher) at build time. If you want to digitally sign these installers, use the Platforms Settings view on the Project page to configure digital signature information. In this view, the Windows category contains some digital signing settings. This is where you specify the certificate file (.pfx) that you want to use to sign your Windows-based installers. It is also where you specify the password for the certificate and the timestamp server that you want to use.

As an alternative to specifying the actual certificate file, password, and timestamp server in the Platforms Settings view, you can use build-time variables in these settings (that is, enclose the name of each variable within at symbols: `@VariableName@`). You can set build-time variables in the Variables view on the Project page, through a .properties file, or through environment variables.

Note that the ability to digital sign Windows-based installers at build time requires that you are using InstallAnywhere on a Windows-based system. If you try to build a Windows-based installer on a non-Windows system, a build warning occurs, and InstallAnywhere does not sign the resulting installer.

To learn more, see:

- Digitally Signing Windows-Based Installers
- Platforms View

Support for JRE 7 on OS X 10.7.3 and Later

InstallAnywhere now has support for building OS X–based installers that use JRE 7 at run time. InstallAnywhere also has support for creating VM packs with Oracle JRE 7, as well as for bundling installers with Oracle JRE 7.

When you configure any OS X build target in the Build Installers view on the Build page, you now have several VM-related options. You can specify whether the installer should search OS X systems for the presence of Oracle JRE 7 or later, or for the presence of Apple JRE 6 or earlier; if the required JRE is not available, the installation fails and exits. If multiple required versions of JRE are present, the installer uses the one that is set by the JRE_HOME environment variable.

An OS X build target also lets you specify whether you want the installer to avoid searching OS X target systems for a required JRE and instead use a specific bundled VM; if you choose to use a bundled VM instead of searching for a required VM, specify which VM you want to bundle with the installer.

For more information, see:

- Defining Build Targets
- Build Targets Subtab
- Creating JRE VM Packs
- JRE VM Pack Structure
- JRE VM Pack Properties
- Run-Time Behavior for LaunchAnywhere Files on OS and OS X–Based Systems

Support for Configuring the Windows Execution Level Now Available in All InstallAnywhere Editions

Support for configuring the Windows execution level is now available in the Professional (formerly known as Standard) edition of InstallAnywhere. This capability lets you select the execution level that your installer launcher requires for target systems that are running Windows Vista and later or Windows Server 2008 or later. To configure the execution level of your Windows-based installers, select the appropriate option in the Windows Execution Level setting in the Advanced Designer (Project page > Platforms view > Windows area).

Previously, this support was available only in the Premier (formerly known as Enterprise) edition of InstallAnywhere.

To learn more, see Platforms View.

Support for Setting the field size for Add or Remove Programs Applet on Windows in all InstallAnywhere Editions

Support for setting the field size is now available in the Professional (formerly known as Standard) edition of InstallAnywhere. This new field **Estimated size (in MB)** is added under **Projects > Platforms > Windows** section for all project types.

By default 'Estimated size(in MB)' field value will be blank. You can enter an estimated size of any value > 0 (Zero) and <= 4194303(4TB).



Note ▪ *Cannot enter floating/decimal value for an estimated size.*

Support for Managing Windows-Based Services Now Available in All InstallAnywhere Editions

Support for managing Windows-based services is now available in the Professional (formerly known as Standard) edition of InstallAnywhere. To manage a service in your installer, use the Sequence page in the Advanced Designer to add any of the service-related actions to your project.

Previously, this support was available only in the Premier (formerly known as Enterprise) edition of InstallAnywhere.

To learn more, see General Actions.

Support for Setting System Environment Variables Now Available in All InstallAnywhere Editions

Support for setting system environment variables on Windows-based and UNIX-based target systems is now available in the Professional (formerly known as Standard) edition of InstallAnywhere. To set a system environment variable, add the Set System Environment Variable action to a sequence through the Sequence page in the Advanced Designer.

Previously, this support was available only in the Premier (formerly known as Enterprise) edition of InstallAnywhere.

To learn more, see Install Actions.

Enhancements

Support for Displaying the Percentage Complete on the Install Progress Panel

The progress bar on the default Install Progress panel now shows the numeric percentage of the installation that is complete. In addition, a new method is available for obtaining the percentage of the installation that is complete:

```
public int getCurrentProgressPercentage();
```

Expanded Search Support for the Find Component in InstallAnywhere Registry Action on 64-Bit Target Systems

A new **Search in both 32-bit and 64-bit InstallAnywhere registries** check box is available when you add the Find Component in InstallAnywhere Registry action to a sequence in your project. If you want the action to search for a component in both 32-bit and 64-bit InstallAnywhere registry locations (for example, Program Files (x86)\Zero G Registry and Program Files\Zero G Registry) on 64-bit target systems, select this check box.

To learn more, see Find Component in InstallAnywhere Registry Action.

Using Help

Revenera understands the importance of having useful information and help resources at your fingertips. In addition to the inline help that is embedded within various areas of the InstallAnywhere interface, InstallAnywhere includes the Help Library (the online help library that is installed with InstallAnywhere) and HelpNet.

InstallAnywhere Help Library

When you have questions about InstallAnywhere, first consult the InstallAnywhere Help Library, which is the complete user's guide for using InstallAnywhere. You can access the InstallAnywhere Help Library from the Help menu in InstallAnywhere, or by clicking the Help button in the Advanced Designer.

No Internet connectivity is required to view the InstallAnywhere Help Library. Essentially, the online help viewer is a tool that you can use to display, search, and filter technical information based on your personal needs.

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://docs.revenera.com

Using Embedded Help

When you are working on a project in InstallAnywhere, note that clicking a setting in a grid displays help information in the help pane. This help is called embedded help, or context-sensitive help. If you click a different setting, the help pane is updated to show specific information about that corresponding setting.

Contacting Us

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

http://www.revenera.com

2

Getting Started

This section describes the InstallAnywhere interface authoring environments and explains the differences between the different editions of InstallAnywhere. It also explains how to obtain product updates.

- Introduction to the InstallAnywhere Interface
- Upgrading from Earlier InstallAnywhere Versions
- Obtaining Updates for InstallAnywhere

Introduction to the InstallAnywhere Interface

This section describes the InstallAnywhere interface authoring environment and how to work with InstallAnywhere projects.

- About the InstallAnywhere Advanced Designer Authoring Environment
- Working with InstallAnywhere Projects

About the InstallAnywhere Advanced Designer Authoring Environment

InstallAnywhere's Advanced Designer authorizing environment provides fine 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.

Information that you enter in the Advanced Designer is saved directly to the underlying project file.

The InstallAnywhere Advanced Designer has an intuitive, graphical interface that enables you to manage all aspects of your installation projects. The Advanced Designer enables you to add complex and powerful elements to your installation.

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

The Advanced Designer has two main navigational elements:

- The main navigational element is the horizontal row of tabs near the top of the InstallAnywhere interface. You can click these tabs to open various pages—**Project**, **Installer UI**, **Organization**, **Sequence**, and **Build**.
- The secondary navigational element is the vertical column of tabs on the left side of the InstallAnywhere interface. The list of tabs in this area changes, depending on which page in the Advanced Designer is currently open.

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

- To become familiar with using the Advanced Designer interface, review the InstallAnywhere Tutorial.
- For detailed information on the Advanced Designer interface, see Advanced Designer Reference.

Working with InstallAnywhere Projects

This section explains how to create a new InstallAnywhere project or open an existing project.

- Creating a New InstallAnywhere Project
- Opening an Existing InstallAnywhere Project

Creating a New InstallAnywhere Project

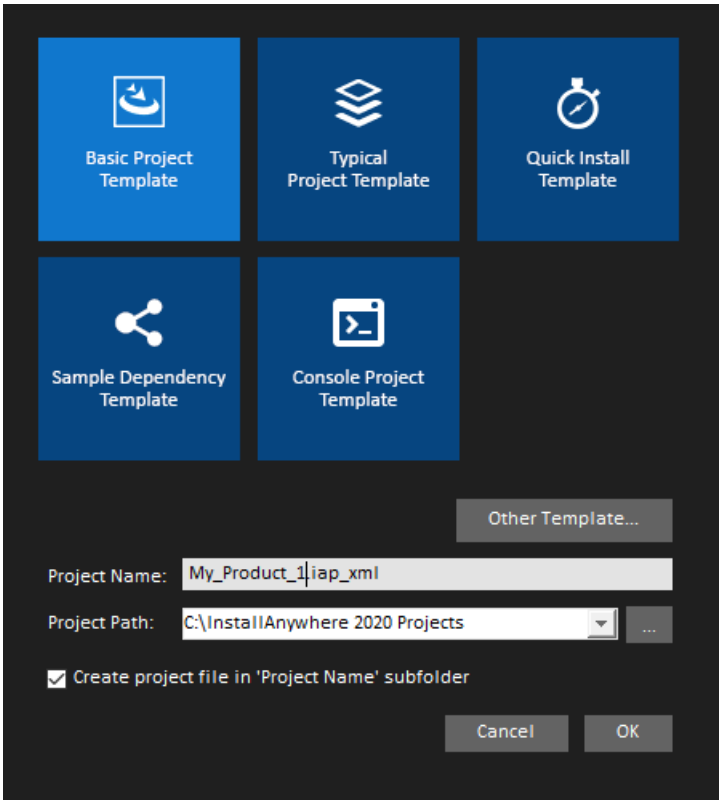
Perform the following steps to create a new InstallAnywhere project.

Task ***To create a new InstallAnywhere project:***

1. Launch InstallAnywhere. The InstallAnywhere **Create/Open Project** dialog box opens.



2. Click the **Create new project** button. The **Create a New Project** dialog box opens.



3. Select a template to use.

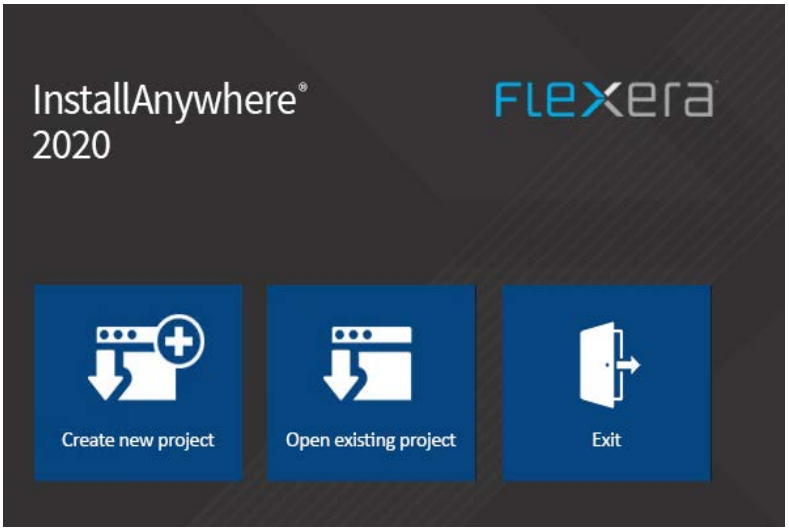
4. Enter a **Project Name** and **Project Path**.
5. Click **OK**. The project opens in the InstallAnywhere interface.

Opening an Existing InstallAnywhere Project

To open an existing InstallAnywhere project, perform the following steps.

Task ***To open an existing project:***

1. Launch InstallAnywhere. The InstallAnywhere **Create/Open Project** dialog box opens.



2. Click the **Open existing project** button. The **Open Project File** dialog box opens.
3. Locate and select the project file that you want to open.
4. Click the **Open** button.

InstallAnywhere opens the project in the Advanced Designer.

Switching to a Different Project

If you are working on one project in the InstallAnywhere user interface, you can easily switch to a different project.

Task ***To switch to a different project:***

5. On the **File** menu, click **Open**. The **Open Project File** dialog box opens.

Note ▪ *If you have made changes to the existing project, you are prompted to save the current project before proceeding.*

6. Locate and select the project file that you want to open.
7. Click the **Open** button. InstallAnywhere opens the project in the Advanced Designer.

Upgrading from Earlier InstallAnywhere Versions

This section describes possible upgrade issues that may occur when you upgrade projects that were created with earlier versions of InstallAnywhere. It also alerts you to possible changes in behavior that you may notice between InstallAnywhere 2026 R1 projects and projects that are upgraded from an earlier version to InstallAnywhere 2026 R1.

- Converting Projects to InstallAnywhere 2026 R1
- Upgrading from InstallAnywhere 2014 or Earlier
- Upgrading from InstallAnywhere 2013 or Earlier
- Upgrading from InstallAnywhere 2012 or Earlier
- Upgrading from InstallAnywhere 2010 or Earlier
- Upgrading from InstallAnywhere 2009 or Earlier
- Upgrading from InstallShield MultiPlatform

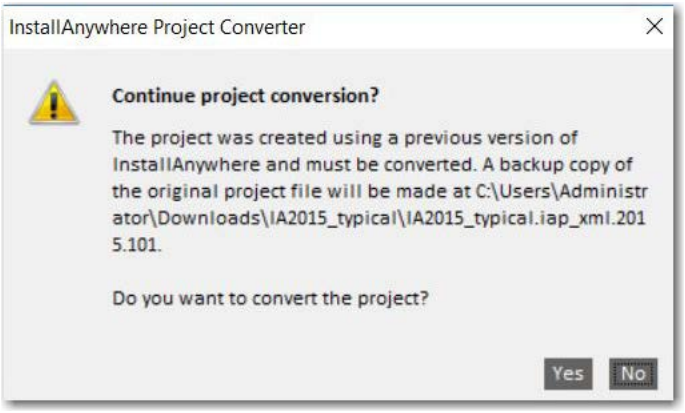
Converting Projects to InstallAnywhere 2026 R1

When you open a project that was created with a previous version of InstallAnywhere, you will be prompted to convert it to 2026 R1 format.

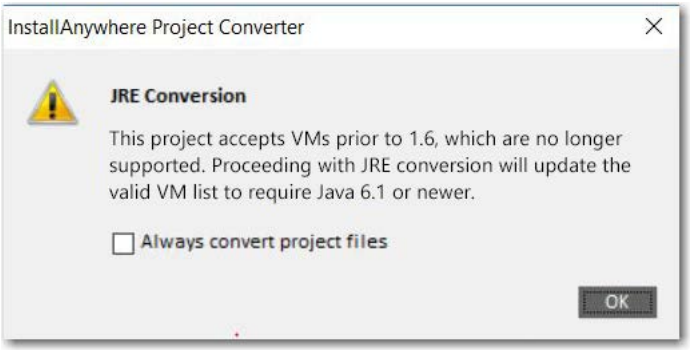


Task ***To convert an existing project to InstallAnywhere 2026 R1 format:***

1. Launch InstallAnywhere. The InstallAnywhere **Create/Open Project** dialog box opens.
2. Click the **Open existing project** button. The **Open Project File** dialog box opens.
3. Locate and select the project file that you want to open and click the **Open** button. The following dialog box opens informing you that the project is going to be converted, and that a backup copy will automatically be created.



4. Click **Yes**. If the InstallAnywhere project accepts VMs prior to 1.6, the following message will be displayed stating that proceeding with JRE conversion will update the valid VM list to require Java 6.1 or later.



5. Click **OK**. The project conversion is completed.

Upgrading from InstallAnywhere 2014 or Earlier

Database Server Support for Microsoft SQL Server and IBM DB2

If you have an InstallAnywhere 2014 or earlier project that contains a Microsoft SQL Server or IBM DB2 or database server host and you open the project in InstallAnywhere 2026 R1, InstallAnywhere displays a warning and informs you that may need to make changes to incorporate the new run-time panel action that lets end users specify connection information. To learn more, see Enabling End Users to Specify Database Connection Information.

InstallAnywhere no longer has support for the following versions of Microsoft SQL Server:

- Microsoft SQL Server 6.5
- Microsoft SQL Server 7.0
- Microsoft SQL Server 2000
- Microsoft SQL Server 2005

InstallAnywhere supports Microsoft SQL Server 2008 R2, 2012, and 2014 databases.

Removal of InstallAnywhere Collaboration Support

All support for the InstallAnywhere Collaboration plug-in and DIM references has been removed from InstallAnywhere. That is, the DIM References view and the DIM-related actions (the Create DIM Reference action and the Create Alias, Link, Shortcut to DIM File action) are no longer be available in InstallAnywhere. It will no longer be possible to build installers that reference DIM files. If you upgrade a project that contains DIM references or DIM actions from InstallAnywhere 2014 or earlier to InstallAnywhere 2026 R1, InstallAnywhere 2026 R1 removes them from your project.

It is recommended that before upgrading a project with DIM support to InstallAnywhere 2026 R1, all DIM references and DIM-related actions be removed from InstallAnywhere projects and replaced with standard files and actions where appropriate.

Upgrading from InstallAnywhere 2013 or Earlier

Changes in OS and OS X Version Requirements for Target Systems

Apple made full authentication support available starting with OS X 10.8. The authentication support that Apple provided in OS X 10.7 was limited and is no longer recommended on OS X 10.7–based systems. If you specify in your InstallAnywhere project that your OS or OS X–based installer requires administrator credentials (Project page > Platforms view > OS X area > Authentication), the installer and corresponding uninstaller that InstallAnywhere builds cannot be run on OS X 10.7–based systems. On OS X 10.8 and later systems including OS 10.12, root users and administrator users can run the installer; standard users need to enter administrator credentials before the installer can continue running.

Apple no longer supports OS X 10.6 and earlier. Therefore, InstallAnywhere no longer supports the creation of installers for OS X 10.6 or earlier. If end users have OS X 10.6 on their machine and they try to run an installer that was built with InstallAnywhere 2026 R1, the installer may run successfully, or unexpected results may occur.

Changes to the List of Supported OS and OS X Versions for Running InstallAnywhere

InstallAnywhere now supports the OS Sierra (10.12) for running installer run-time environment, as well as for the InstallAnywhere authoring environment. The minimum version of OS X that is required for OS X–based systems that run InstallAnywhere (the authoring environment) is now OS X 10.7.5 with Oracle Java 7.

Changes to the List of Supported Windows Versions for Running InstallAnywhere

The minimum version of Windows that is required for Windows-based systems that run InstallAnywhere (the authoring environment) is now Windows Vista or Windows Server 2008 (x86 and x64). Previously, the minimum Windows requirement was Windows XP.

New Requirement for Using Project Automation APIs with Node-Locked Licenses of InstallAnywhere

If you are using a node-locked license of InstallAnywhere, the licensing libraries must be loaded while using the project automation API. This is typically done by configuring an IDE setting or by passing the following parameter through the command line when running the Java class:

```
-Djava.library.path=IA_HOME\resource\fnp\libraries
```

If the library path includes spaces, ensure that you enclose the library path within quotes:

```
-Djava.library.path="IA_HOME\resource\fnp\libraries"
```

Discontinuation of the InstallAnywhere Collaboration Tool; Deprecation of InstallAnywhere Collaboration Support

The InstallAnywhere Collaboration plug-in for Eclipse has been discontinued. This tool is not available for installation when you install InstallAnywhere 2026 R1, and a separate installer that installs only InstallAnywhere Collaboration is not available.

If you have developer installation manifest (DIM) files that were created in an earlier release of InstallAnywhere Collaboration, you can add them to InstallAnywhere 2014 projects. However, support for DIM files in InstallAnywhere 2014 has been deprecated and is no longer supported. In a future release, all support for DIM files will be removed from InstallAnywhere. That is, the DIM References view and the DIM-related actions will no longer be available, and it will no longer be possible to build installers that reference DIM files.

It is recommended that all DIM references and DIM-related actions be removed from InstallAnywhere projects and replaced with standard files and actions where appropriate.

OS and OS X Build Targets and Java Virtual Machine Searches

If you have a OS or OS X build target in your project and its Without VM check box is selected in the Build Installers view on the Build page, the installer searches for Oracle JRE 7 or later on target systems; if it is not present, the installation exits. The Without VM check box no longer has an option to search for Apple JRE 6 or earlier. This applies to all new projects that are created in InstallAnywhere 2026 R1. It also applies to all projects that were created in InstallAnywhere 2013 or earlier and upgraded to InstallAnywhere 2026 R1.

Database Server Support for MySQL, Firebird, Interbase, and Sybase ASE

If you have an InstallAnywhere 2013 or earlier project that contains a MySQL database server host and you open the project in InstallAnywhere 2026 R1, InstallAnywhere displays a warning and informs you that may need to make changes to incorporate the new run-time panel action that lets end users specify connection information. To learn more, see Enabling End Users to Specify Database Connection Information.

InstallAnywhere no longer has support for the following versions of MySQL:

- MySQL 4.1
- MySQL 5.0

InstallAnywhere supports MySQL 5.1.29, as well as MySQL 5.5 and 5.6.

InstallAnywhere no longer has support for connecting to the following database servers and running SQL scripts:

- Firebird
- Interbase
- Sybase ASE

If you have an InstallAnywhere 2013 or earlier project that contains a host that is configured to be one of those server types and you open the project in InstallAnywhere 2026 R1, InstallAnywhere displays a warning and changes the server type of that host to the default type, which is MySQL. You can change the server type to one of the other supported types as needed.

To learn more, see Managing Database Servers.

Application Server Support for Tomcat, Resin, and Sun Application Server

If you have an InstallAnywhere 2013 or earlier project that contains a Tomcat application server host and you open the project in InstallAnywhere 2026 R1, InstallAnywhere displays a warning and informs you that may need to make changes to incorporate the new run-time panel action that lets end users specify connection information. To learn more, see Enabling End Users to Specify Apache Tomcat Server Information.

In addition, InstallAnywhere no longer has support for the following versions of Tomcat:

- Tomcat 5.0
- Tomcat 5.5

If you have an InstallAnywhere 2013 or earlier project that contains a host with a server type of Tomcat 5.0, 5.5, and 6.0 and you open the project in InstallAnywhere 2026 R1, InstallAnywhere changes the server type of that host to Tomcat 6.0.x, 7.0.x & 8.0.

InstallAnywhere no longer has support for deploying Web applications to the following application servers:

- Resin
- Sun Application Server

If you have an InstallAnywhere 2013 or earlier project that contains a host that is configured to be one of those server types and you open the project in InstallAnywhere 2026 R1, InstallAnywhere displays a warning and changes the server type of that host to Tomcat 6.0.x, 7.0.x & 8.0. You can change the server type to one of the other supported types as needed.

To learn more, see Managing Application Servers.

Upgrading from InstallAnywhere 2012 or Earlier

New Build Target for Pure 64-Bit Windows-Based Systems

InstallAnywhere 2026 R1 has support for building installers that target pure 64-bit Windows-based systems on which 32-bit Windows-on-Windows (WOW64) support is disabled.

If you create a new project in InstallAnywhere 2026 R1, it automatically includes a build target for pure 64-bit Windows-based systems by default. If you upgrade an InstallAnywhere 2012 or earlier project to InstallAnywhere 2026 R1, InstallAnywhere automatically adds this build target to your project.

Upgrading from InstallAnywhere 2010 or Earlier

Default Uninstall Behavior for Imported Merge Modules

If you import a merge module into an InstallAnywhere 2026 R1 project, the **Uninstall Merge Module when parent is uninstalled** check box is selected by default. This check box lets you specify whether you want the merge module to be uninstalled from target systems when the parent project is uninstalled.

If you have an InstallAnywhere 2010 or earlier project that includes one or more imported merge modules, and you upgrade the project to InstallAnywhere 2026 R1, the **Uninstall Merge Module when parent is uninstalled** check box is cleared by default for those merge modules.

If you want to change the behavior in either case, you can change the state of the check box:

- To change the behavior for an individual merge module: On the Sequence tab, click Install. In the Visual Tree, select the merge module whose behavior you want to change. In the Install Merge Module customizer, on the Properties subtab, select or clear the **Uninstall Merge Module when parent is uninstalled** check box.
- To change the behavior for a dynamic merge module: On the Organization tab, click Modules. In the Imported Merge Module List, select the dynamic merge module whose behavior you want to change. In the Dynamic Merge Module customizer, on the Install tab, select or clear the **Uninstall Merge Module when parent is uninstalled** check box.

VM Pack Creation

VM packs that are created using the Create VM Pack utility, which was introduced in InstallAnywhere 2011, can be used only with installers that are created in InstallAnywhere 2011 or later, not with installers that were created in earlier versions of InstallAnywhere (such as InstallAnywhere 2009 or InstallAnywhere 2010). If you want to use VM packs with any older installers (those that were created in InstallAnywhere 2010 or earlier), you must follow the manual process, as described in Creating a JRE VM Pack Manually.

Project Automation APIs and Configuring Behavior for Maintenance Mode and Instance Management

If you are upgrading an InstallAnywhere 2010 (without SP1) project to InstallAnywhere 2026 R1 and you want to use the project automation APIs to modify maintenance mode and instance management in your project, it is recommended that you first open and save your project in the InstallAnywhere 2026 R1 Advanced Designer.

Configuring Build Target Information through the BuildProperties.xml File

If you upgrade an InstallAnywhere 2010 or earlier project to InstallAnywhere 2026 R1 and you were using a `BuildProperties.xml` file to configure build targets, note that some of the XML code has been changed. In InstallAnywhere 2011 and later, the `buildWithVM`, `BuildWithNoVM`, `outputDir`, and `bundledVM` settings are attributes of the `<target>` element. In InstallAnywhere 2010 and earlier, these settings were subelements of the `<target>` element. Therefore, if you upgrade an InstallAnywhere 2010 or earlier project to InstallAnywhere 2026 R1, ensure that you update your `BuildProperties.xml` file accordingly.

Configuring Build Target Information through Ant Tasks

If you upgrade an InstallAnywhere 2010 or earlier project to InstallAnywhere 2026 R1 and you were using an Ant task to configure build targets, note that some of the XML code has been changed. In InstallAnywhere 2011 and later, the `buildWithVM`, `BuildWithNoVM`, `outputDir`, and `bundledVM` settings are attributes of the `<target>` element. In InstallAnywhere 2010 and earlier, these settings were subelements of the `<target>` element. Therefore, if you upgrade an InstallAnywhere 2010 or earlier project to InstallAnywhere 2026 R1, ensure that you update your Ant task accordingly.

Upgrading from InstallAnywhere 2009 or Earlier

Change to the Default Name of the Uninstaller Executable File

If you create a project in InstallAnywhere 2026 R1, the default name for the uninstall launcher is `Change ProductName Installation.exe`.

If you created a project in InstallAnywhere 2009 or earlier, the default name for the uninstall launcher was `Uninstall ProductName.exe`; if you upgrade the project to InstallAnywhere 2026 R1, InstallAnywhere does not change the name of the uninstaller launcher automatically. You may want to manually change the name. To do so: On the Sequence page, click Install. Then select the uninstall launcher and edit the value in the Name setting on the Properties tab.

Changes for Organizing Build Settings

InstallAnywhere has support for creating and maintaining different build configurations. Each build configuration defines how InstallAnywhere builds an installer for a particular set of platforms, build distributions, JVMs, locales, and other settings. You can store as many build configurations with a project as necessary. When you create a new project in InstallAnywhere 2026 R1, it contains a default build configuration called *Default Configuration*.

InstallAnywhere 2009 and earlier did not have support for build configurations. If you open an InstallAnywhere 2009 or earlier project in InstallAnywhere 2026 R1, InstallAnywhere displays the Build Settings Conversion Notice message box, which asks you if you want InstallAnywhere to convert the existing build settings automatically to the build configuration model. If you reply that you do want to convert it, InstallAnywhere upgrades all of the project's existing build settings into a new build configuration called *Migrated Configuration*. If you want to modify the settings for this build configuration, you can do so. To learn more, see:

- Defining Build Targets
- Specifying Build Distribution Options
- Using Tags to Customize Build Configurations
- Specifying Locale Settings

Changes for Locales

In InstallAnywhere 2009 and earlier, the locales directory in the build output folder was named <project_name>locales. If you are upgrade an InstallAnywhere 2009 or earlier project to InstallAnywhere 2026 R1, you must rename this directory with the following naming convention:

<project_name>locales_<build_configuration_name>

Once you have renamed the folder, rebuild the project in InstallAnywhere 2026 R1.

Changes Related to Rollback Behavior

InstallAnywhere has built-in support for rollback behavior: On the Rollback tab of action customizers in the Install view of the Sequence page, you can specify rollback behavior 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.

InstallAnywhere 2009 and earlier did not have built-in support for rollback behavior; these versions included support for custom code that triggered a rollback handler; if an end user cancelled the installation, the rollback handler was called.

If you upgrade a project from InstallAnywhere 2009 or earlier with the rollback handler code to InstallAnywhere 2026 R1, the rollback handler code is launched only if the Enable Rollback check box is cleared; this check box is in the Advanced view on the Project tab. To learn how to use the new built-in rollback behavior, see Configuring Installation Rollback Behavior.

Upgrading from InstallShield MultiPlatform

Changes for Organizing Build Settings

InstallAnywhere has support for creating and maintaining different build configurations. Each build configuration defines how InstallAnywhere builds an installer for a particular set of platforms, build distributions, JVMs, locales, and other settings. You can store as many build configurations with a project as necessary. When you create a new project in InstallAnywhere 2026 R1, it contains a default build configuration called *Default Configuration*.

If you use a project manifest to upgrade an InstallShield MultiPlatform project to InstallAnywhere 2026 R1, InstallAnywhere creates a new build configuration called *Migrated Configuration* for the upgraded project. If you want to modify the settings for this build configuration, you can do so. To learn more, see:

- Defining Build Targets
- Specifying Build Distribution Options
- Using Tags to Customize Build Configurations
- Specifying Locale Settings

Obtaining Updates for InstallAnywhere

If you have an Internet connection, you can use the Check for Updates feature in InstallAnywhere to obtain the latest InstallAnywhere service packs and other updates for the version of InstallAnywhere that you are using. You can also configure InstallAnywhere to automatically check for updates.

- Manually Checking for Updates
- Installing a Critical Update
- Installing a Non-Critical Update
- Configuring InstallAnywhere to Automatically Detect Updates

Manually Checking for Updates

To manually check for updates, perform the following steps.



Task ***To manually check for updates:***

1. On the **Help** menu, click **Check for Updates**.
    - **If a file sync update is available**, the **Update Available** dialog box opens. A file sync update represents a critical update to the product.
    - **If a noncritical update or message is available**, a FlexNet Connect window opens, enabling you to see a list of available updates.

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, specify 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 opens, prompting you to save the project. Once you save your changes, InstallAnywhere closes, and a dialog box that shows the progress of the update opens.
   - Click **On Exit** to install the available update the next time that you close InstallAnywhere. Once you close InstallAnywhere, a dialog box that shows the progress of the update opens.
   - Click **Later** to dismiss the dialog box. The update is not installed, but this update is still available the next time that you check for updates.
2. When the update is downloaded and installed, click the **Finish** button.



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

To install a non-critical update, perform the following steps.



Task ***To install a non-critical update or message:***

1. On the **FlexNet Connect** window, in the list of updates and messages, select the update that you want to install.
2. Click **Install**. The update installation starts.

Configuring InstallAnywhere to Automatically Detect Updates

To configure InstallAnywhere to automatically detect when updates are available, perform the following steps.



Task ***To configure InstallAnywhere to automatically detect when updates 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 server address and port number, select the **Use the following HTTP proxy setting** check box and enter the proxy address and port in the appropriate boxes.
5. Click **OK**.



Note ▪ *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 Preparing Your Installer for Update Notifications.*

3

Concepts of InstallAnywhere Installer Development

InstallAnywhere provides many flexible options for creating installer functionality. This section introduces you to the basic of installer development in InstallAnywhere.

Table 3-1 ▪ InstallAnywhere Installer Development Concepts

Topics	Description
Actions	Explains InstallAnywhere's support for an extensible action architecture that gives you 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.
Hosts	Explains the different types of server 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 (which is also suitable for other physical media such as DVDs).
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.

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

Topics	Description
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.

Table 3-2 ▪ Topics About Actions (cont.)

Topic	Description
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

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.

You can schedule actions to various sequences (Pre-Install, Install, Post-Install, Pre-Uninstall, Uninstall and Post-Uninstall) in your projects.

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.

Table 3-3 ▪ Actions (cont.)

Action	Description
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 enable you to apply rules to a collection of actions and to make InstallAnywhere projects more manageable and easier to understand. They give you the ability to logically group a set of actions or panels in the Install sequence, as well as other sequences. Rules that are applied to an action group affect all of the actions or panels that are included in the group. 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.

Table 3-4 ▪

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. To learn more, see Assigning Tags to Project Elements.
- **Rollback**—Use the Rollback tab to specify rollback options for the selected action. For more details, see Configuring Installation Rollback Behavior.

About the Action Execution Sequence

Actions are executed in the order that they appear from top to bottom in a view on the Sequence page. You can use the up and down arrows to change the order of actions. The left and right arrows move actions out of or into folders.

Pre-Install and Pre-Uninstall actions are executed before their corresponding Install and Uninstall actions, which are executed before Post-Install and Post-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 that are added to the Install sequence 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 are not automatically uninstalled.



Note ▪ *When an action is scheduled for the Pre-Install or Post-Install sequence, it may be executed more than once if an end user repeatedly clicks Previous and then Next repeatedly. 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, it is recommended that you schedule these actions during the actual installation (within the Install sequence).*

About Get User Input Panels

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

- Mandatory Field

Input Methods/Component Types

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, check boxes, radio buttons, pop-up menus, or lists.
- **Components** available on the Get User Input - Advanced panel include above input methods in addition to choice groups, file choosers, and labels.

  The following table describes some of the commonly used settings (selection boxes) available in the **Control Properties** area of the **Configure File Chooser Field** dialog box—opens when you click the **Configure Selection** control (button) in the **General Settings** tab of the **Get User Input – Advanced** panel:

Table 3-5 ▪ Control Properties Area Settings

Settings	Description
Control Alignment	Allows you to manage the alignment of text fields. By default, the Alignment selection will be **Left**. There are two possible values for the text alignment controls that is **Left** and **Right**.
Remove Restore Button	Allows you to manage the display of the **Restore Default** button on the File/Folder Chooser component in the Get User Input panel during installation and uninstallation. You can configure the **Remove Restore Button** selection box based on your requirement. Setting this selection box to **Yes** removes the **Restore Default** button, while setting it to **No** retains it. By default, the selection box is set to **No**.
Restrict Selection Type	Allows you to control whether a directory or file path manually entered in the File Chooser component on the Get User Input panel during installation or uninstallation must match the option/path type (**Directory** or **File**) configured in the **File Chooser Type** selection box. Setting the **Restrict Selection Type** selection box to **Yes** restricts users to enter only the directory or file path that matches the option/path type configured in the **File Chooser Type** selection box, while setting it to **No** allows the user to enter any directory or file path, regardless of the option/path type configured in the **File Chooser Type** selection box. By default, the **Restrict Selection Type** selection box is set to **No**.

Bidi Orientation and Text Reading Order

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

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

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 3-6 ▪ User Input Results By Input Type

Input Type	Description

Textfields	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.
	
	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:
	
	`$X$="Peter","Paul","Mary"`
	`$X_1$=Peter`
	`$X_2$=Paul`
	`$X_3$=Mary`
	
	However, if only two values were inserted (**Peter** and **Mary**), the values are:
	
	`$X$="Peter","","Mary"`
	`$X_1$=Peter`
	`$X_2$=`
	`$X_3$=Mary`

Check Boxes	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 check boxes (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.
	
	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:
	
	`$X$="Paul","","George"`
	`$X_1$=Paul`
	`$X_2$=`
	`$X_3$=George`

Table 3-6 ▪ User Input Results By Input Type (cont.)

Input Type	Description

List	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 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 list entry that was selected.
	
	Lists behave somewhat differently than check boxes. 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:
	
	`$X$="Paul","","George"`
	`$X_1$=Paul`
	`$X_2$=`
	`$X_3$=George`

Radio Buttons and Popup Menus	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. Additional variables are not derived from the Results Variable unless Defaults are specified. 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: `$X$="Michael","",""`
Control Alignment	The **Control Alignment** selection box on the **Get User Input - Advanced** > **Configure Textfield** or **Configure File Chooser Field** dialog box, in the **Control Properties** area, allows you to manage the alignment of text fields. By default, the Alignment selection will be **Left**. There are two possible values for the text alignment controls that is **Left** and **Right**.
Editable	The Editable option will help to configure Textfields to display as read-only in the **Configure Textfield** dialog box. ● By default, the selection box will be set to True and will remain editable. ● When the Editable option is set to False, it is non editable.

Defaults

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** (check boxes, 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

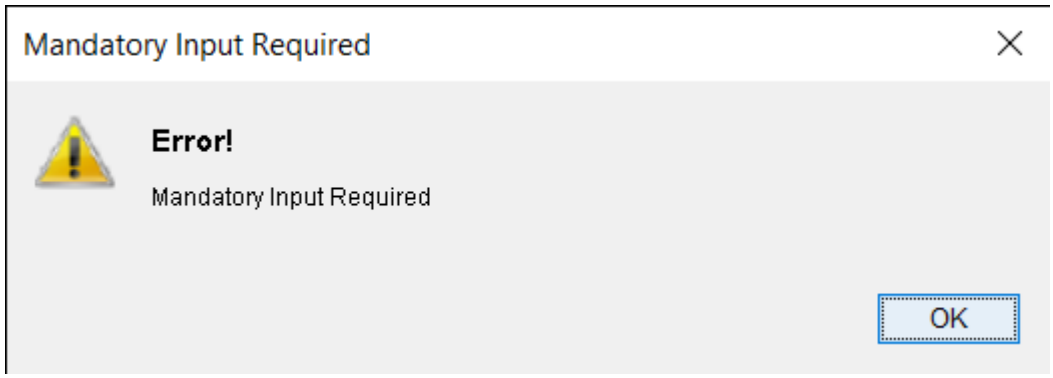
There is also a variable to correspond with the *USER_INPUT_RESULTS_1*, *USER_INPUT_RESULTS_2*, and *USER_INPUT_RESULTS_3*. If your installer requires localization, the values and characters that correspond with this user input panel may vary from language to language. The *VAR_BOOLEAN* variable deals 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
```

Mandatory Field

Mandatory Field control is added for **Textfield** and **Choice Group**. It controls the navigation of Installer based on value/selection of component. Component if marked `Yes`, the component value cannot be left blank at runtime. Component or subcomponent of installer marked `Mandatory` and without value/selection will notify the user with one of the interpretations,

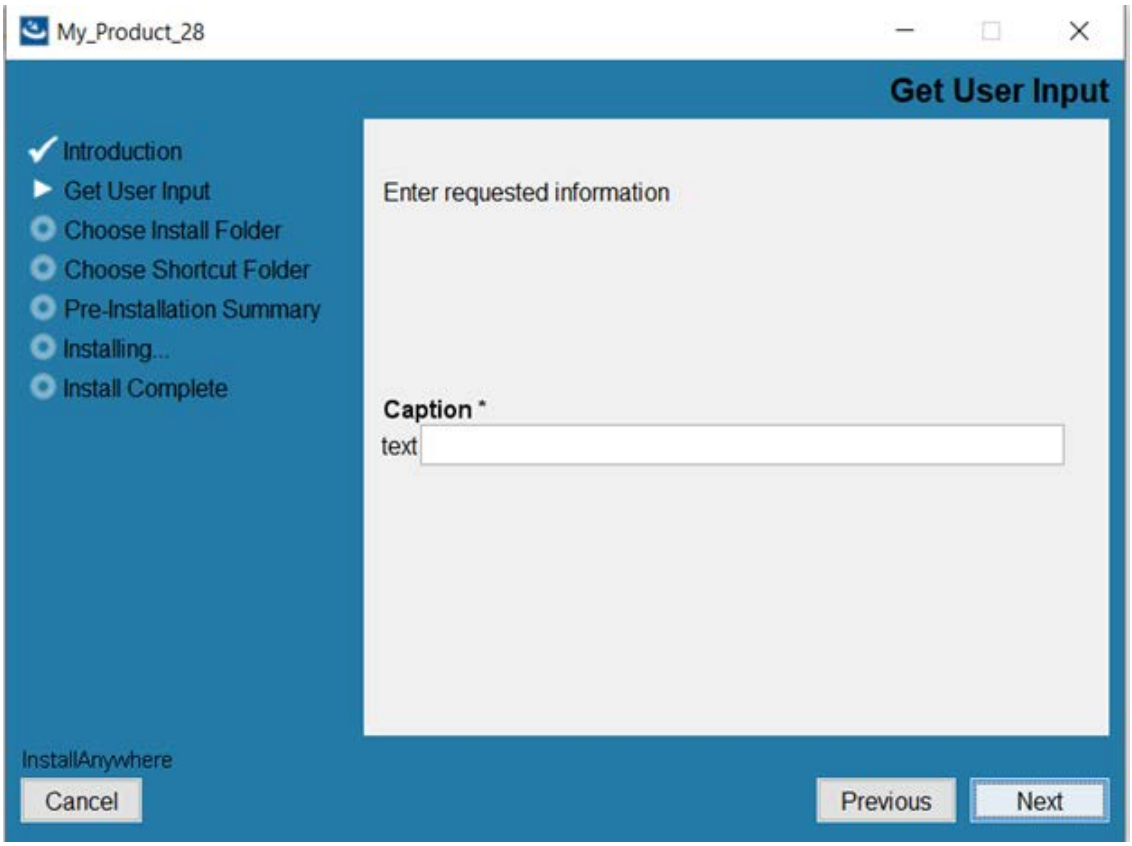
- the following warning message at Installer runtime and will not allow navigation to next panel.



or

- disable the **Next** button if the **Disable Next button if mandatory fields are empty** checkbox is selected in the **Get User Input - Advanced** panel.

To indicate component or subcomponent as mandatory, add Caption to component or subcomponent to display it in red color at runtime.



If `Mandatory Field` is marked as **No** for component, it allows with blank value and no selection in choice group and does not stop navigation to next panel.

Note ▪ *Font and Color selection should be revised when Mandatory Field value is changed from **Yes** to **No** for a component.*

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 that are 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, you can create your own custom components using the Custom Code API.

Table 3-7 ▪ Custom Code

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 3-8 ▪ 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.

Table 3-8 ▪ Major Forms of Custom Code (cont.)

Custom Code	Description
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.*

Hosts

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

- **Operating System host** is the default host 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, Tomcat, WebLogic, and WebSphere.
- **Database Server hosts** target a database server such as MySQL, Oracle, Microsoft SQL Server, and DB2; these hosts enable you to connect to databases on local or remote servers and run SQL scripts.

Support for Application Servers

InstallAnywhere includes support for the following application servers.

Table 3-9 ▪ Supported Application Servers

Application Server	Versions	Required JVM
Geronimo	1.1.1 or newer	1.4 or later
JBoss	4.0.5 or newer	1.4 or later
Tomcat	6.0.x, 7.0.x, and 8.0	1.4 or later
WebSphere	7, 8, and 8.5	1.6 or later

Table 3-9 ▪ 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 JBoss. WAR and EAR deployments to that server requires the presence of connection libraries on the target system.*

Support for Database Servers

InstallAnywhere 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).

Table 3-10 ▪ Supported Databases

Database	Versions
DB2	9.0, 9.7, 10.1, and 10.5
MySQL	5.1.29, 5.5, 5.6
Microsoft SQL Server	2008 R2, 2012, and 2014
Oracle	10.2.0.1, 10.2.0.2, 10.2.0.3, and 10.2.0.4
PostgreSQL	7.0-8.3
Generic JDBC Connection	n/a

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 end 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 end 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

A component is the smallest installable part of a product from the installation developer's perspective; components are invisible to the end user.

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 features. 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.

The following types of components are available in InstallAnywhere:

- Standard Components
- Dependencies
- Shared Components

Each component type has different install, uninstall, and upgrade behavior.

Standard Components

Standard components are always installed. It does not matter if the component is already installed on the target 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 uninstallation time, the uninstaller uninstalls the component even though it did not install it—unless another installed application still needs it.

The installer searches the product registry to check if a component with the component ID of the specified dependency 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 sets a series of InstallAnywhere variables based on its findings. The Matched Key File location variable contains 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 is an empty string. If the dependency check fails, the Dependency State Variable contains the Dependency Failed Message.

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

Shared Components

A shared component is a component that is installed on a target system if it is not already present. A shared component can be made available to other products on target systems.

During uninstallation or upgrade, a shared component is removed only if no other product on the target system references it. The uninstaller for the last product that references the shared component removes it.

If one or more products on the target system reference a shared component, an upgrade may overwrite it and lead to unexpected results in the other products that reference it. For more information, see Special Considerations for Upgrade Support.

Installer Modes

InstallAnywhere installers can run in three different modes:

- **Graphical user interface (GUI)**—GUI mode renders an installer with wizard panels and dialog boxes.
- **Console**—Also known as command-line interface, console-mode installers can perform remote installations over telnet or on systems without a graphical windowing environment.
- **Silent**—Silent installers do not interact with end users. They are suitable for distribution when all of the settings are already known, or they are provided in a response file.



Note ▪ *For more information on using response files, see Silent Installers.*

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. InstallAnywere supports 31 different locales.

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 ▪ *InstallAnywhere also supports an alternative localization scheme based on the use of custom resource bundles. See External Resource Bundles for more information.*

Customization of the User Interface

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 static GIF, an animated GIF, a PNG, or a JPEG file of different sizes. It is recommended to use the preferred size of 470 X 265 pixels.

The preferred sizes for the different DPI are:

- Image (100% DPI) - 470 pixels for the width and 265 pixels for the height.
- Image (150% DPI) - 520 pixels for the width and 315 pixels for the height.
- Image (200% DPI) - 570 pixels for the width and 365 pixels for the height.



Note ▪ *These sizes are not recommended for InstallAnywhere versions before 2021 and sizes apply to InstallAnywhere 2021 and later versions.*

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 images that appear in the large right pane of the installer while files are being installed. You can add billboards to your projects to display information to end users during the installation process. The billboards can be used to communicate, advertise, educate, and entertain end users. For example, billboards can present overviews on new features of the product being installed or other products from your company. Each billboard is a file that you or your company's graphics department creates for control over the look and feel of your installers. Each billboard in the project is displayed for the same amount of time, based on actions within the installation. If an installation has very few files, and many billboards, each billboard is displayed for only a short time.

Billboards can be static 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.

If you specify more than one billboard for your project, you can control how much time each billboard is displayed while the installation is in progress by using the **Display billboards for every** *nn* **seconds** option. Note the following details about using multiple billboards in a project:

- The default time for a billboard to be displayed is 3 seconds.
- To change this time period, select the **Display billboards for every** *nn* **seconds** option and enter a number in the seconds box.
- 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.
- If the install phase completes or is canceled before all billboards are displayed for the specified time period, some of the specified billboards may not be displayed.

Console Installers

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
```

Behavior of Console Launchers

If an installer is bundled with a console launcher, the behavior of the installer varies depending on how the installer was launched. The following table identifies how the console launcher will behave in each situation (Launch Type).

Table 3-11 ▪ Behavior of Console Launchers

Default Windows UI Mode	Launch Type	Ctrl Key Pressed	Expected Result
GUI	Double-click	No	There will not be any console sub-window.
GUI	Double-click	Yes	A console sub-window will display.
GUI	Console with switch	No	There will not be any console sub-window.
GUI	Console with switch	Yes	There will not be any console sub-window.
GUI	Console without switch	No	There will not be any console sub-window.
GUI	Console without switch	Yes	There will not be any console sub-window.
Console	Double-click	No	A console window will display.
Console	Double-click	Yes	A console window will display.
Console	Console with switch	No	There will not be any console sub-window.

Table 3-11 ▪ Behavior of Console Launchers (cont.)

Default Windows UI Mode	Launch Type	Ctrl Key Pressed	Expected Result
Console	Console with switch	Yes	There will not be any console sub-window.
Console	Console without switch	No	There will not be any console sub-window
Console	Console without switch	Yes	There will not be any console sub-window.
Silent	Double-click	No	There will not be any console sub-window.
Silent	Double-click	Yes	A console window will display.
Silent	Console with switch	No	There will not be any console sub-window.
Silent	Console with switch	Yes	There will not be any console sub-window.
Silent	Console without switch	No	There will not be any console sub-window.
Silent	Console without switch	Yes	There will not be any console sub-window.

Note the following regarding the information in this table:

- Launching the installer by double-clicking, which was built using the console launcher, will now kill the spawned console window immediately after launching the installer. That means that the initial (spanned) console window will display for a fraction of second.
- If the installer is built with a console launcher and is launched from the command prompt, the prompt will wait until the launched process is complete before opening a new prompt.
- In the above table, the term “switch” means any kind of supported command-line switches, such as `-i silent` or `-i gui`.

Silent Installers

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 OS or 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, the installer has no-end user interaction, and runs by using either of the following:

- Default values that the installer developer provided
- A response file that contains the values for various InstallAnywhere variables that are used to control the installation

A response file contains a record of a specific installation session. The installer 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 configuring a setting in the Advanced Designer or by using the `-r` command-line switch. To learn how, see Generating Response Files.



Important ▪ *A response file must be saved with the appropriate encoding.*

For Windows-based target systems, the response file must be saved in one of the following encodings:

- *UTF-8 without a BOM*
- *UTF-16 little endian*

For Linux-based and OS or OS X–based target systems, the response file must be UTF-8 without a BOM.

If an unsupported encoding is used, the installer is unable to read the file properly.

The silent installs on only support the -i silent parameters and response files, not passing InstallAnywhere variables, such as $USER_INSTALL_DIR$ using the -DUSER_INSTALL_DIR=/Applications/TestFolder syntax.

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 use 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` lets you 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 that 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 that you would like to install.



***Note** ▪ A JVM version defined in the `installer.properties` or `[installername].properties` file is recognized or applied when launching an application installed via a silent installer. This ensures both the application (installed using a silent installer) and uninstaller launchers utilize only the JVM version specified in the response file, instead of the available installer's default JVM.*



***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. They use a self-extractor to prepare the source files at the start of the installation, and therefore require more available temporary disk space.

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-generated installers are Java based. They require a Java virtual machine (JVM) to run.

You can bundle a JVM with your installer, thus eliminating the possibility that the installer will not find a suitable JVM for the product or for the installer. You also have the ability to configure how the installer searches for a JVM and customize the search that the installer does on behalf of the launchers it deploys.

Finally, you can use the Choose Java VM panel in the Pre-Install sequence of your installer to enable end users to choose from a set of JVMs to use for the products that they are installing.

- About JVM Selection
- How the Installer's Launcher Selects a JVM
- About the Installer's JVM Search
- About the Launcher's JVM Selection Behavior at Run-Time
- About the Choose Java VM Panel
- When Are VM Packs Installed?
- Configuring Upgrade Installers that Use a Different JRE from the Previous Installer
- About Java VM Selection Criteria

About JVM Selection

The Java virtual machine that is selected to run an InstallAnywhere-generated installer, or any Java application that was installed by an InstallAnywhere-generated installer, depends on the settings in your InstallAnywhere project and the Java executable files that are available on the end user's system.

The core of the JVM selection process is performed by InstallAnywhere's LaunchAnywhere technology. LaunchAnywhere launchers are capable of performing their own JVM search each time that they run. However, it is also possible to use any of the following methods for JVM selection:

- Bundle a JVM with the installer to be used by the installer and, optionally, to be used by the launchers that your installer deploys. See Adding a VM Pack to Your Project and Customizing Individual Launcher Settings.
- Employ the results of the installer's JVM search. The installer's JVM search is a customizable scan of end users' systems. The results of this search can then be used to configure the launchers that the installer deploys.
- Include in the Pre-Install sequence of your project a Choose Java VM panel. This panel lets end users specify whether they want to use the bundled JVM, use the local JVM that matches the installer's JVM search criteria, start a JVM search in a non-standard directory, or browse for a local Java executable file.

Basic JVM 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 both of the following are true:

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

A launcher first attempts to use the JVM that is specified in its current VM property (`lax.nl.current.vm`). If the JVM that is specified by that property is either unavailable or invalid, the launcher searches the system for a valid JVM.

When a LaunchAnywhere launcher must search for a JVM, it scans common system locations for Java executable files and uses the first one that it finds that matches the criteria in its valid JVM list (`lax.nl.valid.vm.list`).



***Note** ▪ The default JVM selection criteria that installers use is version 1.5 or later. For information on changing the valid JVM list for the installer's launcher, see Controlling the JVM That Your Installers Use. For information on changing the valid JVM list for a launcher that your installer deploys, see Customizing Individual Launcher Settings.*

Impact of Bundled JVMs, Installer's JVM Search, and the Choose Java VM Panel

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

- **Bundled JVM**—Include a bundled JVM with your installer. See Controlling the JVM That Your Installers Use.
- **Custom JVM search**—Customize the installer's JVM 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 that the installer deploys, the JVM selection process can be much more complex. The JVM selected depends on:

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

For detailed discussions of these conditions, see Controlling the JVM that Your Launchers Use and Using the Choose Java VM Panel.

Behavior that Triggers a JVM Search

The installer's launcher, the installer itself, and the installed LaunchAnywhere launchers are all capable of making JVM selection decisions and performing JVM searches—some during the install process and some after. The following behavior triggers when a JVM search is performed.

- **The installer's launcher determines which JVM to use to run the installer**—When an end user runs an InstallAnywhere installer, the installer's launcher searches the target system for a JVM bootstrap that meets the criteria that are expressed in your project's installer valid VM list (which is set in the Advanced Designer: Project page > JVM Settings view > Installer Settings tab > Valid VM list). For more information, see Controlling the JVM That Your Installers Use.
- **The installer can determine which JVM the LaunchAnywhere launchers use**—Once the installer is running, it performs a search for a JVM to use with any LaunchAnywhere launchers it will install. The criteria the installer uses for this search depends on JVM search settings you provide on the Search Panel Settings tab of the Project > JVM Settings task. For details, see Controlling the JVM that Your Launchers Use.

  These settings can be further customized for Windows-based and UNIX-based installers. See Customizing the VM Search Paths and Patterns.
- **The LaunchAnyhwere launchers can determine which JVM to use**—LaunchAnywhere launchers are also capable of performing an independent JVM search when they are executed. LaunchAnywhere performs this search when it does not receive a JVM setting from the installer or when specifically set to perform its own search. For information on how to require LaunchAnywhere to perform its own JVM search, see Customizing Individual Launcher Settings.

Locations on a Target System that Are Searched for JVMs

The common locations that are examined during a JVM search that is performed by an InstallAnywhere launcher or an installer are similar.

Table 3-12 ▪ Location that Are Searched for JVMs

Platform	Search Location

Windows	The directories that are listed in the PATH environment variable as well as the following registry keys are searched for JVMs:
	`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. The installer JVM search includes the PATH environment variable as well as the following paths:
	`/bin`
	`/usr/bin`
	`/usr/local/bin`
	`/opt`
	`/opt/gnu/bin`
	`/usr/gnu/bin`

How the Installer's Launcher Selects a JVM

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

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

The installer's launcher searches the end user's machine for a JVM that matches the "bootstrap" JVM search criteria set for the installer's installer valid VM list (which is set in the Advanced Designer: Project page > JVM Settings view > Installer Settings tab > Valid VM list). If a JVM has been bundled with the installer, the installer uses the bundled JVM. If no JVM is bundled with the installer, the installer uses the first JVM on the target system that matches the "bootstrap" search criteria. If no JVM 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 JVM pack. Free JVM packs are available on the InstallAnywhere Web site. For more information, see Working with JRE VM Packs.*

About the Installer's JVM Search

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

- **Criteria used in bootstrap phase**—The criteria that the installer's launcher uses in the bootstrap phase (the installer valid VM list, which is set in the Advanced Designer: Project page > JVM Settings view > Installer Settings tab > Valid VM list) can be used.
- **Criteria in lax.nl.valid.vm.list property**—The criteria that are 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 JVM that meets all the JVM criteria of these launchers.
- **Custom criteria**—Custom criteria that are set specifically provided for the project's launchers can be used. You specify this criteria using the **Use a specific valid VM list** setting (which is set in the Advanced Designer: Project page > JVM Settings view > Search Panel Settings tab > General subtab).



***Note** ▪ The installer applies the result of this search to your launchers only when the installer includes a bundled JVM 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-based and UNIX-based target systems, InstallAnywhere also supports further refinement of the JVM search paths and search patterns (Java executable patterns). These describe the locations in which the installer searches for valid JVMs and the patterns it uses to recognize Java executable files on those platforms. For more information, see Customizing the VM Search Paths and Patterns.

Every LaunchAnywhere launcher is capable of performing an independent JVM 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 JVM Selection Behavior at Run-Time

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

- Use the JVM that is bundled with the installer.
- Use the JVM that was found by the installer's JVM search.
- Use the JVM that was found by the installer's launcher during the bootstrap search (the same JVM that was used to run the installer when the installer has no bundled JVM).
- Use the JVM that is found by an independent LaunchAnywhere JVM search.
- Use the JVM that is identified by an end user's search in a non-standard path.
- Use the JVM that is 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 sequence. 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
- The first VM found in the system matching the VM Search Settings defined under 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. If you choose this option on the Advanced Settings tab, the result depends on whether a Choose Java VM panel is included in the project:

Table 3-13 ▪ Result of Choosing the "VM Selected by the Installer or by the End User" Option

Scenario	Result
The 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: ● **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 `lax.nl.valid.vm.list` 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.
The Choose Java VM Panel is included	If a Choose Java VM panel is included in the project, the installer configures the launcher to use the VM that the end user selects. Depending on the settings in the Choose Java VM customizer and whether a bundled VM is installed, the end user may do one of the following: ● Choose the bundled JVM. ● Choose the JVM that matched the installer's VM search. ● Browse to locate a specific JVM. For details, see Using the Choose Java VM Panel.

VM Used by the Installer

This option configures the launcher to use the VM that was used to run the installer. Hence the VM that the launcher will use depends on whether 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 that the installer found using the criteria in the installer valid VM list(, which is set in the Advanced Designer: Project page > JVM Settings view > Installer Settings tab > Valid VM list). This is the same VM that was 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 (Project page > JVM Settings view) result in a build error:*

- *The bundled VM is not configured to be installed.*
- *The bundled VM is configured to be installed only when a compatible VM is not found on 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 (Project page > JVM Settings view).

- **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 *▪ Consider the following information:*

- *You can customize the default value (11.0+) 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.*
- *Now, only JVMs located in secure folders are permitted to launch the installer with administrative privileges on the Windows platform. JVMs found in unsecured folders are not permitted for use and are restricted by the launcher with a security error message. This behavior can be turned off by using one of the following methods:*
  - *Configure the* `designer.jvm.securefolder.check` *property to* `false` *in the* `com.zerog.ia.Designer.properties` *file.*
  - *Configure the* ***Secure JVM Folder Check*** *setting to* ***Disable*** *in the* ***Project*** *>* ***Platforms*** *>****Windows*** *area.*
  - *Run the installer using the* `<installer_name> -Djvm.securefolder.check=<false>` *command line, where* `<installer_name>` *refers to your installer executable.*

About the Choose Java VM Panel

In addition to the JVM selection decisions that the installer and its launchers make, you can provide your end users with the option to choose a VM that they want your product 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 JVM found by the installer's JVM search.
- Choose the bundled JVM (only shown when a bundled JVM is installed).
- Search for a JVM in non-standard locations.
- Browse for a JVM 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 JVMs and presents those JVMs that match the selection criteria to the end user. The options for choosing the bundled JVM, searching non-standard locations, and browsing for a specific Java executable file are represented by check boxes 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 executable files 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 (Project page > JVM Settings view).
- 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 executable files were not installed (including the Uninstaller's LaunchAnywhere executable file).



Note ▪ *For more information about VM packs, see Working with JRE VM Packs.*

Configuring Upgrade Installers that Use a Different JRE from the Previous Installer

If you are creating an upgrade installer that contains a JRE with a different processor type or version than the previous version of the installer, problems could occur during installation of that JRE.

To prevent those problems, select the **Advanced JRE handling for Version Upgrades** option on the **Project > Installer Settings** tab of the **JVM Settings** view.

Selecting the **Advanced JRE handling for Version Upgrades** option will enable Maintenance Mode and Instance Management Panels to display the existing instances of the same product across both 32-bits and 64-bits. If this option is not selected, the panels will display only instances which correspond to the bits of the upgrade installer.

The following are scenarios when you should select this option:

- **New installer has a JRE with a different processor type than previous**—For example, you are currently building App A V2 with a 64-bit JRE version X, but App A V1 had a 32-bit JRE version X.
- **New installer has a JRE with a different version than previous**—For example, you are currently building App A V2 with a 64/32-bit JRE version X, but App A V1 had a 64/32-bit JRE version Y.

If you are creating an upgrade installer that includes a JRE with a different processor type or version than the previous version of the installer, perform the following steps.



Task ***To specify advanced JRE handling for version upgrades:***

1. Open the InstallAnywhere project.
2. Open the **Project > Installer Settings** tab of the **JVM Settings** view.
3. Under **Bundled / Downloaded Virtual Machine**, select the **Advanced JRE handling for Version Upgrades** option.

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 that is required by your installer or your installed product. If no JVM 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.

JVM selection criteria can be used to specify valid VMs (Project page > JVM Settings view) and for the LaunchAnywhere (`lax.nl.valid.vm.list` property).

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

Table 3-14 ▪ Java VM Selection Criteria

Property	Description
VM Type	VM type values include: ● ALL (any VM) ● JDK (any JDK) ● JRE (any JRE)  ***Note ▪*** *The optional JDK or JRE specifies which type of VM is valid.*
VM Version	Version values can be specific versions, such as 11.0.15_0 or partial versions that include a + or * wildcard character. The version number can have varying degrees of precision; however, it is recommended that you specify at least the major and minor version numbers: ● **JRE_11.0+**—This entry indicates that any JRE of version 11.0.0_0 or later is suitable. ● **JDK_11.0.15***—This entry indicates that any JDK of the 11.0.15 series is suitable. The + or * operator specifies a version range: ● The + operator means “at least this version.” ● The * operator means “of this version.” When you use these operators, any unspecified version part is assumed to be zero (specifying **`11.0`** is interpreted as 11.0.0_0). If you do not specify * or +, only versions that exactly match the specified version are valid (**`11.0`** does not validate for 11.0.1_07 VMs).
Strict VM Selection	Alternately, you can use a strict VM expression, such as JRE_11.0.15_03 or JDK 17.0.08_07, by joining a type (ALL, JDK, or JRE) and version number (17.0.08_07, 11.0*, and so on) with an underscore character. ● **JRE_11.0.15_03** allows the installer or application to run only against the JRE 11.0.15_03. ● **JDK_17.0.08_07** allows the installer or application to run only against a JDK 17.0.08_07.  ***Note ▪*** *If more than one of these expressions is present, consider them to be combined with the OR operator. That is, a VM is valid if it matches any of the given expressions.*

LaunchAnywhere

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

LaunchAnywhere enables end users to launch a Java application by double-clicking an icon (Windows and OS or 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 performing 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 JVM. Use the following syntax for 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 offers various features that enable you to customize your installers for global distribution.

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 that can be entered and customized in the InstallAnywhere Advanced Designer is *dynamic text*. Text that cannot be edited in the Advanced Designer is referred to as *static text*. Standard items such as file choosers as well as text for actions and panels that you are unlikely to want to change have static text. Dynamic text is written to the locale files (located next to the project file). Translations of static text is available in the following location:

IA_HOME\resource\i18nresources

Although it is unlikely that you would need to change static text, InstallAnywhere provides the ability to change 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 version of InstallAnywhere, 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 that are 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

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 time. They are 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 on the value of `$INSTALLER_LOCALE$`. So, for example, when an end user chooses German on your installer's splash screen, the installer uses the value of the key that your installer references from the German locale file in subsequent panels.

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 Sequences

Merge modules are able to access the external resource bundle of their parent installer during the Install and Uninstall sequences. This is also applicable to one-level nested merge modules and to multiple merge modules that are installed by a single parent.

Magic Folders

A magic folder represents a specific destination location, such as the end 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. InstallAnywhere offers predefined magic folders for standard locations on target system platforms that InstallAnywhere supports. InstallAnywhere also lets you define custom magic folders. A project can contain a maximum of 25 user-defined magic folders.



Note ▪ *For information on how magic folders are associated with variables, see Magic Folders and Variables.*

Merge Modules

Merge modules are essentially installer subprojects 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.



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 3-15 ▪ 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.
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 third-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 (select the Build Merge Module/Template option, which is available on the Distribution subtab on the Build Configurations tab in the Build Installers view) 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 3-16 ▪ Methods to Add Merge Module to an Existing Installer

Method	Description
Import Dynamic Merge Module	To merge a merge module into the current installer: On the Organization page, in the Modules view, click Import Dynamic Merge Module. • 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. 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.
Import Merge Module	To merge a static merge module into the current installer: On the Organization page, in the Modules view, click Import Merge Module. • 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	Merge modules can also be installed as self-contained sub-installer units, without merging them into the current project. 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 that are added in this manner are run as silent sub-installers.

Integrating Merge Modules Into Projects

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

Table 3-17 ▪ 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.

Table 3-17 ▪ Methods to Integrate Merge Modules Into Projects (cont.)

Method	Description
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 ▪ *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 fails depends on the amount of memory that is allocated to the build process. If the amount of memory that is allocated is very low, even small 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 3-18 ▪ Memory Allocation of Build Process

Property	Code

Heap Size Initial	`#  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`

Heap Size Maximum	`#  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`

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

You can apply InstallAnywhere rules 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 enables you 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 UNIX-based and Windows-based systems.

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 variables for source paths that exist in any project. The values of these variables cannot be changed or edited:

Table 3-19 ▪ Predefined Source Paths

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

SpeedFolders

Using SpeedFolders can dramatically increase the speed and memory efficiency of your installer. 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 (except items that have been marked to be filtered out) are added to the installer recursively. SpeedFolders are used to specify that a folder and all of its contents and subfolders 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. SpeedFolders cannot be converted into traditional directories, and traditional directories cannot be converted into SpeedFolders. To convert one type to the other, you must delete the one folder and replace it with the other type of folder manually.*

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 on an older one.

Uninstaller

InstallAnywhere automatically creates an uninstaller for each build target in a project. The uninstaller removes all files and actions that occur during the Install sequence of the installation. Actions that were added in other sequences of the installation cannot be removed using the uninstaller and should be accounted for in the Install sequence.

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 sequence are called. Lastly, Post-Uninstall actions are run.

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. One way is to include custom code actions, panels, or consoles in the Pre-Uninstall or Post-Uninstall sequences. The uninstall() method is called for custom code actions.

Another method of including custom code in the uninstaller is to schedule a custom code action in the Install sequence. In this case, the install() method of the action is called at install time and the uninstall() method of the same action is called at uninstall time.

About Uninstallers and Variables

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

Special Considerations for Apple OS or OS X–Based Uninstallers

When you build an installer for OS or OS X–based systems, the uninstaller is created at build time on the build machine instead of at run time on target systems so that it can be code-signed as part of builds. Thus, you can use build-time variables in your project if you are building OS or OS X–based installers. However, run-time variables are not resolved for the uninstaller. Thus, run-time variables are not supported for OS or OS X–based uninstallers.

Feature Uninstallation

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 end user is offered the option to uninstall only certain features.

A feature-level uninstallation 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 that 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 work. Each uninstaller is tied to a certain product. It does this through its product code (which is defined in the General Settings view on the Project page).

InstallAnywhere offers various options for developing an uninstaller for a project that installs multiple products (or uses merge modules).

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 code, 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 code.
- Each separate product should then be “demoted” to a feature of the parent 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 3-20 ▪ 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 • **Product Code**: 97338341-1ec9-11b2-90e2-a43171489d33 • **Features**: Acme Word Processor and Acme Spreadsheet
Installer 2	Acme Slide Show • **Product Code**: 97338341-1ec9-11b2-90e2-a43171489d33 • **Features**: Acme Slide Show
End Result	Acme Office Suite uninstaller for Acme Word Processor, Spreadsheet, and Slide Show.

Parent Uninstaller that Executes Sub-Uninstallers

You can choose to produce a parent uninstaller that executes the other sub-uninstallers. This option is most similar to concept of merge modules that are used in the installer.

You can use the Execute Uninstaller action to cause the parent uninstaller to execute other uninstallers during uninstallation. The advantage of this technique is that is lets you separate your uninstallation logic. If you use this technique, you may want to use the InstallAnywhere variables `$NEVER_UNINSTALLS_VM$` and `$REGISTER_UNINSTALLER_WINDOWS$`.

Uninstalling Merge Modules When Main Project is Uninstalled

For dynamic merge modules, you can specify that you want to implement a single uninstaller that uninstalls 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 (Organization page > Modules view).

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 sequence.

Uninstalling Merge Modules During Upgrades

If your project includes one or more merge modules as well as upgrade support, the installer can directly handle upgrades of the merge modules—the uninstallation of the earlier versions of merge modules as well as the installation of the new version. To learn more, see Special Considerations for Upgrade Support.

Similarities Between the Sequences in an Installer and an Uninstaller

The basic Install and Uninstall sequences follow the same organization and have the same options. The differences between the sequences are determined solely by the time of their occurrence in the installation or uninstallation processes.

Installer

- Pre-Install sequence
- File installation (determined by the Install sequence)
- Post-Install sequence

Uninstaller

- Pre-Uninstall sequence
- File uninstallation (determined by the Install sequence)
- Post-Uninstall sequence

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

Use these sequences to add the Panel, Console, and General actions that are needed by your installer and uninstaller. Some General actions (such as Modify Text File) that are used in these sequences trigger behavior at both install time and uninstall time if they are scheduled in the Install sequence. The uninstall functionality (reverting the changes that were made) is not activated if the action is not in the Install sequence.

Advanced Designer Interface

Because these sequences are so similar, the options that are displayed in the Advanced Designer for the Pre-Install, Install, Post-Install, Pre-Uninstall, Uninstall, and Post-Uninstall sequence panels are also very similar to each other. For more information, see:

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

Variables

During installation, InstallAnywhere-generated installers keeps track of dynamic values through the use of variables. Almost every dynamic value in InstallAnywhere, such as the path to which a magic folder refers, 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-generated installation. They enable you to control the flow of information and the flow of the installation. Variables let you store information from the system and information input by end users, and create rules to determine operations based on that information. You can even have your installer write that information to configuration files or other resources for the application.

Variable Notation

Variables are referenced by a variable name. For instance, `USER_INSTALL_DIR` is the variable that specifies the folder to which the end user has selected to install files. You can set `USER_INSTALL_DIR` if you 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 dollar sign (`$`) notation means "substitute the value of the variable here." For simplicity in this document, the dollar sign 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 changes the destination to which the magic folders will be installed. Changing the value of the `$USER_INSTALL_DIR$` through InstallAnywhere changes where the files will be installed.

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 view on the Project page 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 in the Platforms view on the Project page 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: $JAVA_HOME$**—This exception has two variations:
  - **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 that is specified on the General subtab of the Search Panel Settings tab (Project page > JVM Settings view). It can change when the `$USER_INSTALL_DIR$` location 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 the Evaluate at Assignment check box selected, you cannot set* `USER_MAGIC_FOLDER_1 = U$SER_MAGIC_FOLDER_1$/$test` *to append* `/test` *to* `USER_MAGIC_FOLDER_1`. *InstallAnywhere allows direct and indirect recursion only 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 3-21 ▪ 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-based and UNIX-based systems. These values are stored in the variable set `$lax.nl.env.VARIABLE_NAME$` (For example, on UNIX-based systems, `$lax.nl.env.PATH$` 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 `$INSTALL_SUCCESS$`).  ***Important*** ▪ *Run-time variables are not supported for the uninstaller on OS or OS X–based systems. For more information, see About Uninstallers and Variables.*
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 be evaluated at assignment. Hence, these variables keep the value they have when they are set.

Using the Evaluate at Assignment option 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 3-22 ▪ 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 Texts

You can search a project to locate all references to a specific text. You can choose to search for a text in any of the installation sequences (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.

Performing a Search for a Text



Task ***To search your project for all references to a specific text:***

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. In the **Enter Text** setting, enter the name of the text you want to search for.
4. If you want to search for a variable, select the **Variable Search** check box.

   Note ▪ *If the Variable Search check box is checked, it will enclose with a dollar sign symbol (**$**) before and after the variable.*
5. If you want to search for a string and variable, unselect the **Variable Search** check box.

   Note ▪ *If the Variable Search check box is unchecked, it will not be enclosed with a dollar sign symbol (**$**) before and after the text.*
6. If you want to perform a case-insensitive search, select the **Ignore Case** check box.
7. In **Search for text in** area, select the installation phases (**Pre-Install**, **Install**, **Post-Install**, **Pre-Uninstall**, **Uninstall**, and/or **Post-Uninstall**) and product elements (**Install Set**, **Features**, and/or **Components**) that you want to search.
8. Click the **Search** button.

InstallAnywhere shows the results of the search in a tree structure with the total number of texts that were found listed at the top. Items that are associated with the selected variable are listed below, and grouped by category.

Performing a Search and Replace of a Text



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

1. Search for a text, as described in the afore mentioned instructions.

   Note ▪ *Enter the full text in the **Enter Text** setting, and do not select the **Partial Search** check box.*

   InstallAnywhere shows the results of the search.
2. To replace all instances of the found text with a replacement value, click the **Replace All** button. The **Replace Text** dialog box opens.
3. In the **Replace Text with** setting, enter the replacement value.



Note ▪ *It is mandatory that you always enter the dollar sign symbol (**$**) before and after the replacement text. For example, instead of searching for* **$NAME$** *and replacing it with* **NEWNAME**, *you should replace it with* **$NEWNAME$**.

4. Click the **Replace All** button.

InstallAnywhere replaces all instances of the text with the replacement text.

About Replacing Magic Folder Variables

If you choose to replace a magic folder variable with another variable, InstallAnywhere simply copies 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.

The Choose Variable dialog box lists all of the variables that are 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

4

InstallAnywhere Tutorial

The following tutorial teaches you 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
- Adding Pre-Install Actions
- Defining the Install Sequence
- Adding a LaunchAnywhere Executable to the Install Sequence
- Adding Post-Install Actions
- Building the Installer
- Testing the Installer

Creating a New Project

The first step is to create a new project and open it in the InstallAnywhere interface.



Task ***To create a new project:***

1. Launch InstallAnywhere. The InstallAnywhere **Create/Open Project** dialog box opens.
2. Click the **Create new project** button. The **Create a New Project** dialog box opens.
3. Select the **Basic Project Template**. This template should already be selected by default.
4. Enter a **Project Name** and **Project Path**. By default, the project is named `My_Product.iap_xml`.
5. Click **OK**. The newly created project file is open in InstallAnywhere on the **Project > General Settings** view.

In the **General Settings** view, you set basic installer options such as the name of the product, the installer title, and the installer name. InstallAnywhere uses the installer name for the name of the executable file that it creates for your installer. This view also sets the location to build the installer and the settings for the generation of installation logs.

6. In the **Product Name** setting, enter `OfficeSuitePro`.
7. In the **Installer Title** setting, enter `OSPro`.
8. On the **Sequence** page, click **Pre-Install**. The **Pre-Install** view 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 your project. 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** view opens.

   The **Pre-Install** view is where you schedule 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 introduces 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 OS or OS X aliases, Windows shortcuts, and UNIX symbolic links (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.

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

   - **Order of actions**—Actions in the **Pre-Install** view occur in the order in which they are listed in this view, from top to bottom. To modify the order of panels and actions, use the arrow buttons in the middle right of the Advanced Designer.
   - **Customizers**—To modify the behavior and content of a panel, select it. The area at the bottom of the view displays the settings for the selected panel. In InstallAnywhere's vocabulary, this is known as a customizer, and is available for each action and panel in the installer.

2. In the **Pre-Install Action List**, select **Panel: Choose Alias, Link, Shortcut Folder** and then click the **Add Action** button. The **Choose an Action** dialog box opens.
3. Click the **Panels** tab.

4. Click the **Panel: Choose Java VM** action and then click the **Add** button. InstallAnywhere adds the new action to the **Pre-Install Action** list and displays **Choose Java VM** customizer at the bottom of the view.
5. Click the **Close** button. The **Choose an Action** dialog box closes.
6. Click the link for the **Install** view and proceed with the steps in Defining the Install Sequence.

Defining the Install Sequence

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



Task ***To define the installation sequence:***

1. Perform the steps in Adding Pre-Install Actions. The **Install** view on the **Sequence** page opens. The **Install** view 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** view has a folder named `_$PRODUCT_NAME$_installation`, which contains any InstallAnywhere uninstaller/maintenance mode actions, and a comment action with instructions that pertain to the uninstaller.

   Actions (including, but not limited to, the installation of files) in the **Install** view occur in order with actions at the top of the installation occurring first.

2. 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 views. In the* ***Install*** *view, 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 the **Add Files** button. The **Add Files To Project** dialog box opens.
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** view.

   You can expand or contract file trees within the **Install** view by clicking the “+” or “-” boxes at the apex of the tree branches. You can move objects 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. Proceed with the instructions in Adding a LaunchAnywhere Executable to the Install Sequence.

Adding a LaunchAnywhere Executable to the Install Sequence

A LaunchAnywhere executable (LAX) is a unique built-in executable file that InstallAnywhere creates. It is used to launch a Java application. You can add as many launchers as you would like. There are two ways to add a LaunchAnywhere launcher to an InstallAnywhere project.



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

1. Perform the steps in Defining the Install Sequence. The **Install** view on the **Sequence** page opens and the `OfficeSuite2000` files and folders are listed in the **Visual Tree**.
2. In the **Visual Tree**, select the **User Install Folder**, and click the **Add Launcher** button. InstallAnywhere displays a message box asking if you would like 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**.

   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 sequence.
5. To customize the appearance of the launcher's shortcut, select the **OfficeSuite** launcher. The customizer in the bottom of the view changes 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 that is associated with the launcher. By default, a 32x32 pixel coffee cup icon and a 16x16 pixel rocket ship icon are chosen.
6. Click the **Change** button. The **Choose Icon** dialog box opens.
7. On the **Windows** tab next to the **32x32 Icon**, click **Choose GIF File**.
8. 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 OS or OS X, provide an ICNS file (created with*

iconbuilder—part of the OS or OS X Developer Tools). InstallAnywhere maintains a general classpath that is used to create launchers for the Java Application.

9. For the **16x16 Icon**, select `officeIconSmall.gif` from the same directory.
10. Click **OK**. InstallAnywhere displays the new icons in the **Create LaunchAnywhere for Java Application** customizer.
11. Click the **Set Classpath** button. InstallAnywhere displays a message box asking if you would like to automatically set the classpath.
12. Click **OK**. InstallAnywhere adds a blue CP icon on folders and archives that the process has added to the classpath.
13. To view the classpath as determined by the Set Classpath action: On the **Project** page, click **JVM Settings**, and then click the **General Settings** tab.

    Note the following regarding the **General Settings** tab:

    - 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**, you can remove it from within the **JVM Settings** view. As an alternative, you can select that file in the Visual Tree in the **Install** view and clear 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 view specifies actions and panels to occur after the installation of files. Like the Pre-Install sequence, the Post-Install sequence is ordered with the top actions occurring first. By default, InstallAnywhere has added two actions to the InstallAnywhere project:

- **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-based system if the installer determines that it is necessary. Also, this action will be included as a dependency of the Uninstall Complete Panel's prompt to restart the machine.

InstallAnywhere-generated installations are controlled primarily by InstallAnywhere rules. As an example of an InstallAnywhere rule, select the Restart Windows action in the Office Suite project. In the customizer at the bottom of the view, select the Rules tab. The Rules customizer opens.

The rules that are set for the Restart Windows action are 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 setting. If you want to write complex rule expressions, you can edit the expression in the Rule Expression setting 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 Sequence.
2. On the Sequence page, click **Post-Install**. The **Post-Install** view opens.
3. Click the **Add Action** button. The **Choose an Action** dialog box opens.
4. Click the **General** tab.
5. Click the **Execute Target File** action.

   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** sequence of the installation. The **Execute Target File** action is not available in the **Pre-Install** view because files cannot be executed if are not installed.
6. Click the **Add** button. InstallAnywhere adds the **Execute Target File** action to the **Post-Install Action List** and displays its customizer at the bottom of the view.
7. If the Execute action panel was added at a location other than the bottom of the **Post-Install** view, move it now. Either use the up and down arrows, or drag the action to the bottom of the action list.
8. In **Execute Target File** customizer at the bottom of the view, click the **Properties** tab.
9. In the **Name** setting, enter a name for the action. Naming the action will help you identify the action when you looking at the visual tree.
10. To select the target, click the **Choose Target** button next to the **Target** setting. The **Choose an Action** dialog box opens and displays the file installation tree that is specified in the **Install** view.
11. You can add one or more files that you want to be launched 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**) and click **OK**.

    You need to choose the actual OfficeSuite launcher, and not the shortcut, because shortcuts—especially on Windows and OS or OS X systems—are pointers and are not inherently executable. InstallAnywhere-generated installers will not execute a shortcut.

    Note ▪ *By using the **Command Line** setting, modifications can be made to the command line that the installer uses 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, add an executable path to the begging of the entry; to specify an argument, append a file path. These paths must be absolute; however, the paths can include InstallAnywhere variables.*
12. Continue with the steps in Building the Installer.

Note that to customize the user experience for the Execute Target File action, you can select **Suspend the installation until the process completes** check box 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 suboption, the **Show indeterminate dialog** check box, to specify an indeterminate progress bar with a message. You can use this option if the execution may take some time (for example, launching a file that installs another product, or configures a database or other application).

Building the Installer

The Build Installers view on the Build page contains the settings that InstallAnywhere uses to build the installers. This view is also where you specify information such as which platforms your installers are targeting, 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 that is 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. On the Build page, click **Build Installers**. The **Build Installers** view opens.
3. In the **Select Build Configuration** list, select **Default Configuration**.

   Each InstallAnywhere project can have multiple build configurations, each representing how InstallAnywhere builds the installer for a 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 Working with Build Configurations.
4. On the **Build Targets** tab, click the plus sign next to **Windows**. InstallAnywhere displays the **Windows** platform customizer.
5. Select the **Without VM** check box, and under it select the **Search for VM; if not found exit** check box.
6. Select the **With VM** check box, 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 selectable only for platforms that have a VM pack.
   - VM packs should be placed in the `[InstallAnywhere]/resource/installer_vms` folder, and then InstallAnywhere should be restarted to refresh the available VM packs.

   The **Build Configurations** tab in the **Build Installers** view also includes three additional subtabs:

   - **Distribution**—Configure options for the types of installers to build, and specify the optimization options for each installer. If the installer 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 Installers** view displays the XML log of previous builds.*

7. Click **Build Project**. The **Building** information dialog opens displays the progress of the build.

   When the build is complete, the dialog box displays the message `Build successful`.

8. Click **OK**.
9. Proceed with the steps in Testing the Installer

Testing the Installer

After the build process is complete, try the installer.



Task ***To test the installer:***

1. Perform the steps in Building the Installer.
2. In the **Build Installers** view, click the **Try Web Install** button. InstallAnywhere launches the machine's default Internet browser and opens the Web install page. The browser should request security access.
3. Grant access to allow the Web Install Applet to run the installer.
4. Under **includes Java VM**, click the **Download** link. The download begins.
5. When the download is complete, run the installer, accepting the default values.
6. On the **Install Complete** panel, click **Done**. The installer close and OfficeSuite opens.



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

5

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
Working with InstallAnywhere Projects	Includes background information on the file for an InstallAnywhere project; also explains how to create new projects and open existing projects.
Specifying Installer Information	Describes how to define basic information for your installer, including how to include a software identification tag for your product.
Organizing Files for Your Installer	Describes various tasks and best practices for managing files, components, features and other items in your InstallAnywhere projects. Also discusses how to work with source paths instead of hard-coded paths, which is especially helpful in team development environments.
Customizing Sequences	Explains how to schedule the panels that are displayed and the actions that are launched at run time.
Defining Rules and Rule Expressions that Evaluate Conditions on Target Systems	Describes how to define rules for InstallAnywhere installers, specific actions, and groups of actions.
Configuring Servers	Describes how to manage SQL databases and deploy Web applications to application servers.
Preparing Your Installer for 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.

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

Section	Description
Configuring Installation Rollback Behavior	Explains how to enable rollback behavior to avoid application corruption if an end user cancels an installation before it is completed or if an installation encounters a fatal error.
Building and Testing Installers	Explains how to build and test installers.
Working with JRE 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 that your installer deploys.
Launching Additional Installers	Explains how to launch another installer from your installer.
Working with Source Code Control Software	Describes the use of source code control software with InstallAnywhere.

Working with InstallAnywhere Projects

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 they 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 for working on the XML project file are available in the InstallAnywhere application folder, inside the XML Project File Tools directory.

Creating a New Project

To create a new project, perform the following steps.



Task ***To create a new project:***

1. Launch InstallAnywhere. The InstallAnywhere **Create/Open Project** dialog box opens.
2. Click the **Create new project** button. The **Create a New Project** dialog box opens.
3. Select a template to use.
4. Enter a **Project Name** and **Project Path**.
5. Click **OK**. The project opens in the InstallAnywhere interface.

Opening an Existing Project

To open an existing project, perform the following steps.



Task ***To open an existing project:***

1. Launch InstallAnywhere. The InstallAnywhere **Create/Open Project** dialog box opens.
2. Click the **Open existing project** button. The **Open Project File** dialog box opens.
3. Locate and select the project file that you want to open.
4. Click the **Open** button. The project is opened in the InstallAnywhere interface.

Specifying Installer Information

When you create your installer project, you need to configure basic information about your project and your product. This includes details such as the name and version number of your product, and whether you want run-time log files to be created for troubleshooting issues with installers.

Configuring General Information

When you create your installer project, you need to configure basic information about your project and your product. This includes details such as the name and version number of your product.



Task ***To configure general information about your project and product:***

1. In the Advanced Designer, on the **Project** page, click **General Settings**. The **General Settings** view opens.
2. Configure each of the settings as needed.

For detailed information about each of the project and projects settings that are available, see General Settings View.

Specifying the Product Version

When you are specifying the version number for your product, you must ensure that you enter a valid product version. The version must contain only numbers. It is typically in the format *a*.*b*.*c*.*d*, where *a* represents the major version number, *b* represents the minor version number, *c* represents the revision number, and *d* represents the subrevision number.



Task ***To specify the version number of your product:***

1. In the Advanced Designer, on the **Project** page, click **General Settings**. The **General Settings** view opens.
2. In the **Product Information** area, in the `Version` setting, enter the appropriate version number for your product.

Configuring Run-Time Logging Preferences

InstallAnywhere includes support for generating run-time log files that you or your end users can use to troubleshoot issues with installers. Your installers can generate log files during installation, during uninstallation, when features are added, when the product is repaired, or when features are removed. By default, installers do not generate log files.



Task ***To enable logging and configure logging preferences:***

1. In the Advanced Designer, on the **Project** page, click **General Settings**. The **General Settings** view opens.
2. In the **Log Settings** area of the view, modify the values of the settings as needed.

For information about all of the log-related settings that you can configure, see General Settings View.

Log File Locations

The default target system path for log files is:

```
$USER_INSTALL_DIR$$\$_$PRODUCT_NAME$_installation$\$Logs$/$
```

When you are specifying the path for a log file, you can use an absolute path. If the path is invalid at run time, the installer saves the log file in the default location.

If you edit one or more log file paths and then want to revert back to the default values for all of the enabled log files, click the Restore Default Paths button in the Log Settings setting.

The default log location is resolved by the InstallAnywhere variable *$INSTALL_LOG_DESTINATION$*.

The default file name for the install log is:

```
$PRODUCT_NAME$_<Mode>_<MM_DD_YYYY_HH_MM_SS>.log
```

For example:

- My_Product_Install_08_13_2010_11_48_46.log
- My_Product_Add_08_13_2010_11_48_46.log

Logging Installers that Include Merge Modules

For this merged logging functionality 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.

To append the merge module logs to the parent log, you must select **Plain text format** in the Log Format setting, which is available in the General Properties view of the Project page. Adding the merge module log to the parent log is not available for the XML log format.

Including a Software Identification Tag for Your Product

ISO/IEC 19770-2 is an international standard for the creation of software identification tags. A software identification tag is a small, XML-based file that contains descriptive information about the software, such as the product name, product edition, product version, and publisher. Software asset management tools collect the data in the tags to provide accurate application identification for software that is installed in an enterprise.

Software identification tagging is evolving as an industry standard, enabling independent software vendors to create smarter applications that give their customers better information for software asset management and license optimization initiatives. Including the identification tag in your product's installation makes it possible for your customers to use tools that can monitor their internal usage of your product, allowing them to understand, manage, and optimize the number of licenses of your product that they obtain from you.

Proper tag creation requires that you configure a few identification-specific settings in the General Settings view on the Project page.



Task ***To enable and configure software tagging:***

1. On the **Project** page, click **General Settings**. The **General Settings** view opens.
2. In the **Software Tag** area of the view, modify the values of the settings as needed.

   The **Enable Software Tagging** check box lets you specify whether you want to include a tag in your installation. Select this check box, and then configure the other settings in the **Software Tag** area as needed.

For information about all of the settings that you can configure for software identification tag, see General Settings View.

Build-Time Support for Software Tagging

At build time, if the following conditions are true, InstallAnywhere includes the software identification tag with the installers that it builds:

- The Enable Software Tagging check box in the Software Tag setting is selected.
- The Product ID, Unique ID, Tag Creator Name, and Tag Creator ID settings have valid values.

Note that if tagging is enabled but you have not entered valid values in one or more of the aforementioned tag identification settings, InstallAnywhere generates a build warning to inform you that the tag could not be included in your installers. To resolve this warning, configure the tag-related settings in the General Settings view as needed.

Location of Software Tag Files on Different Platforms

The following table describes the location where installers place the software identification tag at run time on target systems.

Table 5-2 ▪ Examples of Tag Locations on Different Platforms

Operating System	Location
Apple OS or OS X	`<root>/Library/Application Support/<software creator regid>`
Apple intosh OS X Pre-Leopard	`Application Directory/<program.app package>/contents`
UNIX and Linux	`usr/share/<software creator regid>`
Microsoft Windows XP **Microsoft Windows Server 2003**	`%AllUsersProfile%\Application Data\<software creator regid>`

Table 5-2 ▪ Examples of Tag Locations on Different Platforms (cont.)

Operating System	Location
Microsoft Windows Vista and later	`%Program Data%\<software creator regid>`
Microsoft Server 2008 and later	

Organizing Files for Your Installer

Components are the lowest level of organization in an installer. Each product must have at least one component; most installers contain at least two components, since 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.

Best Practices for Components

When you are planning what components you want to include in your project, consider the following best practice recommendations:

- **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; this enables you to define a unique User Guide component and to version and update it individually as needed.
- **Components should make logical sense.** If you build a suite installer, keep track of 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 Advanced Designer > Organization page > Components view) when the component is added to the project.
- **Components with similar UUIDs**—If the UUIDs of the components are shared across installers or along with merge modules and installed in the same `USER_INSTALL_DIR`, that component is treated as a shared component. The uninstaller for the last product that references this component will uninstall it.

Adding Components

To add components to a project, use the Components view on the Organization page. This view let you add and remove components, configure component settings, 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 a project:***

1. In the Advanced Designer, on the **Organization** page, click **Components**. The **Components** view opens.
2. Click the **Add Component** button.

InstallAnywhere adds a new, untitled component to the Component List. Configure the component's settings as needed. For more information, see Components View.

Adding Folders to a Project

To add folders that you want your installer to create on target systems, add the folders to the Install sequence of your project.



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

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. In the **Visual Tree**, select the folder that you want to contain the folder that you are adding.
3. Click the **Add Action** button. The **Choose an Action** dialog box opens.
4. On the **Install** tab, click the **Create Folder** action, and then click the **Add** button. InstallAnywhere adds the action to the **Visual Tree**. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
5. To move the action up or down in the sequence, use the up or down arrow keys.

   To move the folder to be a parent folder or a subfolder, use the left or right arrow keys.

To specify the name of the folder and configure other settings about the folder's action, select the action and use its customizer at the bottom of the view. For detailed information on each of the settings, see Create Folder Action.

Adding Files to a Project

To add files that you want your installer to transfer to target systems, add the files to the Install sequence of your project.



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

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. Click the **Add Files** button. The **Add Files to Project** dialog box opens.
3. Browse to directory that contains the files you want to add.
4. Click 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.

5. Click **Done**.

The selected files and folders are now listed in the Visual Tree.

Select each new item in the Visual Tree and configure its settings on its customizer, which is displayed at the bottom of the view. For information on each of the settings, see Install File Action.

Assigning Files to Components

To assign files to a component, use the Install view on the Sequence page.



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

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. In the **Assign to** list, select **Components**. All of the project components are listed.
3. In the **Visual Tree**, find the file that you want to assign to the component. In the same row, select the option that corresponds with the component to which you want to assign the selected file.



Tip ▪ *To learn how to add a file to your project, see Adding Files to a 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 in your project:***

1. In the Advanced Designer, on the **Organization** page, click **Components**. The **Components** view opens.
2. Click the **Clean Components** button. InstallAnywhere prompts you to confirm that you want to delete all of the components that have no actions assigned to them.
3. Click **OK**.

InstallAnywhere removes all of the empty components from your project.

Integrating Components that Are Already Installed on Target Systems

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

When you are configuring a Find Component in InstallAnywhere Registry action in your project, you can optionally request that only the latest version be found. You can also indicate whether you want the installer to compare versions, or to search for the key file. At run time, the action sorts the count of components, the versions, and the locations that it finds in variables that you specify.

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

Use the Features view on the Organization page to manage the features in your project. You can add and remove features, as well as assign features to install sets.



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

1. In the Advanced Designer, on the **Organization** page, click **Features**. The **Features** view opens.
2. Click the **Add Product Feature** button.

InstallAnywhere adds a new, untitled feature to the Feature Tree. Configure the feature's settings as needed. For more information, see Features View.

Assigning Files to Features

To assign a file to a feature, use the Install view on the Sequence page.



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

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. In the **Assign to** list, select **Product Features**. All of the project features are listed.
3. In the **Visual Tree**, find the file that you want to assign to the feature. In the same row, select the option that corresponds with the feature to which you want to assign the selected file. A file can belong to more than one feature.

Assigning Components to Features

You can manually assign components to features in the Components view on the Organization page.



Note ▪ *If you are manually managing the components in your project, avoid assigning files directly to features, since this creates and changes component assignments. Instead, ensure that you assign files to components yourself.*



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

1. In the Advanced Designer, on the **Organization** page, click **Components**. The **Components** view opens.
2. In the **Product Features** column, find the component that you want to assign to the feature. In the same row, select the option that corresponds with the feature to which you want to assign the selected component.



Tip ▪ *You can also assign files to features directly, which creates the feature-to-component assignments automatically. To learn more, see Assigning Files to Features.*

If you assign a file to a feature that does not already have a component, InstallAnywhere automatically creates a component for the file.

Installing Fonts

InstallAnywhere handles font installation with the help of the $FONTS$ magic folder.



Task ***To configure the installation of fonts on a target system:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. Click the **Add Files** button. This **Add Files to Project** dialog box opens.
3. Navigate to the font file you want to install, select it, and click **Add**. The font file that you selected appears in the **Files to Add** list.
4. Click **Done**. InstallAnywhere adds the file to the **Visual Tree** and displays its settings in the **Properties Customizer** at the bottom of the view.
5. Under **Destination**, in the **Path** list, change the value from **User Install Folder** (the default) to **Fonts Directory**.

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.

Expanding Archive Files on the Target System

To expand an archive file on target systems, use one of the Expand Archive actions. The following archive file types are supported:

- ZIP files (.zip)
- Java archives (.jar)
- Binary files (.bin)
- StuffIt files (.sit)
- WAR files (.war)
- Enterprise archives (.ear)

- 7-Zip archives (.7z or .xz) with the LZMA and LZMA2 compression methods.
- TAR files (.tar, .gz, and .Z)

The archive files can be installed as part of your installer, or they can be present on target systems.



Task ***To expand an archive file on target systems:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. In the **Visual Tree**, select the destination folder for the archive that you are expanding.
3. Click the **Add Action** button. The **Choose an Action** dialog box opens.
4. On the **Install** tab, click one of the following actions:
   - **Expand Archive**—This action lets you expand any of the following types of archives: .zip, .jar, .bin, .sit, .war, or .ear.
   - **Expand Archive (7-zip)**—This action lets you expand 7-Zip archives (.7z or .xz) with the LZMA and LZMA2 compression methods.
   - **Expand Archive (TAR)**—This action lets you expand .tar, .gz, and .Z archives.
5. Click the **Add** button. InstallAnywhere adds the action to the **Visual Tree**. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
6. To move the action up or down in the sequence, use the up or down arrow keys.

   To move the archive file to be in a parent folder or a subfolder, use the left or right arrow keys.

To configure the archive's settings, select the action and use its customizer at the bottom of the view.

About File Permissions of the Extracted Files

The permissions that the installer assigns to files that are extracted from an archive file at run time vary, depending on the type of expand archive action. In some cases, the original permissions of the file are retained. In others, the settings that are made in the Default Permissions setting (Project page > Platforms view > UNIX or OS X areas) are applied to the extracted files. The following table indicates how permissions are applied for these three expand archive actions.

Table 5-3 ▪ Expand Archive Actions and Extracted File Permissions Comparison

Action	Are Original File Permissions Preserved?	Apply Permissions that Are Set in the Platforms View?
Expand Archive Action	Not guaranteed to be preserved	Yes
Expand Archive (7-zip) Action	No	Not guaranteed to be applied
Expand Archive (TAR) Action	Yes	No

Working with Source Paths

Working with source paths enables you use variable paths instead of absolute paths when you are referencing file resources in your projects. This enables 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.

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. On the **Edit** menu, click **Preferences**. The **InstallAnywhere Preferences** dialog box opens.
2. Click the **Source Paths** tab.
3. Select **Enable Source Paths** check box.

Adding and Removing Source Paths

InstallAnywhere supports two different methods for adding source paths to a machine.

Adding Source Paths Using the InstallAnywhere Preferences Dialog Box

To add source paths using the InstallAnywhere **Preferences** dialog box, perform the following steps.



Task ***To add source paths using the InstallAnywhere Preferences dialog box:***

1. On the **Edit** menu, click **Preferences**. The **InstallAnywhere Preferences** dialog box opens.
2. Click the **Source Paths** tab.
3. Select **Enable Source Paths** check box.
4. Click the **Add** button. InstallAnywhere adds an empty row to the list of source paths.
5. In the **Access Path Name** column, click the empty box and then enter a variable name for the source path. For example:

   ```
   RESOURCE
   ```

6. In the **Folder** column, click the empty box. The **Choose Folder** dialog box opens.
7. Browse to the source path and click the **Select** button. InstallAnywhere adds the full path to the **Folder** box.
8. 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 source paths 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-based systems, open the **System Properties** dialog box. On the Advanced tab, click the **Environment Variables** button.
   - On UNIX-based or OS or OS X–based systems, modify the proper shell configuration file or set the variable directly using the shell.
2. Add an environment variable for the source path, preceded by the string IA_PATH_. For example, to set the source path SOURCE_PATH, set the environment variable as follows:

   ```
   IA_PATH_SOURCE_PATH
   ```

Removing Source Paths

To remove source paths, perform the following steps.



Task ***To remove a customized source path:***

1. On the **Edit** menu, click **Preferences**. The **InstallAnywhere Preferences** dialog box opens.
2. Click the **Source Paths** tab.
3. In the list of source paths, click the one that contains the source path that you want to delete.
4. Click the **Remove** button.
5. 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.

Customizing Sequences

Customizing the sequences of an installer and its uninstaller is an important part of developing an installer. Sequences specify the order in which panels are displayed and actions are executed at run time.

To learn more, see:

- Customizing the Pre-Install Sequence
- Customizing the Install Sequence
- Customizing the Post-Install Sequence
- Customizing the Uninstall Sequence

Customizing the Pre-Install Sequence

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

- **Introduction**—This panel introduces the product or installation process.
- **Choose Install Folder**—This panel enables end users to choose the installation location for the product.
- **Choose Alias, Link Shortcut Folder**—This panel lets end users specify the location for any OS or OS X aliases, Windows shortcuts, and UNIX symbolic links (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.



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

1. In the Advanced Designer, on the **Sequence** page, click **Pre-Install**. The **Pre-Install** view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. Click the action you want to add, and then click the **Add** button. InstallAnywhere adds the action to the **Pre-Install Action List**. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. Use the arrow keys to move the action up or down in the action list.

To specify an action's settings, select the action and use its customizers at the bottom of the view. For detailed information on each action's customizer settings, see Actions.

Customizing the Install Sequence

The Install sequence contains the installation actions that occur when the installer transfers files to target systems. In the Install view on the Sequence page, you can 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 the installer deploys files to the target system.

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

By default, the following items are available in the Install sequence of all new InstallAnywhere projects:

- **_$PRODUCT_NAME$_installation**—By default, this folder contains any InstallAnywhere uninstaller and maintenance mode actions, and a comment action with instructions that pertain 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 Install sequence (although the folder structure can be changed).
- **Shortcuts' Destination Folder ($USER_SHORTCUTS$)**—This magic folder is the directory that is specified by the end user as the Alias, Link, Shortcut folder location. If you include the Choose Alias, Link, Shortcut Folder panel action in the Pre-Install sequence of the installer, end users can change this location.

You can see these items in the Visual Tree that is displayed in the Install view on the Sequence page of your project.



Important ▪ *By default, the Uninstall Launcher, Change $PRODUCT_NAME$ Installation, is included in the _$PRODUCT_NAME$_installation folder of the Install sequence. Ensure that the Uninstall Launcher is associated with the InstallAnywhere Uninstall component. (That is, when Components is selected in the **Assign to** list of the Install view, the Change $PRODUCT_NAME$ Installation action in the $PRODUCT_NAME$_installation folder must be associated with the Uninstall component.) If the Uninstall Launcher is not associated with this component (which is automatically created and is listed in the Components page (Advanced Designer > Organization > Components), the project build fails with a message indicating that is necessary to associate the Uninstall Launcher to the Uninstaller component.*

Overview on Defining the Install Sequence

Defining the Install sequence of a project involves adding files, components, features, and actions to your project.



Task ***To define the Install sequence:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. Add files to the project, as described in Adding Files to a Project.
3. Add actions to the **Install** sequence, as described in Adding Actions to the Install Sequence.

   Tip ▪ *The Advanced Designer provides a drag-and-drop interface in many views and on many tabs. In the **Install** view, you can move actions and files by selecting and dragging them. A dark underline appears in the location where the file or action will be placed.*
4. Organize features and components, as described in Organizing Files for Your Installer.

Adding Actions to the Install Sequence

You can add actions directly into the file and folder hierarchy that is shown in the Install view on the Sequence page of your project. Actions are sequence specific, so the actions that are available for the Install sequence are different than those that are available for other sequences. The actions that are available in the Install sequence create folders and aliases, copy or delete files and folders, expand folders, or add functionality—like creating LaunchAnywhere launchers or adding platform-specific actions.



Task ***To add actions to the Install sequence:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. Click the action you want to add, and then click the **Add** button. InstallAnywhere adds the action to the **Visual Tree**. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. Use the arrow keys to move the action up or down in the action list.

To specify an action's settings, select the action and use its customizers at the bottom of the view. For detailed information on each action's customizer settings, see Actions.

Adding LaunchAnywhere Executable Files to the Install Sequence

A LaunchAnywhere executable file (LAX) is a unique built-in executable file that InstallAnywhere creates to launch a Java application. You can add as many launchers as you would like.



Task ***To add a LaunchAnywhere executable file to your project:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. In the **Visual Tree**, select the **User Install Folder** item.
3. Click the **Add Launcher** button. InstallAnywhere displays a message box asking if you want InstallAnywhere to automatically find classes with main methods.
4. Click **OK**. The **Choose a main class** dialog box opens with a class listed.

   Note ▪ *When you add a launcher, InstallAnywhere automatically inspects the added files (including introspecting into JAR and ZIP files) to find class files with main methods specified.*
5. Select one of the listed items as the main class for the application and click **OK**.

   InstallAnywhere adds the launcher to the Visual Tree, and also creates a Shortcut, Link, or Alias action in the Shortcuts' Destination Folder magic folder. The location varies, depending on what is specified for the Choose Alias, Link, and Shortcut panel in the Pre-Install view.
6. Click the **Set Classpath** button. A message box opens, asking whether you want to automatically set the classpath.
7. Click **OK**.

InstallAnywhere adds a blue CP icon on folders and archives that the process has added to the classpath.



Note ▪ *To view the classpath as determined by the Set Classpath action: On the Project page, click JVM Settings. On the General Settings tab, review the classpath settings.*



Tip ▪ *To customize the appearance of the shortcut for the launcher, select the Create LaunchAnywhere for Java Application action in the Visual Tree. The customizer in the bottom area of the view changes to reflect the options for the launcher. Configure its settings as needed.*

Re-creating the InstallAnywhere Uninstaller and Associated Components

When you create a new InstallAnywhere project, it contains an InstallAnywhere Uninstall Component by default. This component is available in the Components view of the Organization page.

By default, the component's name is InstallAnywhere Uninstall Component and its short name is Uninstall_ZG-IA; these names are not editable in the Components view.

If you inadvertently delete the InstallAnywhere Uninstall Component and want to re-create it, you can do so.



Task ***To re-create the InstallAnywhere Uninstall Component:***

1. If one an Create Uninstaller action is not available in the project, add one:
   a. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
   b. In the **Visual Tree**, add a **Create Uninstaller** action to the project, if one is not already present.
2. Add the InstallAnywhere Uninstall component:
   a. In the Advanced Designer, on the **Organization** page, click **Components**. The **Components** view opens.
   b. Click the **Add Component** button. InstallAnywhere adds a new, untitled component to the **Component List**.
   c. In the **Component** customizer at the bottom of the screen, in the **Name** setting, enter the following value:

      `InstallAnywhere Uninstall Component`

      InstallAnywhere adds the name `Install` automatically in the **Short Name** setting and changes this setting to read-only.
3. Configure the Create Uninstaller action:
   a. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
   b. In the **Assign to** list, select **Components**.
   c. In the **Visual Tree**, select the **Change $PRODUCT_NAME$ Installation** item, and then select the **Install** option in the **Components** list.
4. Configure the InstallAnywhere Uninstall component:

a. In the Advanced Designer, on the **Organization** page, click **Components**. The **Components** view opens.

b. In the **Component List**, select the **InstallAnywhere Uninstall Component**.

c. In the **Component** customizer, click the **Choose Key File** button. The **Choose a Key File** dialog box opens.

d. Select the location of the uninstaller: _$PRODUCT_NAME$_installation.

5. Save and close the project, and exit InstallAnywhere.

6. Edit the **shortName** property of the InstallAnywhere Uninstall component:

a. Open this InstallAnywhere project file 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**.

b. Set the **shortName** property of this component from **Install** to `Uninstall_ZG-IA`.

c. Save the project file.

7. Check the **shortName** property:

a. Open the project file in the InstallAnywhere Advanced Designer.

b. In the Advanced Designer, on the **Organization** page, click **Components**. The **Components** view opens.

You will see that the Short Name field for the InstallAnywhere Uninstall Component is now set to Uninstall_ZG-IA.

Customizing the Post-Install Sequence

The Post-Install sequence contains actions and panels that occur after the installation of files. You can use the Post-Install view on the Sequence page to add, configure, and remove items in the Post-Install sequence of your project.

By default, a new InstallAnywhere project contains the following items in the Post-Install sequence:

- **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**—If the installer determines that it is necessary to restart a Windows-based system, this action restarts the system.

Installations that are created in InstallAnywhere are controlled primarily by InstallAnywhere rules. The rules that are set by default for the Restart Windows action are simple rules that are 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 actions to the Post-Install sequence:***

1. In the Advanced Designer, on the **Sequence** page, click **Post-Install**. The **Post-Install** view opens.

2. Click the **Add Action** button. The **Choose an Action** dialog box opens.

3. Click the action you want to add, and then click the **Add** button. InstallAnywhere adds the action to the **Post-Install Action List**. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. Use the arrow keys to move the action up or down in the action list.

To specify an action's settings, select the action and use its customizers at the bottom of the view. For detailed information on each action's customizer settings, see Actions.

Customizing the Uninstall Sequence

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.

In some scenarios, you may want additional flexibility and more control over how the uninstallation is performed. Therefore, you may want to use the Uninstall view on the Sequence page 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 panels, or execute some custom code at uninstall time.

Customizing the Uninstall sequence 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 uninstallation 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 sequence, rather than adding an Execute Uninstaller action to end of the Pre-Uninstall sequence to perform this task.

About Uninstaller Customization

To customize the uninstaller, use the Uninstall view on the Sequence page in the Advanced Designer. In the Uninstall view, actions are grouped into uninstall categories.

By default, the following uninstall categories are created:

- Files
- LaunchAnywheres
- Shortcuts/Links/Aliases
- Registry Entries

- Built-in 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 change the order of the 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 Sequence

To customize uninstall behavior, you can add actions to the Uninstall sequence, and group the actions into uninstall categories.



Task ***To add an uninstall category or action to the Uninstall sequence:***

1. In the Advanced Designer, on the **Sequence** page, click **Uninstall**. The **Uninstall** view opens.

   Note ▪ *By default, all of the Uninstall actions are listed in the **Visual Tree** of the **Uninstall** view. If you delete one of these Uninstall actions and want to re-add it later, you can do so.*

2. To add an uninstall category, do the following:

   a. Click the **Add Action** button. The **Choose an Action** dialog box opens.

   b. On the **Uninstall** tab, select **Uninstall Category** and click **Add**. InstallAnywhere adds a new uninstall category to the **Visual Tree** and displays its settings at the bottom of the view.

   c. On the **Properties** tab, in the **Category Name** box, enter a name for the new category.

   d. With the new uninstall category selected, use the arrow keys to move the category up or down in the **Visual Tree**.

   To add an action, do the following:

   a. Select the uninstall category to which you want to add an action.

   a. Click the **Add Action** button. The **Choose an Action** dialog box opens.

   b. Select the action that you want to add, and then click **Add**. InstallAnywhere adds the new action to the **Visual Tree** and displays its settings at the bottom of the view.



Tip ▪ *If you added a General, Plug-In, or Console action to the Uninstall sequence, you can assign it to a product feature or component by selecting the appropriate check boxes. To learn more, see Assigning Actions in the Uninstall Sequence to Product Features and Components.*

Special Considerations for Adding Uninstall Categories and Actions to the Uninstall Sequence

Note the following guidelines about adding uninstall categories and actions:

- Only Uninstall actions, General actions, and Panel 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 end user during a console installation, one Console action—Show Message Console 'Dialog'— is also available.
- Each of the uninstall categories and Uninstall actions can be added to the Visual Tree in the Uninstall view only once. If they are added more than one time, the second instance is ignored.
- You can assign rules to uninstall categories and to actions in those categories to prevent them from being executed or uninstalled.
- Although you can assign General, Plug-In, and Console actions in the Uninstall view to components or features, it is not possible to assign uninstall categories or Uninstall actions to components or features. This is because uninstall categories and Execute Uninstall actions are special actions that are not feature or component specific; they are shared across all components and features.

Assigning Actions in the Uninstall Sequence to Product Features and Components

To assign a General, Plug-In, and Console actions to a product feature or component, use the Uninstall view on the Sequence page.



***Note** ▪ Although you can assign General, Plug-In, and Console actions in the Uninstall sequence to components or features, it is not possible to assign uninstall categories or Uninstall actions to components or features. This is because uninstall categories and Execute Uninstall actions are special actions that are not feature or component specific; they are shared across all components and features.*



Task ***To assign an uninstall action to a feature or component:***

1. In the Advanced Designer, on the **Sequence** page, click **Uninstall**. The **Uninstall** view opens.
2. In the **Assign to** list, select **Product Features** or **Components**. All of the project features or components are listed.
3. In the **Visual Tree**, find the action that you want to assign to a feature or component. In the same row, select the option that corresponds with the feature or component to which you want to assign the selected action.

Preventing the Actions in an Uninstall Category from Being Uninstalled

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 the category is skipped during uninstallation, or you can assign a rule so that the uninstall category is not executed.

Removing an Action from an Uninstall Category



Task ***To remove an action from an uninstall category:***

1. In the Advanced Designer, on the **Sequence** page, click **Uninstall**. The **Uninstall** view opens.
2. In the **Visual Tree**, select the action that you want to prevent from being uninstalled.
3. Click the **Remove** button.

InstallAnywhere removes the action from the Visual Tree.

Assigning a Rule to an Uninstall Category to Prevent Its Actions from Running



Task ***To assign a rule to an uninstall category:***

1. In the Advanced Designer, on the **Sequence** page, click **Uninstall**. The **Uninstall** view opens.
2. In the **Visual Tree**, select the uninstall category that you want to configure.
3. Click the **Rules** tab.
4. Click the **Add Rule** button. The **Choose a rule** dialog box opens.
5. Select the rule that you want to add, and then click the **Add** button.

InstallAnywhere adds the rule to the rules list. Configure its settings as needed to specify the conditions that must be met for the uninstall category's actions to be run.

Reordering Uninstall Categories and Actions in the Uninstall Sequence

The actions and uninstall categories that are listed in the Visual Tree in the Uninstall view on the Sequence page are organized in chronological order, according to when they are launched during the uninstallation. To change the order of an action or an uninstall category, move them to the appropriate location in the Visual Tree.



Task ***To change the order of actions and or categories in the Uninstall sequence:***

1. In the Advanced Designer, on the **Sequence** page, click **Uninstall**. The **Uninstall** view opens.
2. In the **Visual Tree**, select the action or the uninstall category that you want to reorder.
3. Use the **Move Action Up** and **Move Action Down** buttons to change the sequence as needed.

Defining Rules and Rule Expressions that Evaluate Conditions on Target Systems

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

InstallAnywhere also includes a rule expression manager which allows installer authors to:

- Create complex rule expressions at the project level
- Combine multiple rules and save them to a single rule expression
- Reuse a rule or a set of rules across the project
- Associate a rule expression to a file extension, including the option to automatically associate the rule expression to a file whenever a file containing that file extension is added to the project
- Remove associated rules from custom file extensions

The following sections provide procedures for various functionality that you can do with project level rule expressions:

- Configuring and Saving a New Rule Expression
- Associating a Rule Expression to a File Extension
- Loading a Rule Expression
- Deleting a Saved Rule Expression

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 that you assign do not evaluate to true, your installer does not run.



Task ***To assign a rule to the installer:***

1. In the Advanced Designer, on the **Project** page, click **Installer Rules**. The **Installer Rules** view opens.
2. Click the **Add Rule** button. The **Choose a Rule/Expression** dialog box opens.
3. Select the rule that you want to add for the installer and then click the **Add** button.

InstallAnywhere adds the rule to the rules list. Configure the rule's settings as needed.



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 executes only when that rule returns a true result.



Task ***To assign a rule to an action:***

1. In the Advanced Designer, on the **Organization** page, click the sequence that contains the action that you want to configure. The corresponding view opens.
2. In the list of actions, select the action that you want to configure.

3. In the rule's customizer, click the **Rules** tab.
4. Click the **Add Rule** button. The **Choose a rule** dialog box opens.
5. Select the rule that you want to add for the installer and then click the **Add** button.

InstallAnywhere adds the rule to the rules list. Configure the rule's settings as needed.



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

Assigning a Rule to a Group of Actions

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. Create an Action Group action:
   a. In the Advanced Designer, on the **Organization** page, click the sequence that contains the action that you want to configure. The corresponding view opens.
   b. Click the **Add Action** button. The **Choose an Action** dialog box opens.
   c. Click the action you want to add, and then click the **Add** button. InstallAnywhere adds the action to the action list. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
   d. Use the arrow keys to move the action up or down in the action list.

   ***Tip** ▪ When you nest actions within an action group, any rules that you assign to the action group must evaluate to true for these actions to run.*
2. Configure the rules for the action group:
   a. On the Action Group's customizer, click the **Rules** tab.
   b. Click the **Add Rule** button. The **Choose a rule** dialog box opens.
   c. Select the rule that you want to add for the installer and click the **Add** button.

InstallAnywhere adds the rule to the rules list. Configure the rule's settings as needed.

Comment Action Groups

To prevent an action group (and all of the actions that it contains) from being bundled with the installer, select the Comment Action Group check box on the Properties tab of the Action Group customizer. When this check box is selected, a overlay icon appears on top of the action group icon in the action list to indicate its change in status:



Figure 5-1: Comment Action Group

The Comment Action Group option gives you an easy way to retain the actions in an action group in your installation project but to not include them in the built installer. If, at some future time, you wish to include these actions in the installer, you can clear the selection of the Comment Action Group option.

Building Complex Rule Expressions

InstallAnywhere lets you 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 on the item to which you are assigning it:

- **Installers**—To evaluate a rule before any installation takes place, use the Manage Expressions view on the Project page or the Installer Rules view on the Project page. The Manage Expressions view on the Project page lets you create and save complex expressions. For more information see Using the Rules Manager to Create Complex Rule Expressions at the Project Level.
- **Files, Panels, Consoles, Actions, Action Groups**—To evaluate a rule for a file, panel, console, action, or action group, assign it on the Rules tab of the item customizer in the appropriate view on the Sequence page.

When you click the Add Rule button 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.



Important ▪ *When you add a rule, 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)
```

For the logical operators, use the following symbols:

Table 5-4 ▪ Logical Operators

Symbol	Operator	Description
&&	and	Both rules must evaluate as true for the installer to continue. If any rule fails to pass, the installer stops and issues the failure message.
||	or	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	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:

```
CP799 && CSA304 || (CL990 && ! CIAVN203)
```



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. For more information, 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

To validate a rule expression when you are defining it in the Advanced Designer, click the Validate Rule Expression button. InstallAnywhere indicates above the rule whether the rule expression is valid.

To clear all content in the Rule Expression field, click the Clear Rule Expression button.

Validation at Build Time

InstallAnywhere performs rule validation when a project is built. An invalid rule does not prevent the build from completing. However, it does trigger a warning message on the Building dialog box (or the console); the warning message indicates the sequence where the validation failed. If the validation of a rule fails, it is not evaluated at run time.

Using the Rules Manager to Create Complex Rule Expressions at the Project Level

InstallAnywhere includes a rule expression manager on the Manage Expressions view of the Project page which allows installer authors to:

- Create complex rule expressions at the project level
- Combine multiple rules and save them to a single rule expression
- Reuse a rule or a set of rules across the project
- Associate a rule expression to a file extension, including the option to automatically associate the rule expression to a file whenever a file containing that file extension is added to the project
- Remove associated rules from custom file extensions

The following sections provide procedures for various functionality that you can do with project level rule expressions:

- Configuring and Saving a New Rule Expression
- Associating a Rule Expression to a File Extension
- Loading a Rule Expression
- Deleting a Saved Rule Expression

Configuring and Saving a New Rule Expression



Task ***To configure and save a new rule expression at the project level***

1. Click the **Project** page and then click the **Manage Expressions** view.
2. Click **New Expression**. A name is automatically generated for the rule in the **Expression Name** field. You can enter a new name such as one that is more descriptive of what rule expression you are saving.
3. Click Add Rule to add a rule to the expression. The **Choose a rule** dialog box opens. Select the rule that you want to add, and then click the **Add** button. Configure its settings as needed to specify the conditions that must be met.

   As soon as you add a rule, the expression is saved and available to be loaded later.
4. To add additional rules, repeat Step 3 for as many additional rules you want to combine into the expression.
5. To add the expression to a selected project action:
   a. Click the **Sequence** page.
   b. Click the action you want to add a rule to.
   c. Click the **Rules** tab and click **Add Rule**. The **Choose a Rule/Expression** dialog box appears.
   d. On the **Choose a Rule/Expression** dialog box, select the **Expressions** tab, and then click the expression you want to add to the project action and click **Add**.



Note ▪ *Any changes that you make to a saved rule expression in the Manage Expressions tab are cascaded down to any actions that the rule expression has previously been applied to.*

Associating a Rule Expression to a File Extension

This section describes how to associate a rule expression to a file extension, and describes how to automatically associate the rule expression to a file whenever a file containing that file extension is added to the project on the Sequence page.



Note ▪ *The procedure provided assumes that the rule expression you will be associating to a file extension has already been saved. If you do not have a rule expression saved, refer to Configuring and Saving a New Rule Expression for information about how to create a rule expression.*



Task ***To associate a rule expression to a file extension***

1. Click the **Project** page and then click the **Manage Expressions** view.
2. In the **Associate Expression to a File Extension** section, click Add. A **Choose an Expression** dialog appears.
3. Choose the expression that you want to associate to a file and then click **Load**.
4. Click the **File Extension** section in the same row where the expression has been added, and enter the name of the file extension (such as, .sh for a Linux shell executables files.
5. By default, the **Apply** check box is checked. This means that any files in the project that contain the file extension you have specified will be updated to have the rule expression applied to them. Conversely, if you uncheck the **Apply** check box and then click the **Apply** button, any files with the specified file extension will remove the rule expression from them.
6. Optionally, click the **Apply automatically when new files are added** check box to automatically associate the rule expression to a file whenever a file containing that file extension is added to the project on the Sequence page.

Loading a Rule Expression



Task ***To load an existing project level rule expression to reconfigure it***

1. Click the **Project** page and then click the **Manage Expressions** view.
2. Click **Load Expression**. A **Choose an Expression** dialog box appears. Select the expression that you want to load and reconfigure the expression as needed by:
   - Removing rules using the **Remove Rule** button.
   - Adding new rules to the expressing by clicking the **Add Rule** button.



Note ▪ *Any changes that you make to a saved rule expression in the Manage Expressions tab are cascaded down to any actions that the rule expression has previously been applied to.*

Deleting a Saved Rule Expression



Task ***To delete an existing rule expression***

1. Click the **Project** page and then click the **Manage Expressions** view.
2. Click **Load Expression**. A **Choose an Expression** dialog box appears. Select the expression that you want to delete.
3. Click **Delete Expression**.
   - Removing rules using the **Remove Rule** button.
   - Adding new rules to the expressing by clicking the **Add Rule** button.



Note ▪ *Any project actions that the rule expression has been previously been applied to will no longer use the rule expression.*

Customizing Built-in Rules

This section describes how to configure settings for some of InstallAnywhere's built-in rules.

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 do the following:

- Specify a file or folder to check
- Determine if that file or folder exists
- Determine if that file is empty
- 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 installations, 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. In the Advanced Designer, on the **Project** page, click **Installer Rules**. The **Installer Rules** view opens.
2. Click the **Add Rule** button. The **Choose a Rule/Expression** dialog box opens.
3. Click the **Check File/Folder Attributes** action, and then click the **Add** button. InstallAnywhere adds the rule to the rules list.
4. In the **File/Folder Path** setting, enter the following value:

   ```
   $INSTALLER_LAUNCH_DIR$/installer.properties
   ```

5. Leave 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 sequence, InstallAnywhere includes a separate Check if File/Folder Exists rule. This rule is not available in Installer Rules view on the Project page, but you can add it via the customizer's Rules tab in the Install view.*

Customizing a Check If File/Folder Exists Rule



Note ▪ *This rule is available for file- and folder-related actions in the Install view on the Sequence page.*

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 and check if its empty.



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.*

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

When you enable maintenance mode in a project, InstallAnywhere automatically adds Check Running Mode rules to the action groups that are in the Pre-Install and Pre-Uninstall sequences.

In addition, you can also manually apply a Check Running Mode rule to action groups, actions, or panels in your installation project to specify whether that element should be executed during install mode, upgrade mode, or maintenance mode. This enables you to customize the events that occur when an end user launches various modes.



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, on the **Sequence** page, click the appropriate sequence.
2. Select the action group, action, or panel to which you would like to add a **Check Running Mode** rule.
3. Select the **Rules** tab.
4. Click the **Add Rule** button. The **Choose a rule** dialog box opens.
5. Select the **Check Running Mode** action and then click the **Add** button. The rule is now listed in the **Rules List**.
6. In the Check Running Mode customizer, select the appropriate option to associate the selected element with one of the maintenance mode options. The list of options varies, depending on which sequence contains the action.

   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 are 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 that contains the custom code and package it in a `.jar` or `.zip` file.
2. In the Advanced Designer, on the **Sequence** page, click the sequence that contains the action group, action, or panel that should contain a custom rule.
3. In the list of actions, select the one that you want to contain a custom rule.
4. Select the **Rules** tab.
5. Click the **Add Rule** button. The **Choose a rule** dialog box opens.
6. Select **Evaluate Custom Rule** and then click the **Add** button. InstallAnywhere adds the rule to the rules list.
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 that is associated with it will be installed or performed; if the rule fails, the action is skipped.

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, on the **Sequence** page, select the action group, action, or panel to which you would like to add a **Compare InstallAnywhere Variable Numerically** rule.
2. Select the **Rules** tab.
3. Click the **Add Rule** button. The **Choose a Rule** dialog box opens.
4. Select **Compare InstallAnywhere Variable Numerically** and then click the **Add** button. InstallAnywhere adds the rule to the rules list.
5. 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`*).*
6. In the list, select one of the following operators:
   - greater than or equal to (>=)
   - equal to (==)
   - less than (<)
   - less than or equal to (<=)
   - greater than (>)
   - greater than or equal to (>=)

How Rules Are Evaluated at Run Time

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.*

Configuring Servers

InstallAnywhere enables you to configure support for managing application and database servers:

- Managing Application Servers
- Managing Database Servers

Managing Application Servers

InstallAnywhere includes built-in support for deploying Web applications to the following types of application servers:

- Geronimo 1.1.1 or later
- JBoss 4.0.5 or later

- Tomcat 6.0.x, 7.0.x, and 8.0
- WebLogic 10.0
- WebSphere 7, 8, and 8.5

The application support within InstallAnywhere enables you to do the following:

- Deploy Web applications to application servers
- Enable remote deployment for Geronimo, Tomcat, WebLogic, and WebSphere application servers. Remote deployment enables end users to run the installer on a machine that can connect to and deploy Web applications to an application server.

  All of the supported application server types support local deployment; that is, they enable end users to deploy Web applications to the application server on which they are running the installer.
- For Tomcat and WebSphere servers: Display run-time panels or consoles that enable end users to specify where they would like to deploy the Web application. You can optionally provide end users with the option of deploying to local or remote servers, or to save the WAR locally on the target system for later deployment.

This section of the documentation explains how to configure the aforementioned support in your projects.

Adding an Application Host to Your Project

Before you can configure WAR and EAR files to your project, you need to add an application host to your project.



Task ***To add an application host to your project:***

1. In the Advanced Designer, on the **Organization** page, click **Hosts**. The **Hosts** view opens.
2. Click the **Add Host** button. The **Choose a Host** dialog box opens.
3. Click the **Application Server** host.

InstallAnywhere adds a default Geronimo host to the host list. In addition, InstallAnywhere adds the host to the Install sequence of your project.

To learn how to specify a WAR or EAR archive that you want to deploy to this host, see Deploying Web Applications (WAR and EAR Archives) to Servers.

To change the type of application server or specify other information for the host that you added, configure the server settings in the area below the list of hosts in the Hosts view.

Specifying Which Deployment Options to Support for Apache Tomcat Servers

When you are configuring a Tomcat server host in your project, you can specify which types of deployment options you want your installer to support. You can offer all or any combination of the following options:

- Local Tomcat server—The Web application is deployed to the server on which the installer is running.
- Remote Tomcat server—The installer runs on one machine and connects to a separate Tomcat server to deploy the Web application.

Note that the remote Tomcat server must be running at the time of deployment. In addition, the Tomcat Manager application on that server must be deployed with all of the appropriate roles and permissions, and autodeployment functionality must be enabled.

To test deployment to a remote Tomcat server, add the following child elements to the <tomcat-users> element of the `tomcat-users.xml` file (which is in the conf folder in the Tomcat install location). Substitute the user name and password as needed. Use the same credentials when specifying the remote Tomcat server information during the installation.

```
<role rolename="manager-script" />
<user username="MyUserName" password="MyPassword" roles="manager-script"/>
```

Also note that when the product is uninstalled, the uninstaller does not remove the Web application resources from the remote Tomcat server.

- Save the WAR file locally on the machine on which the installer is running for later deployment.

  Note that the installer saves the WAR file in the following location:

  ```
  $USER_INSTALL_DIR$/_$PRODUCT_NAME$_installation/tomcat_undeployed_war
  ```

  When the product is uninstalled, the uninstaller removes the undeployed WAR file from that location.



Task ***To specify which deployment options that you want your installer to support:***

1. In the Advanced Designer, on the **Organization** page, click **Hosts**. The **Hosts** view opens.
2. In the **Host List**, click the Tomcat server host that corresponds with the server that you want to configure.
3. At the bottom of the view, in the **Deployment Options** area, select the check box of each option that you want your installer to support. Available options are:
   - Local Tomcat
   - Remote Tomcat
   - Save the WAR locally on the target system for later deployment



Tip ▪ *As an alternative, you can configure deployment options when you click the Tomcat server host in the Install view on the Sequence page.*

By default, InstallAnywhere uses the user-defined variable $TOMCAT_DEPLOYMENT_OPTION$ to indicate which option to use for deploying WAR files at run time. Valid values for this variable are:

Table 5-5 ▪ Values for the $TOMCAT_DEPLOYMENT_OPTION$ Variable

Value	Description
1	Local Tomcat
2	Remote Tomcat

Table 5-5 ▪ Values for the $TOMCAT_DEPLOYMENT_OPTION$ Variable (cont.)

Value	Description
3	Save the WAR locally on the target system for later deployment

You can use a run-time panel or console to enable end users to select which option that they want to use. As an alternative, you can add Set InstallAnywhere Variable actions to a sequence in your project to determine which option to use for deployment. To learn more, see Enabling End Users to Specify Apache Tomcat Server Information.

Specifying Which Deployment Options to Support for IBM WebSphere Servers

When you are configuring a WebSphere server host in your project, you can specify which types of deployment options you want your installer to support. You can offer one or both of the following options:

- Local or remote WebSphere server—The Web application is deployed to the server on which the installer is running, or the installer runs on one machine and connects to a separate WebSphere server to deploy the Web application.

  Note that the remote WebSphere server must be running at the time of deployment. In addition, it must be a stand-alone applications server.

- Save the EAR or WAR file locally on the target system for later deployment.

  Note that the installer saves the EAR or WAR file in the following location:

  `$USER_INSTALL_DIR$/_$PRODUCT_NAME$_installation/websphere_undeployed_archive`

  When the product is uninstalled, the uninstaller removes the undeployed EAR or WAR file from that location.



Task ***To specify which deployment options that you want your installer to support:***

1. In the Advanced Designer, on the **Organization** page, click **Hosts**. The **Hosts** view opens.
2. In the **Host List**, click the WebSphere server host that corresponds with the server that you want to configure.
3. At the bottom of the view, in the **Deployment Options** area, select the check box of each option that you want your installer to support. Available options are:
   - Local or remote WebSphere
   - Save the WAR/EAR locally on the target system for later deployment



Tip ▪ *As an alternative, you can configure deployment options when you click the WebSphere server host in the Install view on the Sequence page.*

By default, InstallAnywhere uses the user-defined variable $WEBSPHERE_DEPLOYMENT_OPTION$ to indicate which option to use for deploying EAR and WAR files at run time. Valid values for this variable are:

Table 5-6 ▪ Values for the $WEBSPHERE_DEPLOYMENT_OPTION$ Variable

Value	Description
1	Local or remote WebSphere server
2	Save the EAR or WAR locally on the target system for later deployment

You can use a run-time panel or console to enable end users to select which option that they want to use. As an alternative, you can add Set InstallAnywhere Variable actions to a sequence in your project to determine which option to use for deployment. To learn more, see Enabling End Users to Specify IBM WebSphere Server Information.

Deploying Web Applications (WAR and EAR Archives) to Servers

InstallAnywhere offers a Deploy WAR/EAR Archive action that enables you to deploy Web applications to application servers. Before you can add this action to a sequence on the Sequences page, you need to add an application server host in the Hosts view. To learn how, see Adding an Application Host to Your Project.



Task ***To add a Deploy WAR/EAR Archive action to your project:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. In the **Visual Tree** list, click the application host that corresponds with the type of server to which you want to deploy a Web application.
3. Click the **Add Action** button. The **Choose an Action** dialog box opens.
4. On the **Install** tab, click the **Deploy WAR/EAR Archive** action. InstallAnywhere adds the action to the **Visual Tree**.
5. In the **Properties Customizer** area, configure each of the action's settings as needed.

For detailed information about each of the settings that you can configure for the action, see Deploy WAR/EAR Archive Action.

Enabling End Users to Specify Apache Tomcat Server Information

You can add to your projects run-time panels or consoles that enable end users to specify settings for Apache Tomcat servers before deploying Web applications to these servers. For example, you can enable end users to choose between deploying applications to local or remote Tomcat servers. You can also let end users specify information such as the path on the local server where they want to deploy the Web application or the name of the remote Tomcat server.



Tip ▪ *Before you can add the Tomcat server run-time panel or console to your project, you need to add an application host and specify which deployment options you want to offer. To learn more, see the following:*

- *Adding an Application Host to Your Project*
- *Specifying Which Deployment Options to Support for Apache Tomcat Servers*



Task ***To add a Tomcat Runtime Deployment panel or console to your project:***

1. In the Advanced Designer, on the **Sequence** page, click **Pre-Install**. The **Pre-Install** view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. Do one of the following:
   - To add a Choose Tomcat Deployment Option panel: On the **Panels** tab, click the **Panel: Tomcat Runtime Deployment** action, and then click the **Add** button. InstallAnywhere adds the action to the **Pre-Install Action List**.
   - To add a Choose Tomcat Deployment Option console: On the **Consoles** tab, click the **Console: Tomcat Runtime Deployment** action, and then click the **Add** button. InstallAnywhere adds the action to the **Pre-Install Action List**.

   The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. Use the arrow keys to move the action up or down in the action list. The actions are listed in the order that they are run on target systems.
5. In the **Properties Customizer** area, configure each of the action's settings as needed.

For detailed information about each of the settings that you can configure for the actions, see the following:

- Tomcat Runtime Deployment Panel Action
- Tomcat Runtime Deployment Console Action

As an alternative to including a panel or console that enables end users to specify settings for Apache Tomcat servers, you can add Set InstallAnywhere Variable actions to a sequence in your project to configure connection information. To learn more, see Setting Variables in the Advanced Designer.

To configure connection information for silent installers, you can use the user-defined Tomcat-related variables that are available in the Tomcat Runtime Deployment actions or in the Deploy WAR/EAR action.

Enabling End Users to Specify IBM WebSphere Server Information

You can add to your projects run-time panels or consoles that enable end users to specify settings for IBM WebSphere servers before deploying Web applications to these servers. For example, you can let end users specify information such as the local or remote WebSphere server host where they want to deploy the Web application.



Tip ▪ *Before you can add the WebSphere server run-time panel or console to your project, you need to add an application host and specify which deployment options you want to offer. To learn more, see the following:*

- *Adding an Application Host to Your Project*
- *Specifying Which Deployment Options to Support for IBM WebSphere Servers*



Task ***To add a WebSphere Runtime Deployment panel or console to your project:***

1. In the Advanced Designer, on the **Sequence** page, click **Pre-Install**. The **Pre-Install** view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. Do one of the following:
   - To add a Choose WebSphere Deployment Option panel: On the **Panels** tab, click the **Panel: WebSphere Runtime Deployment** action, and then click the **Add** button. InstallAnywhere adds the action to the **Pre-Install Action List**.
   - To add a Choose WebSphere Deployment Option console: On the **Consoles** tab, click the **Console: WebSphere Runtime Deployment** action, and then click the **Add** button. InstallAnywhere adds the action to the **Pre-Install Action List**.

   The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. Use the arrow keys to move the action up or down in the action list. The actions are listed in the order that they are run on target systems.
5. In the **Properties Customizer** area, configure each of the action's settings as needed.

For detailed information about each of the settings that you can configure for the actions, see the following:

- WebSphere Runtime Deployment Panel Action
- WebSphere Runtime Deployment Console Action

As an alternative to including a panel or console that enables end users to specify settings for IBM WebSphere servers, you can add Set InstallAnywhere Variable actions to a sequence in your project to configure connection information. To learn more, see Setting Variables in the Advanced Designer.

To configure connection information for silent installers, you can use the user-defined WebSphere-related variables that are available in the WebSphere Runtime Deployment actions or in the Deploy WAR/EAR action.

Managing Database Servers

InstallAnywhere includes built-in support for managing the following types of SQL databases:

- DB2 9.0, 9.7, 10.1, and 10.5
- MySQL 5.1.29, 5.5, and 5.6
- Microsoft SQL Server 2008 R2, 2012, and 2014
- Oracle 10.2.0.1, 10.2.0.2, 10.2.0.3, and 10.2.0.4
- PostgreSQL 7.0-8.3

InstallAnywhere also lets you use generic JDBC connections if you need to run SQL scripts on a type of database for which InstallAnywhere does not have built-in support.

The database support within InstallAnywhere enables you to do the following:

- Connect to local servers.
- Run different SQL scripts during installation and uninstallation.

Support for MySQL, Microsoft SQL Server, IBM DB2, and PostgreSQL databases offers the following additional support:

- Connect to remote SQL servers.
- Display run-time panels or consoles that enable end users to specify connection information such as the name of the server, as well as the credentials that should be used to connect to the server through server authentication.
- Enable end users to test the connection information that they entered.
- Optionally create SQL databases on servers. (This is not available for DB2 databases.)

This section of the documentation explains how to configure the aforementioned support in your projects.

Adding a SQL Database Host to Your Project

Before you can configure SQL script information in your project, you need to add a database host to your project.



Task ***To add a database host to your project:***

1. In the Advanced Designer, on the **Organization** page, click **Hosts**. The **Hosts** view opens.
2. Click the **Add Host** button. The **Choose a Host** dialog box opens.
3. Click the **Database Server** host.

InstallAnywhere adds a default MySQL host to the host list. In addition, InstallAnywhere adds the host to the Install sequence of your project. To learn how to specify a SQL script that you want to run on this host, see Running a SQL Script.

To change the type of database server or specify other information for the host that you added, configure the server settings in the area below the list of hosts in the Hosts view.

Including JDBC Drivers for Connecting to SQL Databases

The SQL database support in InstallAnywhere requires JDBC drivers for connecting to database servers. The drivers that are required vary, depending on the type of database that you are targeting.

Connecting to Microsoft SQL Server Databases and PostgreSQL Databases

When you are installing InstallAnywhere and you choose the Typical install set or you include the JDBC Drivers feature for the Custom install set, the installer installs drivers for connecting to some of the SQL databases that InstallAnywhere supports:

Table 5-7 ▪ Drivers that Are Installed with InstallAnywhere by Default

Database	Driver	Default Location
Microsoft SQL Server	jTDS	Driver: IA_HOME/resource/db/drivers/jTDS License: IA_HOME/resource/db/drivers/jTDS/LICENSE.TXT
PostgreSQL	PostgreSQL JDBC	Driver: IA_HOME/resource/db/drivers/PostgreSQL License: IA_HOME/resource/db/drivers/PostgreSQL/LICENSE.TXT

The jTDS and PostgreSQL JDBC drivers are type 4 JDBC drivers. These drivers are platform independent. They are written in pure Java and communicate in the database systems' built-in network protocols. InstallAnywhere includes those drivers as dependencies in your projects automatically when you add these database server hosts to your projects.

If you plan on targeting Microsoft SQL Server or PostgreSQL databases, ensure that you review the license for the driver that you plan on using to ensure that your distribution plans comply with the driver's license requirements.

Connecting to DB2, MySQL, Oracle, and Other Databases

Because of licensing restrictions, InstallAnywhere does not provide the drivers for DB2, MySQL, or Oracle. Therefore, if you want to configure your projects to manage one or more of these types of SQL databases, you must obtain the appropriate drivers for these databases and add them to your projects.

In addition, InstallAnywhere includes a Generic JDBC Connection option—which is available as a database type—that you can choose if you want to target other types of databases.

To learn how to obtain the drivers for these database types and add them to your development machine for use with InstallAnywhere projects, see:

- Obtaining DB2 Drivers
- Obtaining a MySQL Driver
- Obtaining an Oracle Driver
- Obtaining a Generic JDBC Driver

Specifying JDBC Driver Information



Task ***To specify the JDBC driver information for a SQL database host in your project:***

1. In the Advanced Designer, on the **Organization** page, click **Hosts**. The **Hosts** view opens.
2. In the **Host List**, click the database host that corresponds with the server that you want to configure.
3. Configure the settings that are displayed at the bottom of the view.

For details on how to configure the available settings, see Database Server Host Settings.



Tip ▪ *As an alternative, you can configure JDBC driver information when you click the database host in the Install view on the Sequence page.*

Obtaining DB2 Drivers

The DB2 database support in InstallAnywhere requires JDBC drivers for connecting to DB2 database servers. Because of licensing restrictions, InstallAnywhere does not provide the drivers for DB2. Therefore, if you want to configure your projects to manage DB2 databases, you must obtain the appropriate drivers and add them to your projects.



Task ***To download a DB2 driver and make it available for inclusion in InstallAnywhere projects:***

1. Download the DB2 UDB JDBC Universal driver (`db2jcc.jar` and `db2jcc_license_cu.jar`) from the following site:

   https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=swg-dm-db2jdbcdriver

2. Extract `db2jcc.jar` and `db2jcc_license_cu.jar` from the DB2 UDB JDBC Universal Driver archive (`db2_v9_db2driver_for_jdbc_sqlj.zip`).
3. Copy `db2jcc.jar` and `db2jcc_license_cu.jar` to the following directory:

   `<InstallAnywhere>/resource/db/drivers/DB2/`

When you add a DB2 database server host to your project, verify that the locations that are shown for the `db2jcc.jar` and `db2jcc_license_cu.jar` dependencies in the Database Server customizer at the bottom of the Hosts view are correct. To find out more, click the **More on drivers** link in the Database Server customizer.

Obtaining a MySQL Driver

The MySQL database support in InstallAnywhere requires a JDBC driver for connecting to MySQL database servers. Because of licensing restrictions, InstallAnywhere does not provide the drivers for MySQL. Therefore, if you want to configure your projects to manage MySQL databases, you must obtain the appropriate drivers and add them to your projects.



Task ***To download a MySQL driver and make it available for inclusion in InstallAnywhere projects:***

1. Download the MySQL Connector/J library (`mysql-connector-java-5.0.6-bin.jar`) from the following site:

   http://www.mysql.com/products/connector-j

2. Extract `mysql-connector-java-5.0.6-bin.jar` from the MySQL Connector/J archive (`mysql-connector-java-5.0.6.zip`).

3. Copy `mysql-connector-java-5.0.6-bin.jar` to the following directory:

   `<InstallAnywhere>/resource/db/drivers/ConnectorJ/`

When you add a MySQL database server host to your project, verify that the location that is shown for the `mysql-connector-java-5.0.6-bin.jar` dependency in the Database Server customizer at the bottom of the Hosts view is correct. To find out more, click the **More on drivers** link in the Database Server customizer.

Obtaining an Oracle Driver

The Oracle database support in InstallAnywhere requires a JDBC driver for connecting to Oracle database servers. Because of licensing restrictions, InstallAnywhere does not provide the drivers for Oracle. Therefore, if you want to configure your projects to manage Oracle databases, you must obtain the appropriate drivers and add them to your projects.



Task ***To download an Oracle driver and make it available for inclusion in InstallAnywhere projects:***

1. Download the Oracle JDBC Driver (`ojdbc14.jar`) from the following site:

   http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/jdbc9201.html

2. Copy `ojdbc14.jar` to the following directory:

   `<InstallAnywhere>/resource/db/drivers/Oracle/`

When you add an Oracle database server host to your project, verify that the location that is shown for the `ojdbc14.jar` dependency in the Database Server customizer at the bottom of the Hosts view is correct. To find out more, click the **More on drivers** link in the Database Server customizer.

Obtaining a Generic JDBC Driver

The database support in InstallAnywhere requires a JDBC driver for connecting to database servers. InstallAnywhere includes a Generic JDBC Connection option—which is available as a database type—that you can choose if you want to target a type of database for which InstallAnywhere does not have built-in support. If you want to configure your projects to connect to databases through a generic JDBC driver, you must obtain the appropriate driver and add it to your machine that has InstallAnywhere.

When you select the Generic JDBC Connection option for a database host in your project, you must provide the connection string and identify the JDBC driver and its dependencies.

Running a SQL Script

InstallAnywhere offers a Run SQL Script action that enables you to run SQL scripts on local database servers. For MySQL, Microsoft SQL Server, IBM DB2, and PostgreSQL databases, this action also lets you run SQL scripts on remote database servers. Before you can add this action to a sequence on the Sequences page, you need to add a database server host in the Hosts view. To learn how, see Adding a SQL Database Host to Your Project.

The Run SQL Script action enables you to optionally create a SQL database on MySQL, Microsoft SQL Server, and PostgreSQL database servers at run time. Note the following details about the run-time behavior that occurs if you specify that you want the action to create the database:

- The user account that is used to connect to the server must have database administrator privileges.
- The action creates a basic database with default parameters.

  For example, for a MySQL database, the default character set and collation are used.
- The action does not verify whether the database name already exists for a database on the target system.



Task ***To add a Run SQL Script action to your project:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. In the **Visual Tree** list, click the database host that corresponds with the server on which you want to run the SQL script.
3. Click the **Add Action** button. The **Choose an Action** dialog box opens.
4. On the **Install** tab, click the **Run SQL Script** action. InstallAnywhere adds the action to the **Visual Tree**.
5. In the **Properties Customizer** area, configure each of the action's settings as needed.

For detailed information about each of the settings that you can configure for the action, see Run SQL Script Action.

Enabling End Users to Specify Database Connection Information

You can add to your projects run-time panels or consoles that enable end users to specify MySQL, Microsoft SQL Server, IBM DB2, and PostgreSQL database connection information such as the name of the server, as well as the credentials that should be used to connect to the server through server authentication. In addition, you can optionally enable end users to test the connection information that they entered.



Tip ▪ *Before you can add a database connection run-time panel or console to your project, you need to add a SQL database host. To learn more, see Adding a SQL Database Host to Your Project.*



Task ***To add a Choose Database Connection run-time panel or console to your project:***

1. In the Advanced Designer, on the **Sequence** page, click **Pre-Install**. The **Pre-Install** view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.

3. Do one of the following:
   - To add a Choose Database Connection panel: On the **Panels** tab, click the **Panel: Choose Database Connection** action, and then click the **Add** button. InstallAnywhere adds the action to the **Pre-Install Action List**.
   - To add a Choose Database Connection console: On the **Consoles** tab, click the **Console: Choose Database Connection** action, and then click the **Add** button. InstallAnywhere adds the action to the **Pre-Install Action List**.

   The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. Use the arrow keys to move the action up or down in the action list. The actions are listed in the order that they are run on target systems.
5. In the **Properties Customizer** area, configure each of the action's settings as needed.

For detailed information about each of the settings that you can configure for the actions, see the following:

- Choose Database Connection Panel Action
- Choose Database Connection Console Action

As an alternative to including a panel or console that enables end users to specify settings for database servers, you can add Set InstallAnywhere Variable actions to a sequence in your project to configure connection information. To learn more, see Setting Variables in the Advanced Designer.

To configure connection information for silent installers, you can use the user-defined SQL-related variables that are available in the Choose Database Connection actions or in the Run SQL Script action.



Important ▪ *SQL databases limit remote administration by users unless otherwise configured. Therefore, for connections to remote SQL database servers, the user name and the password that the end user specifies must be for a user who is running the installer from an appropriate IP address and has been granted adequate privileges that are required to run your SQL script.*

Preparing Your Installer for Maintenance Mode

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. When maintenance mode is enabled, end users can launch maintenance mode through the following methods:

- Execute the Change Installation launcher.
- On Windows-based target systems: In the Windows Control Panel, use Programs or Features (formerly called Add or Remove Programs).

Now you have an option, where you can restrict the display of the installation location for an instance in the Display Name displayed in the Add or Remove Programs (Programs and Features).

When an end user launches maintenance mode, they are prompted to select from the listed options, which may include Add Features, Remove Features, Repair Product, and Uninstall Product.

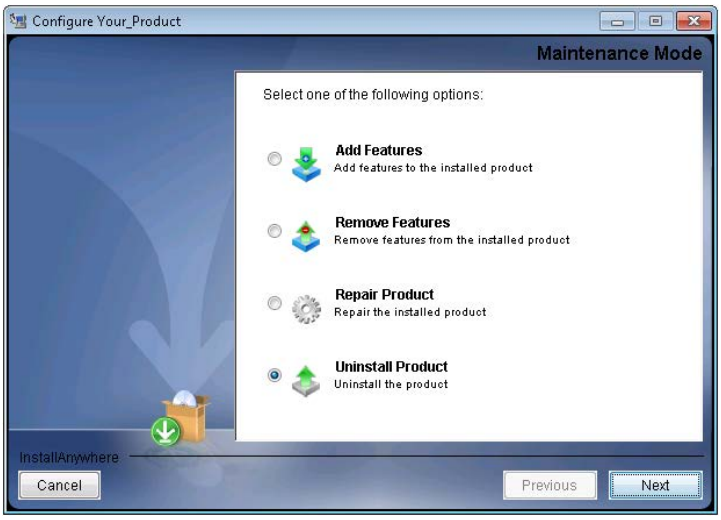


Figure 5-2: Maintenance Mode Panel

Note ▪ *The options listed on this panel depend upon the options that you select in the Maintenance Mode area of the Advanced view on the Project page. For more information, see Enabling and Configuring Maintenance Mode.*

Enabling and Configuring Maintenance Mode

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

Enabling Maintenance Mode

To enable maintenance mode support and identify an installer's supported maintenance mode options, use Advanced view on the Project page of your project.

Task ***To enable maintenance mode support and identify supported options:***

1. In the Advanced Designer, on the **Project** page, click **Advanced**. The **Advanced** view opens.
2. In the **Maintenance Mode** area, select the **Enable Maintenance Mode support** check box. InstallAnywhere enables the maintenance option check boxes, enabling you to select or clear each of them.
3. In the **Maintenance options** list, select the check boxes for the options that you want your installer to provide. You must select at least one check box; otherwise, a build error occurs.

For descriptions of each of the maintenance options, see Advanced View.

Tip ▪ *You can assign a Check Running Mode rule to associate elements of your installation project with the 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.*

About Maintenance Mode Action Groups

When you enable maintenance mode support to your project, InstallAnywhere adds new action groups and panels to the default Pre-Install and Pre-Uninstall sequences. Depending on which maintenance mode options that you selected (Add Features, Remove Features, Repair Installation, or Uninstall Product), separate action groups and panels are created to run when maintenance mode is launched. You can customize these as needed.

Pre-Install Sequence

When maintenance mode is enabled, InstallAnywhere adds the following action groups to the Pre-Install sequence:

- 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 support is not enabled, the Pre-Install sequence contains several panels.

As soon as you enable maintenance mode support, InstallAnywhere groups those panels into a new action group named Pre-Installation; in addition, InstallAnywhere adds a Check Running Mode rule with a value of Pre-Installation to that group.

InstallAnywhere provides these default action groups 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 InstallAnywhere assigns a Check Running Mode rule with the value of Pre-Installation to the Pre-Installation action group, these panels are executed only 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 enable the Add Features type of maintenance mode, InstallAnywhere adds the Add Features action group to the Pre-Install Action List; InstallAnywhere also assigns a Check Running Mode rule with a value of Add Features to that group.

By default, the Add Features action group consists of an Introduction panel and a Choose Install Sets panel.



***Note** ▪ Because InstallAnywhere assigns a Check Running Mode rule with the value of Add Features to the Add Features action group, these panels are executed during the Add Features phase of maintenance mode. For more information, see About Check Running Mode Rules.*

Repair Installation Action Group

If you enable the Repair Installation type of maintenance mode, InstallAnywhere adds the Repair Installation action group to the Pre-Install Action List; InstallAnywhere also assigns a Check Running Mode rule with a value of Repair Installation to that group.

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 is 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 InstallAnywhere assigns a Check Running Mode rule with the value of Repair Installation to the Repair Installation action group, these panels are executed only during the Repair Installation phase of maintenance mode. For more information, see About Check Running Mode Rules.*

Pre-Uninstall Sequence

When maintenance mode is enabled, InstallAnywhere creates the following action groups in the Pre-Uninstall sequence:

- Pre-Uninstallation action group
- Remove Features action group

Pre-Uninstallation Action Group

When you create a new project and maintenance mode support is not enabled, the Pre-Uninstall sequence contains several panels.

As soon as you enable maintenance mode support, InstallAnywhere groups those panels into a new action group named Pre-Uninstallation; in addition, InstallAnywhere adds a Check Running Mode rule with a value of Pre-Uninstallation to that group.



Note ▪ *Because InstallAnywhere assigns a Check Running Mode rule with the value of Pre-Uninstallation to the Pre-Uninstallation action group, these panels are executed only 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 enable the Remove Features type of maintenance mode, InstallAnywhere adds the Remove Features action group to the Pre-Uninstall sequence; InstallAnywhere also assigns a Check Running Mode rule with a value of Remove Features to that group.

By default, the Remove Features action group consists of an Introduction panel and a Choose Features to Uninstall panel.



Note ▪ *Because InstallAnywhere assigns a Check Running Mode rule with the value of Remove Features to the Remove Features action group, these panels are 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.

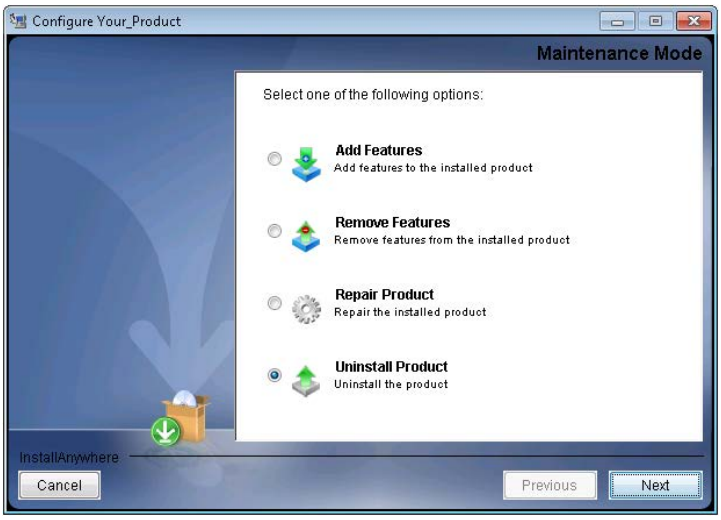


Figure 5-3: Maintenance Mode Panel of the Change Installation Launcher

You can make these customizations on the **Maintenance Mode Runtime Panel Label Settings** area of the **Installer UI > Look & Feel Settings** page.

Task ***To customize the text and images for maintenance mode:***

1. In the Advanced Designer, on the **Installer UI** page, click **Look & Feel Settings**. The **Look & Feel Settings** view opens.
2. Expand the **Maintenance Mode Runtime Panel Label Settings** area.
3. Configure the subsettings as needed.

For information on any of the settings, see Maintenance Mode Runtime Panel Label Settings Area.

About Check Running Mode Rules

As described in About Maintenance Mode Action Groups, when you enable maintenance mode support in your project, InstallAnywhere automatically adds Check Running Mode rules to the action groups that are created in the Pre-Install and Pre-Uninstall views of the Sequence page. 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 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 Behavior

The maintenance mode support in InstallAnywhere includes instance management functionality 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.



Caution ▪ *When setting up instance management behavior,* ***do not*** *include a variable in the name of the uninstaller launcher. This will break the instance management feature.*



Note ▪ *The instance count is only as secure as the security of the InstallAnywhere product registry file. For more information, see Product Registry.*



Important ▪ *The instance management options are only enabled if the* ***Enable Maintenance Mode support*** *check box is selected. However, you can provide maintenance mode support without enabling instance management support.*



Task ***To specify instance management behavior:***

1. In the Advanced Designer, on the **Project** page, click **Advanced**. The **Advanced** view opens.
2. In the **Maintenance Mode** area, select the **Enable Maintenance Mode support** check box. InstallAnywhere enables the maintenance option check boxes, as well as the **Enable Instance Management** options, enabling you to select or clear each of them.
3. In the **Instance Management** area, select the **Enable Instance Management** check box. InstallAnywhere enables the **Instance Management** options.
4. In the **Instance Management** area, select the appropriate check boxes.

To learn more, see Instance Management Settings.



Important ▪ *When an installer that has instance management support 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 file is named* `Change ProductName Installation.exe`*. However, if you have chosen to rename the executable file of the uninstaller, that file will not be located and the instance management support will not work properly.*



Note ▪ *If you include support for instance management functionality in your project, note the following:*

- *Instance management uses the global/local zerog registries. If these directories are altered, the instance management functionality does 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, the instance management functionality may not be able to manage instances, since it would be unable to find the Uninstaller.*

  The key file for the InstallAnywhere Uninstall Component is set in the Organization view (Organization page > Components view > Properties tab).



Note ▪ *A limitation of Windows-based installers in the Maintenance mode—when there is a cross bit installation, a 32-bit installer will not detect a 64-bit installation and a 64-bit installer will not detect a 32-bit installation, which results that both 32-bit installer and 64-bit installer are unable to display the* ***Instance Management*** *panel.*

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, which 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 5-8 ▪ 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 5-9 ▪ 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]

About the Uninstaller/Maintenance Mode Launcher

By default, an InstallAnywhere project is configured to create one uninstaller/maintenance mode executable file named `Change *Product_Name* Installation.exe`. You can modify the name of this executable file. (On the Sequence page, click Install. Then select the uninstall launcher and edit the value in the Name setting on the Properties tab.)

The behavior of this launcher varies, depending upon whether or not you have enabled maintenance mode support in your project:

- If you have not enabled maintenance mode support, opening this launcher launches the standard uninstall run time.

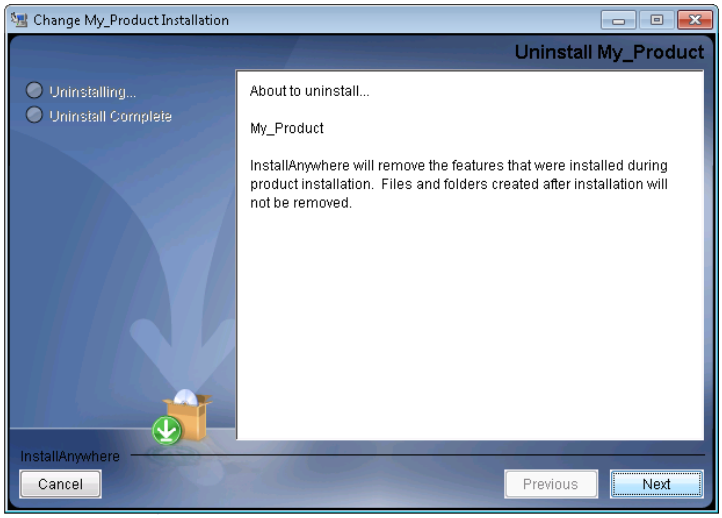


Figure 5-4: Uninstall Panel of the Change Installation Launcher

- If you have enabled maintenance mode support, 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).

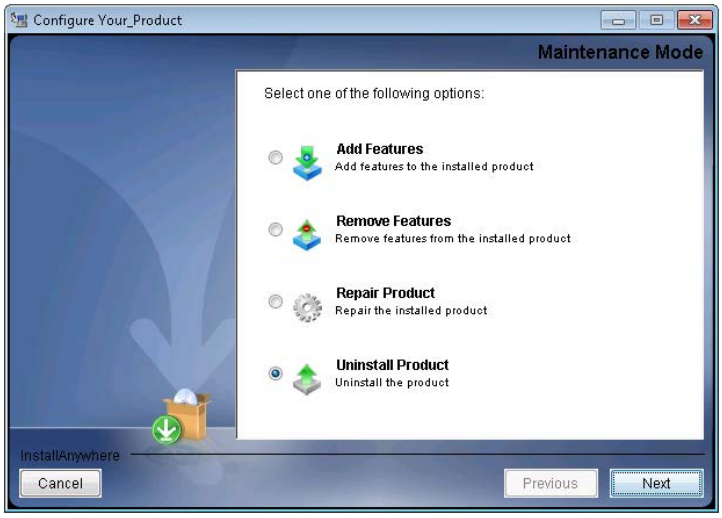


Figure 5-5: Maintenance Mode Panel of the Change Installation Launcher

Run-Time Behavior for Maintenance Mode

End users can launch maintenance mode through the following methods:

- Launch the Change Installation launcher (`Change` *`Product_Name`* `Installation.exe`).
- On Windows-based target systems: In the Windows Control Panel, use Programs or Features (formerly called Add or Remove Programs).

The run-time behavior varies, depending on which of the maintenance mode options that you selected in the Advanced view on the Project page.

Launching Maintenance Mode

The first panel that the maintenance mode run time displays is the Maintenance Mode panel. The Maintenance Mode panel shows the options that were selected in the Advanced view of the Project page:

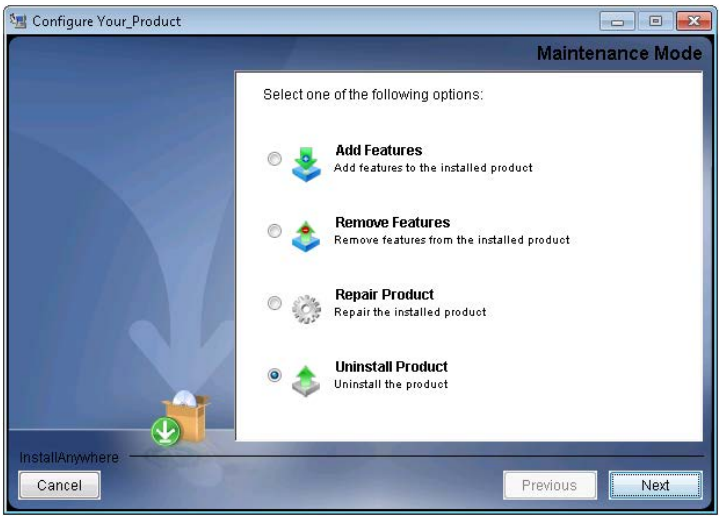


Figure 5-6: Maintenance Mode Panel

The Maintenance Mode panel appears only once. After the end user selects an option and clicks Next, they will be unable to return to this panel by clicking the Previous button.

Adding Features

If the end user selects the Add Features option on the Maintenance Mode panel and clicks Next, the maintenance mode run time displays the Choose Install Set panel. The Choose Install Set panel displays the features that are already installed and allows the selection of only the ones that are not yet installed.

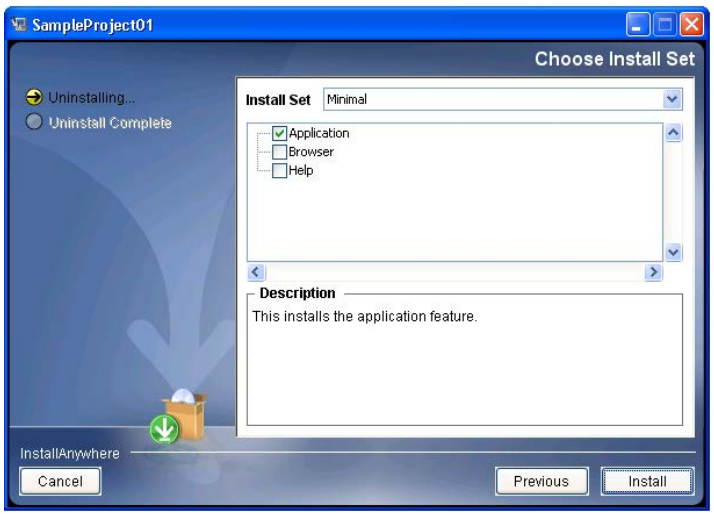


Figure 5-7: Adding Features

The maintenance mode run time executes only the action groups, actions, and panels that have been assigned a Check Running Mode rule with a value of Add Features.

Removing Features

If the end user selects the Remove Features option on the Maintenance Mode panel and clicks Next, the maintenance mode run time displays the Choose Product Features panel. The Choose Product Features panel displays the features that are installed and allows the end user to deselect the ones that need to be uninstalled.

The maintenance mode run time executes only the action groups, actions, and panels that have been assigned a Check Running Mode rule with a value of Remove Features.

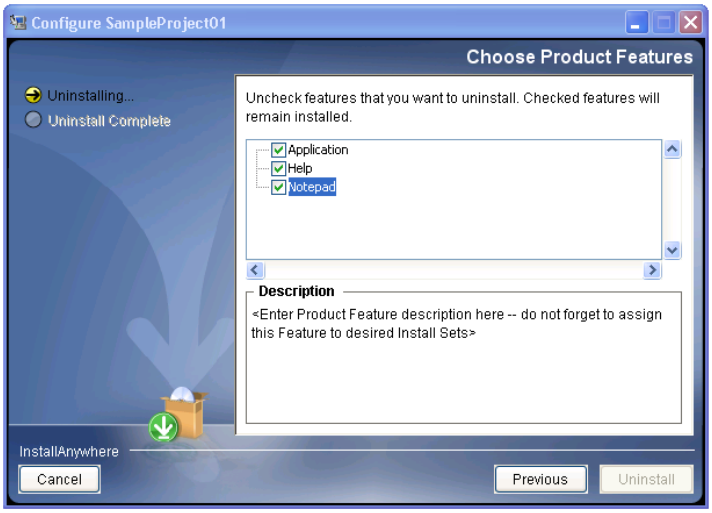


Figure 5-8: Removing Features

Repairing the Product

If the end user selects the Repair Product option on the Maintenance Mode panel and clicks Next, the maintenance mode run time displays the panels in the Repair Installation action group in the Pre-Install sequence and proceeds as a regular installation.

The maintenance mode run time executes only the action groups, actions, and panels that have been assigned a Check Running Mode rule with a value of Repair Installation.

Uninstalling the Product

If the end user selects the Uninstall Product option on the Maintenance Mode panel and clicks Next, the maintenance mode run time displays the panels in the Pre-Uninstallation action group in the Pre-Uninstall sequence and proceeds as a regular uninstallation.

The maintenance mode run time executes only the action groups, actions, and panels that have been assigned a Check Running Mode rule with a value of Pre-Uninstallation.

Maintenance Mode with Multiple Installed Instances

As described in Specifying Instance Management Behavior, 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. This dialog box prompts end users to either choose to install a new instance or select the instance that they want to modify.

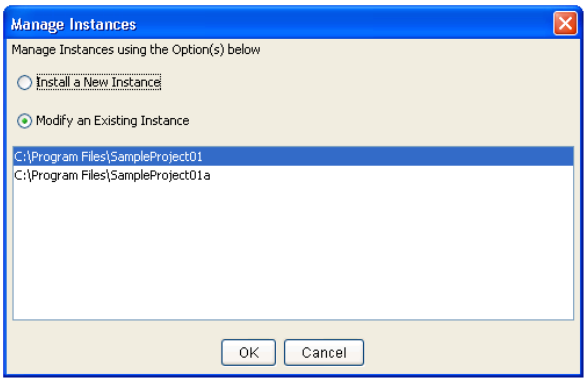


Figure 5-9: Manage Instances Dialog Box

Configuring Installation Rollback Behavior

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 check box in your project (Advanced Designer > Project page > Advanced view). This Enable Rollback check box is selected by default for all new projects that you create in InstallAnywhere.

If rollback is enabled in an installation, and the end user cancels the installation or a fatal error occurs, the installer automatically reverts what has been altered or added to the system. The installation log contains all status messages, including the action that triggered the rollback.

If you select the Enable Rollback check box, the **Restart Windows in case of Rollback** check box becomes available, allowing you to select or clear this check box. Select the **Restart Windows in case of Rollback** check box to instruct a Windows-based target system to restart after a rollback occurs during installation. If this check box is selected, the target system restarts once the end user clicks the Done button on the Install Complete panel.

Note ▪ *If the **Restart Windows in case of Rollback** check box is selected, the Windows-based target machine restarts regardless of whether a Restart Windows action is present in the Post-Install sequence, with one exception. If a Restart Windows action is present and you assign to it a rule that would prompt the end user to choose between **Yes, restart** or **No, I will restart later**, and if the end user selects the No option, the machine does not restart—regardless of whether the **Restart Windows in case of Rollback** check box is selected.*

Fine-Tuning the Rollback Behavior Through the Install Sequence

On the Rollback tab of action customizers in the Install view of the Sequence page, you can specify rollback behavior 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.

If the Enable Rollback check box is selected in the Advanced view of the Project page, you can select the options on the Rollback tab of action customizers in the Install view to specify rollback options for the selected project element. Available check boxes are:

- **If rollback is triggered at install time, uninstall this action/resource**—If you want the selected project element to be uninstalled if the product installation is cancelled or encounters a fatal error, select this check box. If this check box is not selected and a rollback occurs, the project element is not uninstalled.
- **If a fatal error occurs on this action/resource at install time, trigger uninstall**—If you want a rollback to be triggered if the selected project element fails to execute properly or to be installed properly, select this check box. If this check box is not selected and the execution of this action results in a fatal error during installation, a rollback is not triggered.

By default, both check boxes are selected for all project elements.

Disabling the Cancel Button

The end result of disabling the Cancel button depends upon whether the Enable Rollback check box in the Advanced view on the Project page is also selected. The following table presents the two scenarios:

Table 5-10 ▪ Interaction of Disabled Cancel Button with the Enable Rollback Check Box State

State of the Enable Rollback Check Box	Resulting Behavior
Not Selected	• The Cancel button and the window close button are disabled. • If a fatal error is encountered or a Trigger Rollback action occurs, the installation completes with errors. Rollback is suppressed.
Selected	• The Cancel button and the window close button are disabled. • If a fatal error is encountered or a Trigger Rollback action occurs, rollback is initiated.



Note ▪ *This same behavior occurs in maintenance mode–enabled projects when the end user attempts to add features, remove features, or repair the product.*

Using Trigger Rollback Actions

You can also choose to add a Trigger Rollback action to the Install sequence to specifically initiate a rollback if a certain rule or condition is met. For more information, see Trigger Rollback Action.

About Rollbacks and Custom Code

Note that 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 sequence and add a rule to execute it only when the `$IA_ROLLBACK$` variable is set to `TRUE`.

Building and Testing Installers

Once you have configured the install sets, features, components, files, actions, and other elements of your installation project, you are ready to build and test an installer.

- Working with Build Configurations
- Defining Build Targets
- Specifying Build Distribution Options
- Building Installers Using Build Configurations
- Using build.exe to Build Installers from the Command Line
- Using an Ant Task to Build Installers from the Command Line
- Testing Installers

Working with Build Configurations

In some scenarios, 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 define different build configurations in the same project in InstallAnywhere. Each build configuration defines how InstallAnywhere builds an installer for a particular set of platforms, build distributions, JVMs, locales, and other settings. You can store as many build configurations with a project as necessary.

When you are building your InstallAnywhere project, you have the option of building 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, InstallAnywhere generates a separate installer, each in its own directory.

To create and modify build configurations, use the Build Installers view on the Build page of the Advanced Designer. In this view, you can add, rename, copy, or delete a build configuration. This section of the documentation explains how.

From the Default Locale for Targets with Unsupported Languages list, select the language to use if the target machine's operating system uses a language that is not supported by the installer. If the installer is started on a platform that uses an unsupported language, then the selected language will be used.



Note ▪ *For more information on the Default Locale for Targets with Unsupported Languages option, see Locales Subtab.*

Creating a New Build Configuration

You can create multiple build configurations in the same InstallAnywhere project, each with different settings. Each build configuration defines how InstallAnywhere builds an installer for a particular set of platforms, build distributions, JVMs, locales, and other settings. You can store as many build configurations with a project as necessary.



Task ***To add a new build configuration to a project:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. Click the **Add Build Configuration** button. 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 fewer, and it must not contain any of the following characters: asterisk (*), question mark (?), period (.), backslash (\), or forward slash (/).

InstallAnywhere adds the new build configuration and selects it in the Select Build Configuration list. Configure the build configuration's settings as needed. To learn more, see:

- Defining Build Targets
- Specifying Build Distribution Options
- Using Tags to Customize Build Configurations
- Specifying Locale Settings

Renaming a Build Configuration

InstallAnywhere lets you change the name of build configurations in your projects.



Task ***To rename a build configuration:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration that you want to rename.
4. Click the **Rename Build Configuration** button. The **Rename Build Configuration** dialog box opens.
5. Enter a new name for this build configuration and click **OK**.

   The name must be 60 characters or fewer, and it must not contain any of the following characters: asterisk (*), question mark (?), period (.), backslash (\), or forward slash (/).

InstallAnywhere updates the name of the build configuration.

Copying a Build Configuration

InstallAnywhere provides the ability to copy a build configuration in a project. Copying creates a new build configuration with the same settings as the original build configuration. You can modify the copied build configuration as needed without modifying the original build configuration.

You may want to create one build configuration with the appropriate default settings, and to use it as a template to create additional build configurations that have slight changes from the default values.



Task ***To copy a build configuration:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration that you want to copy.
4. Click the **Copy Build Configuration** button. The **Copy Configuration** dialog box opens.
5. Enter a name for the new build configuration and click **OK**.

   The name must be 60 characters or fewer, and it must not contain any of the following characters: asterisk (*), question mark (?), period (.), backslash (\), or forward slash (/).

InstallAnywhere adds the new build configuration and selects it in the Select Build Configuration list. Configure the build configuration's settings as needed. To learn more, see:

- Defining Build Targets
- Specifying Build Distribution Options
- Using Tags to Customize Build Configurations
- Specifying Locale Settings

Removing a Build Configuration

If you are no longer using a particular build configuration in your project, you may want to delete it from your project.



Task ***To delete a build configuration:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration that you want to delete.
4. Click the **Remove Build Configuration** button. InstallAnywhere displays a message box, prompting you to confirm that you want to delete the selected build configuration.
5. Click **Yes**.

InstallAnywhere removes the build configuration from your project and from the Select Build Configuration list.



Note ▪ *A project must have at least one build configuration. If a project contains only one build configuration, the Remove Build Configuration button on the Build Configurations tab is 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** check box. If a build configuration is added to a project build, that build configuration is built each time that one of the following occurs:

- You click the Build Project or Build All buttons on the Build Configurations tab in the Build Installers view (on the Build page).
- You build the project from the command line without specifying any build configurations, such as:

  ```
  build.exe MyProject.iap_xml
  ```



Task ***To add a build configuration to the project build:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration that you want to add to the build.
4. Select the **Add to project build** check box.



Note ▪ *If you click the Build All button in the Build Appliances view (or use the* `-all` *command-line argument), all existing appliance configurations for that project are built, regardless of whether their* ***Add to project build*** *check box has been selected.*

Using Tags to Customize Build Configurations

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 that you can use to organize project elements. For more information, see Creating New Tags.
- **Step 2: Assign Tags to Project Elements**—Assign an appropriate tag to each project element that you want to include in some build configurations but exclude from others. For more information, 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. For more information, see Associating Tags to Build Configurations.

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.

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 on the Tags subtab.
- **Tagged**—A project element is tagged when it has one or more tags listed in the Associated Tags list on the Tags subtab.

When you build a particular build configuration, InstallAnywhere includes project elements in and excludes project elements from the build output, depending on whether the build configuration has any associated tags, and which (if any) tags are associated with a project element.

The following factors determine whether InstallAnywhere includes a project element in a build:

- If a project element is not associated with any tags, InstallAnywhere includes this project element in the installer for all build configurations.
- If a project element is associated with one or more tags and the build configuration has at least one matching tag, InstallAnywhere includes this project element in the installer.
- If a build configuration is not associated with any tags, InstallAnywhere includes all project elements in the installer.



Important ▪ *When you create a new tag, it is automatically associated with all existing build configurations.*

Creating New Tags

You can create new project tags in the Advanced view of the Project page, or on the Tags subtab of a project element customizer.

You can associate each tag with multiple build configurations.



Task ***To create a new build configuration tag:***

1. Do one of the following:
   - In the Advanced Designer, on the **Project** page, click **Advanced**. The **Advanced** view opens.
   - In the Advanced Designer, open the **Tags** subtab of a project element customizer.
2. In the **Manage Tags** area, click the **Create Tag** button. 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 A**ssociate with all project elements** check box.



***Note** ▪ If you select the **Associate with all project elements** check box 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**.

InstallAnywhere adds the new tag to the list of tags.

Assigning Tags to Project Elements

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 each project element 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 that tag is associated with a build configuration, all of those project elements are built for that build configuration.



***Important** ▪ If a project element is not associated with any tag, that project element is included in all build configurations by default.*

Considerations for Assigning Build Configuration Tags

In a project, features take precedence over components, which take precedence over actions. To avoid conflicts when you are assigning tags to project elements, 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 that is 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, InstallAnywhere displays a warning message on the Tags subtab of that project element’s customizer.

Assigning a Tag to a Project Element



Task ***To assign an existing tag to a project element:***

1. Do one of the following:
   - To associate a tag with a feature or component: In the Advanced Designer, on the **Organization** page, click **Features** or **Components**. The **Features** view or the **Components** view opens.
   - To associate a tag with an action, panel, file, or other project element: In the Advanced Designer, on the **Sequence** page, click the appropriate type of sequence. The appropriate view opens.
2. Select a project element that you want to associate with a tag.
3. In the project element customizer at the bottom of the screen, click the **Tags** subtab. All of the tags defined in this project are listed. Tags that are associated with this project element are listed in the **Associated Tags** list, while the tags that 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 the **Add Tag** button to move it to the **Associated Tags** list.

To remove a tag from the Associated Tags list, select it then click the Remove Tag button.

Associating Tags to Build Configurations

To associate tags with build configurations, use the Tags subtab in the Build Installers view.



Task ***To associate tags to a build configuration:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration to which you want to assign tags.
4. Click the **Tags** subtab. All of the tags that are defined in this project are listed. Tags that are associated with this build configuration are listed in the **Associated Tags** list, while tags that are 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 the **Add Tag** button.

InstallAnywhere moves the tag to the Associated Tags list.

To remove a tag from the Associated Tags list, select it then click the Remove Tag button.

Searching for Tags

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 sequences (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. In the Advanced Designer, on the **File** menu, click **Search** (or press CTRL+F). The **Search Results** dialog box opens.
2. Click the **Tags** tab.
3. In the **Select Tag** list, select the build configuration tag that you want to search for.
4. Under **Search for Selected Tag in**, select the check boxes for the sequences and product elements that you want to search.
5. Click the **Search** button.

InstallAnywhere shows which sequences and project elements that are associated with the selected tag.

Importing Tags from Merge Modules

When you statically import a merge module into a project, you have the option of importing tags from the merge module. To import tags from a static merge module, select the Import Tags check box on the InstallAnywhere Merge Module Import Assistant dialog box.

If this check box is selected, InstallAnywhere imports the merge module's associated tags into the parent project. If this check box is cleared, the merge module's associated tags are ignored.

Specifying Locale Settings

InstallAnywhere can generate localized installers when you select the locales that you want to support. The Build Configurations tab in the Build Installers view on the Build page is where you specify the appropriate locale settings for each build configuration in your project.



Task ***To generate multilanguage installers:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration whose locale you want to configure.
4. Click the **Locales** subtab.
5. In the **Locale List**, select the check boxes of the languages that you want the selected build configuration to support.



Note ▪ *Additional localization may be required for resources, customized panels or consoles, and custom code.*

Defining Build Targets

Build targets specify the different installers that InstallAnywhere will build for your project. To define the build targets for your project, use the Build Configurations tab in the Build Installers view on the Build page.



Task ***To define a build target:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration whose build targets you want to define.
4. Click the **Build Targets** subtab.
5. Optionally, to add a new build target, click the **Add Build Target** button. InstallAnywhere adds a new row to the **Customize Build Targets** list; it contains default information for a new build target.
6. In the **Customize Build Targets** list, click the plus sign in the row of the new or existing build target that you want to configure. Configure the build target's settings as needed.

For detailed information on the build target settings, see Build Targets Subtab.

Requirements for UNIX-Based Platforms

The following are required on UNIX-based platforms:

- **Unzip tool**—It is mandatory that all UNIX-based platforms have a valid unzip command/tool installed and be made available on PATH in order to support extraction of installers /jre for the installers to launch. If it is not present, the JVM search settings configured in the InstallAnywhere project will not be respected. For more information, see Unzip Tool Requirement.
- **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.

Unzip Tool Requirement

When running an installer (with a Linux or Solaris build target) on Linux or Solaris, a valid copy of the system unzip command is required for the installer to launch.

If the unzip tool is not present, you need to install the unzip command by either downloading it from http://info-zip.org/ or by running the `apt-get update` or `apt-get install` commands.

InstallAnywhere scripts are written to look for a valid unzip command to be present on UNIX-based platforms. Once the unzip command is installed, it has to be made part of the PATH variable to be available. The InstallAnywhere installer scripts look for the unzip command in the following locations:

```
/usr/bin
/usr/sbin
/usr/local/bin
[directory defined by the PATH variable]
```

If the unzip command is not found in any of those locations, then installers fail to launch.

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:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration whose build targets you want to define.
4. Click the **Build Targets** subtab.
5. In the **Customize Build Targets** list, click the plus sign next to **UNIX_with_VM**. The customizer for **UNIX_with_VM** opens.

6. In the **Target Name/Output Folder** setting, enter the name of the specific UNIX platform for which you want to build.



Tip ▪ *You can use this setting to differentiate between two variations of the same platform target.*

7. In the **VM Search Instructions** area, select the **With VM** check box and select a VM from the **VM to Bundle with Installer** list.
8. Configure other settings as needed.

For detailed information on the build target settings, see Build Targets Subtab.

Specifying 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. The Build Configurations tab in the Build Installers view on the Build page is where you specify the appropriate distribution options.

Creating Web Installers

y.



Task ***To create Web installers:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.InstallAnywhere lets you generate Web installers. A Web installer includes an HTML page and embedded Java applet to make downloading the installer over the web eas
3. In the **Select Build Configuration** list, select the build configuration for which you want to specify distribution options.
4. Click the **Distribution** subtab.
5. In the **Web Installer Options** area, select the **Build Web Installers** check box.
6. Indicate whether 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 the **Optimize Installer Size by Platform** check box.
7. To include the `JRE` file as a `directory` inside the installers for macOS operating systems, select the **Bundle JRE as directory** check box.
8. In the **Web Page Displays In** list, choose the language in which the Web installer pages are rendered.

Creating CD-ROM/DVD Installers

InstallAnywhere enables you to generate CD-ROM/DVD installers. These installers consist of multiple files that are meant to be burned onto one or more CDs, DVDs, or other removable media. They can also be placed on network volumes to provide easy to access to large installers.



Task ***To create CD-ROM/DVD installers:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration for which you want to specify distribution options.
4. Click the **Distribution** subtab.
5. In the **CD-ROM Installer Options** area, select the **Build CD-ROM Installers** check box.
6. Indicate whether to optimize installers for each build target. To produce an installer specific to each build target that exists on the **Build Targets** tab, select the **Optimize Installer Size by Platform** check box.

By default, the installer is 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

The directory structure for CD-ROM installers is as follows:

```
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 Unix (All) and Other Java-Enabled Platforms, the install binary is contained in the `Disk1/InstData` directory. On Unix(All), for example, the directory structure for Disk 1 is:

```
GeneralUnix/Disk1/InstData/
  |-install.bin
  |-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 5-10: Insert New Media Dialog Box

Creating Merge Modules

For each build configuration, you can specify whether you want to create a merge module installer.

Task ***To create merge modules:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration for which you want to specify distribution options.
4. Click the **Distribution** subtab.
5. In the **Merge Module/Template Options** area, select the **Build Merge Module/Template** check box.
6. If you want to produce merge modules that correspond with each build target on the **Build Targets** tab, select the **Optimize Merge Module/Template Size by Platform** check box. If you select this check box, InstallAnywhere creates separate merge modules for each platform. Each contain only the resources that are needed for that specific platform.

   Important ▪ *If the merge modules will be imported into another installer, do not select the Optimize Merge Module/Template Size by Platform check box. Importing a merge module requires a non-optimized merge module.*

7. If you want to create a Merge Module that can be installed but not altered, select the **Read Only** check box.

By selecting the **Read Only** check box, 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. A read-only merge module can be installed only as a self-contained installer unit.



Tip ▪ *To learn how 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, see Advertising Merge Module Installer Variables.*



Important ▪ *The name of the artifacts that are built for merge module projects is determined by the value of the Project Name setting in the General Settings view on the Project page of the Advanced Designer.*

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 you are configuring 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 in the Variables view on the Project page.



Task ***To advertise merge module installer variables:***

1. In the Advanced Designer, on the **Project** page, click **Variables**. The **Variables** view opens.
2. In the **Advertise Variables** area, click the **Advertise Variables** button.The **Advertise Variables** dialog box opens.
3. Select the appropriate tab to indicate the sequence for which you want to advertise variables.

   Note ▪ *Advertised variables are defined by sequence: variables that are advertised in the Pre-Install sequence affect the dynamic merge module in the Pre-Install sequence; the Install Merge Module action uses advertised variables that set in the Install sequence only.*

4. To advertise an input variable so that it is visible in the parent project (the project that is going to install this merge module), click the **Add** button next to the input variables list, and then enter the appropriate information.
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 the **Add** button next to the parent variables list, and then enter the appropriate information.

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 sequences of installation.

In general, $INSTALL_SUCCESS$ and $RESTART_NEEDED$ are two status variables that 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 sequence to the Post-Install sequence (but not from the Install sequence to the Post-Install sequence) for a dynamic merge module.
- **$RESTART_NEEDED$**—The $RESTART_NEEDED$ variable must be advertised in the Post-Install sequence in order for it to be propagated from the merge module to the parent.
- **$IA_ROLLBACK$**—The $IA_ROLLBACK$ variable is passed from merge module to parent automatically.

Specifying How to Show the Progress of a Merge Module Installation



Note ▪ *This information does not apply to nested merge modules.*

The installation progress of merge modules is merged into the main installer's progress information. This enables end users to 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 check box on the Install Merge Module action customizer:

- To display an intermediate progress bar while this merge module is being installed, select this check box.
- To suppress the intermediate merge module progress bar and integrate and the progress of the merge module installation into the progress bar of the parent installer, clear this check box.

Building Installers Using Build Configurations

InstallAnywhere provides multiple ways to build an installer. You can build one or more installers for a project from within InstallAnywhere. You can also build from the command line. This section of the documentation provides more information.

Building a Specific Build Configuration

InstallAnywhere offers the ability to build a particular build configuration on demand.



Task ***To build a specific build configuration in your project:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration that you want to build.
4. Click the **Build Selected** button.

InstallAnywhere builds the selected build configuration.

Building the Build Configurations That Are Enabled for Project Build

InstallAnywhere offers the ability to build all of the build configurations whose **Add to project build** check box is selected.



Task ***To build all build configurations in the project build:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Project** button.

InstallAnywhere builds all of the build configurations that are enabled for the project build.

Building All of the Build Configurations in a Project

InstallAnywhere offers the ability to build all of the build configurations that are defined in a project.



Task ***To build all build configurations:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build All** button.

InstallAnywhere builds all of the build configurations that are defined in the project.

Using build.exe to Build Installers from the Command Line



Note ▪ *For Windows-based development systems, InstallAnywhere provides two versions of the command-line build tool:* `build.exe` *and* `build-as-invoker.exe`*. Using* `build-as-invoker.exe` *is recommended for users who do not have administrative privileges on their Windows-based build system.*

You can use the InstallAnywhere command-line build tool (`build.exe`) to build installers for the build configurations in an InstallAnywhere project.

Syntax

To build through the command line, use the following syntax:

`build.exe` *`Project_File_Path List_of_Build_Configurations`*

The path for the project file (*`Project_File_Path`*) must be specified. The build configuration list is optional.

Command-Line Parameters

The command-line build can be called with many options to override build targets, distribution options, working directory, and more. For details, see Build Command-Line Arguments.

Sample Command-Line Statements

If you do not list build configurations in your command-line statement, all of the build configurations that are defined in the project and that have their **Add to project build** check box selected are built—for example:

```
build.exe C:\MySetups\MyProduct.iap_xml
```

To build all of the build configurations that are defined in a project, you can use the `-all` command-line option—for example:

```
build.exe C:\MySetups\MyProduct.iap_xml -all
```

To build the installers for the build configurations that are named BuildConf1, BuildConf2, and BuildConf3, use the following command line:

```
build.exe C:\MySetups\MyProduct.iap_xml BuildConf1 BuildConf2 BuildConf3
```

You can also include additional parameters in your command-line statements. For example, to add additional platform-related arguments to the aforementioned command line, enter them directly after each build configuration name:

```
build.exe C:\MySetups\MyProduct.iap_xml BuildConf1 +X +L -s BuildConf2 w BuildConf3 a
```

The above statement launches the build of three different build configurations, each with their own platform-related build targets:

- The build configuration named BuildConf1 is built with build targets of OS or OS X without the VM option (+X), as well as Linux without the VM option (+L). The Solaris build target (-s) is not built.
- The build configuration named BuildConf2 is built with a build target of Windows without the VM option.
- The build configuration named BuildConf3 is built with a build target of AIX without the VM option.

Using a BuildProperties.xml File to Specify Build Parameters for Command-Line Builds

You can use a build properties XML file to build installers for a project. To do this, pass the `-p` argument with the path and name of the build properties file to `build.exe`:

```
build.exe C:\MySetups\MyProduct.iap_xml -p C:\Path\BuildPropeties.xml
```

The settings that are specified in the build properties file override the build settings in the project.

InstallAnywhere includes a build properties file template named `BuildProperties.xml`:

IA_HOME`/resource/build/BuildProperties.xml`

This template file provides a sample of all possible build settings; you can use it as a template to meet your build requirements.



Note ▪ *For more information, see BuildProperties.xml File.*

Using a buildproperties.properties File to Specify Build Parameters for Command-Line Builds

You can use a `buildproperties.properties` file to build installers for a project. To do this, pass the `-p` argument with the path and name of the `buildproperties.properties` file to `build.exe`:

```
build.exe C:\MySetups\MyProduct.iap_xml -p C:\Path\buildproperties.properties
```

The settings that are specified in the .properties file override the build settings in the project.

InstallAnywhere includes a sample `buildproperties.properties` file:

IA_HOME`/resource/build/buildproperties.properties`



Note ▪ *For more information, see buildproperties.properties File.*

Determining the Host ID of the Machine Through the Command Line

If the build machine is not connected to a network (but not necessarily to the Internet), Java's API `java.net.InetAddress` is not able to obtain the host ID of the machine. Therefore, the `-generateHostID` argument does not provide the host ID. The following instructions explain how to obtain the host ID in this scenario.

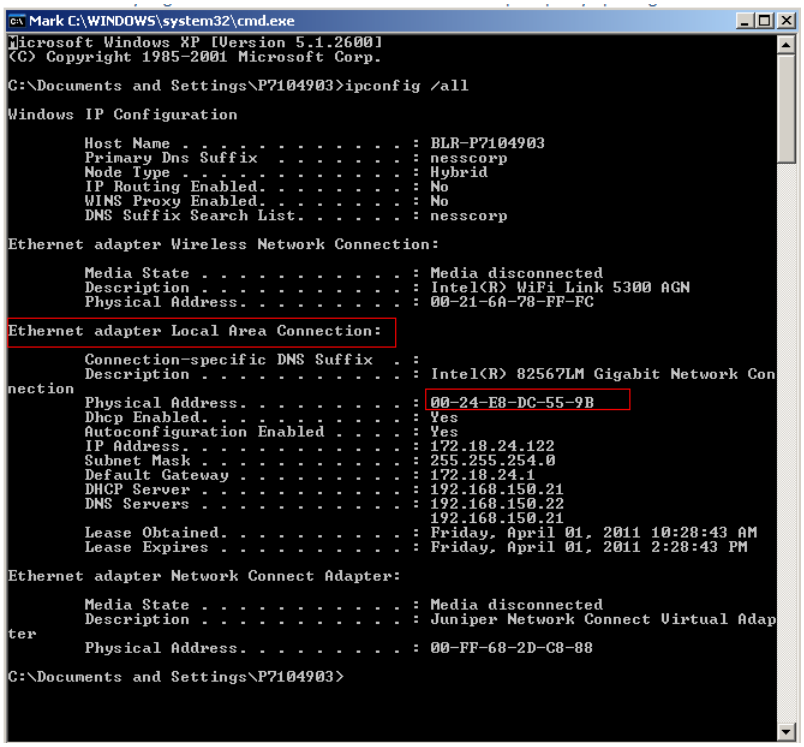


Task ***To obtain the host ID of a machine that is not connected to a network:***

1. Open a Command Prompt window.
2. At the command prompt, enter the following command:

   `ipconfig /all`

   Configuration information is displayed in the Command Prompt window.



3. In the Ethernet adapter Local Area Connection section, look at the Physical Address value.

This physical address value (for example, **00-24-E8-DC-55-9B**) is the host ID of your machine. Use this value (without the dashes, and with all of the uppercase letters converted to lowercase letters—for example, **0024e8dc559b**) when you are obtaining a license file.

Using an Ant Task to Build Installers from the Command Line

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 though *tasks*, which can be part of the core Ant distribution or available as extensions.

InstallAnywhere includes an Ant task to build installers from Ant. The InstallAnywhere Ant task (`iaant.jar`) is located in your InstallAnywhere application folder:

IA_HOME`\resource\build\iaant.jar`



Tip ▪ *To integrate the InstallAnywhere Ant task in an Ant project, 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 later.*

Ant uses an XML file to specify the order of tasks for your build process. For more information on Ant, visit the Apache Foundation's Ant Project Web site (http://ant.apache.org).

The following sections explain how to use the InstallAnywhere Ant task for your InstallAnywhere project.

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="IA_HOME/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"
        i5OSLogin="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"
            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 in the `BuildProperties.xml` file.

Task Parameters

For a list of the supported Ant task parameters, see InstallAnywhere Ant Task Reference.

Sample Ant Tasks

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"
  IAProjectFile="C:\Projects\myproject.iap_xml"
  BuildLinuxWithoutVM="true"
  BuildWindowsWithoutVM="true"
  BuildWebInstallers="true"
  OptimizeWebInstallers="true"
  InstallerStdErrRedirect="C:\console.txt"
  />
  <configuration name="BuildConfig">
    <locales>
        <localeSuffix>en</localeSuffix>
        <localeSuffix>de</localeSuffix>
        <localeSuffix>ja</localeSuffix>
    </locales>
    <target platform="osx" buildWithNoVM="true"/>
  </configuration>
<buildinstaller/>
```

The following samples demonstrate how to use an InstallAnywhere Ant task to configure and build installers.

Using an External File to Specify Build Information

If you want to pass build properties for your Ant task in a single external file, you can use an external `BuildProperties.xml` file or an external `buildproperties.properties` file and the `-p` command-line argument. For example:

```
<buildinstaller
    IALocation="IA_HOME"
    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 included at build time. You can use an environment variable to specify the source path's value at build time.

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 that is being referenced.

The following Ant target example uses the `<env>` element to set the source path `IA_PATH_BILLBOARDS_PATH` to the path `X:\billboards\BillboardProblem`:

```
<env key="IA_PATH_BILLBOARDS_PATH" value="X:\billboards\BillboardProblem" />
```

All references to the `IA_PATH_BILLBOARDS_PATH` source path in the project are substituted with this path at build time.

```
<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="IA_PATH_BILLBOARDS_PATH" value="X:\billboards\BillboardProblem" />
    </buildinstaller>
</target>
```

Specifying Build-Time Variables in a Properties File

In an Ant task, you can specify a property file that contains build-time variables.

```
<project default="test">
    <taskdef name="buildinstaller" classname="com.zerog.ia.integration.ant.InstallAnywhereAntTask">
        <classpath>
            <pathelement path="IA_HOME\\resource\\build\\iaant.jar" />
        </classpath>
    </taskdef>
    <target name="test">
        <buildinstaller IALocation="IA_HOME" IAProjectFile="ProjectFilePath"
            buildtimevarpropfile="PathToPropertiesFileContainingBuildTimeVariables">
            <configuration name="Default_Configuration">
                <target platform="Windows" outputDir="OutputDirectory" buildWithVM="false"
                    buildWithNoVM="true" />
            </configuration>
        </buildinstaller>
```

```
    </target>
</project>
```

Testing Installers

Testing is an essential part of creating an installer. After the build process is complete, you can test the installer by clicking either of the following buttons in the Build Installers view on the Build page in the Advanced Designer.

- **Try Installer**—To run the default installer for your operating system, click this button.

  ***Note** ▪ If two or more installers are built for your operating system, InstallAnywhere shows a control that lists the installers that you can try.*

- **Try Web Install**—To test the installation by launching a Web browser and the Web install page that InstallAnywhere generates at build time, click this button.



***Note** ▪ To assist in troubleshooting when running an installer on a Windows platform, it is possible to display debug output in a console window. To display debug output during installation, hold down the Control (Ctrl) key when launching the installer. This functionality is available only if you are using a Windows-based LaunchAnywhere setup launcher or the Try Installer button in the Build Installers view.*

Multiple Target Installers

If, after the project has been built, more than one supported target installer is available for the authoring platform, InstallAnywhere displays a selection list between the Build All and Try Installer buttons. If there are none or only one supported target installer, InstallAnywhere does not display this list.

The name that InstallAnywhere displays for each target installer uses one of the following formats:

- **Without VM**— `OutputFolderName_BuildConfigurationName`
- **With VM**—`OutputFolderName_BuildConfigurationName - BundledVMName`

Testing Installers for Other Platforms

If you are building installers for platforms other than the one on which the installer is being developed, transfer that installer to the supported target systems, and run those installers manually. By default, InstallAnywhere creates the installers in the Build_Output directory in the same folder as the `.iap_xml` project file.

The Build_Output directory contains a subfolder for each build configuration in the project. Each build configuration directory in the Build_Output folder contains a Web_Installers folder, a CDROM_installers folder, or both. These folders contain subfolders for the various platforms that your project supports. For the CDROM installer, transfer the entire contents of the `CDROM_Installers` folder.

Working with JRE VM Packs

InstallAnywhere lets you bundle JRE VM packs with your installer to provide a Java virtual machine on target systems.

A JRE VM pack is the InstallAnywhere term for a compressed ZIP archive that contains the files and directories that constitute a platform-specific Java Runtime Environment (JRE), which includes the Java Virtual Machine (JVM) that runs an InstallAnywhere-generated installer. InstallAnywhere recognizes the `.vm` file extension for a JRE VM pack, but the JRE VM pack is still just a compressed ZIP archive. A JRE VM pack must include at least a valid JRE, but it can also include a full Java Development Kit (JDK), which implicitly includes a JRE.

Many JRE VM packs are available for download on the InstallAnywhere Web site.

https://www.revenera.com/install/products/installanywhere/installanywhere-files-utilities.html

If none of the JRE VM packs that are available on the Web site meet your requirements, you can use the Create JRE VM Pack Wizard to create your own. This wizard enables you to package existing JREs and JDKs as VM packs; InstallAnywhere uses your custom VM packs for bundling with your project's installers. For more information, see Creating JRE VM Packs.

- Downloading JRE VM Packs for Bundling in Installers
- Adding JRE VM Packs for Your Installers to Your Development and Build Machines
- Adding a VM Pack to Your Project



Note ▪ *For more advanced information about JRE VM packs and Java virtual machines, see Adding Advanced Functionality to Installers.*

Downloading JRE VM Packs for Bundling in Installers

When you are configuring each build target in your project, you can specify the Java virtual machine that you want to bundle with the target's installer. You can download additional VM packs from the InstallAnywhere Web site.



Task ***To download a VM pack:***

1. In the Advanced Designer, on the **Help** menu, click **Download Additional VM Packs**. InstallAnywhere opens the VM pack download page in your system's default Web browser:

   https://www.revenera.com/install/products/installanywhere/installanywhere-files-utilities.html

2. Locate the VM pack that you want to download and save it to disk.

In order for the VM packs that you download to be available for selection in the Build Installers view in InstallAnywhere, you must place them an appropriate location such as the following, or in another valid VM pack location:

```
$IA_HOME$/resource/installer_vms
```

For more information, see Adding JRE VM Packs for Your Installers to Your Development and Build Machines.

Adding JRE VM Packs for Your Installers to Your Development and Build Machines

The JRE VM packs that you download or create are not available for selection in the Build Installers view on the Build page until you add them to the `installer_vms` directory (commonly, `$IA_HOME$/resource/installer_vms`) or another valid JRE VM path that you have defined in the InstallAnywhere Preferences dialog box.



Task ***To add a JRE VM pack to the appropriate location on your development or build machine:***

1. Copy the JRE VM pack to a valid JRE VM pack location. The default location for JRE VM packs is:

   `<InstallAnywhere>\resource\installer_vms\`

   Note ▪ *<InstallAnywhere> does not refer to the project directory containing the InstallAnywhere project (.iap_xml) file. This refers to the location where InstallAnywhere was installed on the build machine.*

2. Exit and restart InstallAnywhere.

The JRE VM packs that you added are now available in the VM to Bundle with Installer lists on the Build Targets subtab, which is available on the Build Configurations tab in the Build Installers view. InstallAnywhere refreshes this list each time that you edit preferences or start InstallAnywhere. Therefore, you cannot simply copy a JRE VM pack into a valid VM pack path. You must first open and close the Preferences dialog box or restart InstallAnywhere.

Adding a VM Pack to Your Project

You can add a VM pack to your project in InstallAnywhere to have it bundled with your installers. Note that not all build targets support VM packs.



Task ***To bundle a VM pack:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Targets** tab.
3. Expand the build target for which you want to bundle a VM, and select its **With VM** check box.
4. In the **VM to Bundle with Installer** list, select the VM that you want to bundle with the build target.

Targeting 32-Bit and 64-Bit Windows-Based Systems

Microsoft designed 64-bit versions of Windows to allow existing 32-bit applications to work seamlessly on 64-bit versions of Windows. They also designed 64-bit versions of Windows in such a way to allow a recompiled version of the same code to work seamlessly as a 64-bit application.

In some cases, such as installation, what is normally a beneficial separation becomes a challenge. Typically installers are 32-bit applications themselves (in order to run on 32-bit machines), and accessing 64-bit locations to install a 64-bit application is more complex than a standard file copy or registry write. InstallAnywhere has support that lets you create installers that target both 32-bit and 64-bit versions of Windows.

Challenges of Supporting Both 32-Bit and 64-Bit Windows-Based Systems

32-Bit vs. 64-Bit File Locations

To provide support for allowing existing 32-bit applications to work on 64-bit versions of Windows, most 64-bit versions of Windows use 32-bit Windows-on-Windows (WOW64) emulation. This WOW64 emulation support isolates the 32-bit and 64-bit files from each other by storing their files in separate locations.

A 64-bit target system typically has two Program Files folders:

- Program Files, which is for 64-bit applications
- Program Files (x86), which is for 32-bit applications

A 64-bit target system typically has two Common Files folders (one in either Program Files folder):

- Program Files\Common Files, which is for 64-bit applications
- Program Files (x86)\Common Files, which is for 32-bit applications

A 64-bit target system also typically has two system folders:

- System32, which is for 64-bit libraries and executable files
- SysWOW64, which is for 32-bit libraries and executable files

Thus, if an end user runs a 32-bit installer on a 64-bit version of Windows that uses WOW64 emulation, files that you may want to be installed to 64-bit file locations may be redirected to 32-bit file locations (such as Program Files x86). However, on 32-bit systems, such files are installed to Program Files.

32-Bit vs. 64-Bit Registry Locations

The isolation of 32-bit and 64-bit data from each other on 64-bit systems also occurs in the registry. A 64-bit target system typically has two HKEY_LOCAL_HINE\Software keys:

- HKLM\Software, which is for 64-bit applications
- HKLM\Software\Wow6432Node, which is for 32-bit applications

Thus, if an end user runs a 32-bit installer on a 64-bit version of Windows that uses WOW64 emulation, registry data that you may want to be installed to 64-bit HKLM\Software locations may be redirected to HKLM\Software\Wow6432Node. On 32-bit systems, such registry data are installed to HKLM\Software.



Tip ▪ *To see how a 32-bit application views the registry on a 64-bit version of Windows, launch the 32-bit version of the Registry Editor (the `regedit.exe` file in the SysWOW64 folder).*

Pure 64-Bit Target Systems

Some 64-bit versions of Windows—such as Windows Server Core systems—may not have WOW64 support. These 64-bit systems cannot run 32-bit installers.

32-Bit Installer Support in InstallAnywhere for 32-Bit and Most 64-Bit Windows-Based Target Systems

In some scenarios, if you are creating 32-bit installers for 32-bit and 64-bit versions of Windows, you may want WOW64 emulation to be enabled for your 32-bit installers. In other scenarios, you may want WOW64 emulation to be disabled for your 32-bit installers.

To learn more, see Enabling or Disabling WOW64 Emulation on 64-Bit Windows-Based Target Systems.

Pure 64-Bit Installer Support in InstallAnywhere for 64-Bit Windows-Based Target Systems

In some other scenarios, you may want to create pure 64-bit installers that run on 64-bit versions of Windows that may or may not have WOW64 support.

For more information, see Creating Pure 64-Bit Installers for 64-Bit Windows-Based Target Systems.

Enabling or Disabling WOW64 Emulation on 64-Bit Windows-Based Target Systems

If 32-bit Windows-on-Windows (WOW64) emulation is enabled during installation or uninstallation on a 64-bit version of Windows, problems may occur. For example, operations for a 64-bit magic folder may be redirected automatically to 32-bit locations; this may cause 64-bit applications to be installed incorrectly on 64-bit systems.

However, if WOW64 emulation is disabled during installation or uninstallation, other issues can occur. For example, custom code that calls InstallAnywhere Windows Service APIs may not be redirected as expected.

Therefore, InstallAnywhere gives you control over when you want WOW64 emulation to be enabled and when you want it to be disabled.

Using the Advanced Designer to Enable or Disable WOW64 Emulation



Task ***To specify whether to enable or disable WOW64 emulation:***

1. In the Advanced Designer, on the **Project** page, click **Advanced**. The **Advanced** view opens.
2. In the **Windows WOW64 Emulator Settings** area, select the check boxes for the elements that should allow WOW64 redirection. Clear the check boxes for the elements that should disable WOW64 redirection.

To learn more about the different elements for which you can enable or disable WOW64 redirection, see Windows WOW64 Emulator Settings.

Calling InstallAnywhere APIs in Custom Code to Enable or Disable WOW64 Emulation

InstallAnywhere's APIs (`IAClasses.zip`) include a class called WOW64 in `com.zerog.ia.platform`. You can use the following APIs with this class to enable and disable WOW64 redirection.

```
WOW64.getInstance().enableRedirection();
WOW64.getInstance().disableRedirection();
```

For example, you can disable WOW64 emulation for the Install sequence but use custom code to enable it and then disable it again in specific situations.



Important ▪ *If you change the behavior for WOW64 redirection through custom code, ensure that you change it back through custom code once the override is no longer required. If you do not, the same WOW64 settings carry forward for the rest of the session.*

Creating Pure 64-Bit Installers for 64-Bit Windows-Based Target Systems

Windows Server Core supports disabling 32-bit Windows-on-Windows (WOW64) support. As this configuration becomes more popular, you may want to ensure that your 64-bit applications can install without any reliance on 32-bit functionality. To make this possible, InstallAnywhere enables you to build pure 64-bit installers for Windows-based target systems; these installers can be run on 64-bit Windows-based systems that do not have WOW64 functionality as well as those that do have WOW64 functionality. These installers contain 64-bit self-extractors, LaunchAnywheres, and VM packs, as needed. They can install to 64-bit file and registry locations and install 64-bit Windows services as required. Note that these installers cannot be run on 32-bit Windows-based systems.



Important ▪ *If you create a pure 64-bit installer, ensure that it includes the necessary 64-bit binaries for your product; 32-bit binaries cannot be loaded on 64-bit target systems without WOW64 support.*

Using the Advanced Designer to Configure a Pure 64-Bit Installer



Task ***To configure build settings for building a pure 64-bit installer:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the build configuration whose build targets you want to configure.
4. Click the **Build Targets** subtab.
5. In the list of build targets, find the built-in **Windows_Pure_64_Bit** build target.

   If one does not exist, you can add one:

   a. Click the **Add Build Target** button. InstallAnywhere adds a new build target row to the **Build Targets** subtab.

   b. In the **Platform Type** list, select **Windows_Pure_64_Bit**.
6. Configure the other settings for the pure 64-bit Windows build target as needed. To learn more, see Build Targets Subtab.

Using Other Methods for Building a Pure 64-Bit Installer

The command-line build tool `build.exe` has support for building pure 64-bit installers. To learn more, see Build Command-Line Arguments.

You can configure pure 64-bit settings through the build properties XML file. For details, see Supported Properties in the BuildProperties.xml File.

As an alternative, you can configure pure 64-bit settings through a .properties file. For more information, see Supported Properties in the buildproperties.properties File.

The InstallAnywhere Ant task has support for configuring pure 64-bit installers. For the supported parameters, see InstallAnywhere Ant Task Reference.

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 sequence. This action creates a platform-built-in executable file for launching a Java application (also known as a LAX executable file). This LAX allows Java applications to be launched as if they were platform-built-in. For example, end users can launch the LaunchAnywhere executable file by double-clicking it or by calling it directly from the command line.

For Windows-based systems, InstallAnywhere appends .exe to the file name. For OS or OS X–based systems, InstallAnywhere creates an .app package.

A LaunchAnywhere executable file 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.

For information on the settings that you can configure for a Create LaunchAnywhere for Java Application action, see Create LaunchAnywhere for Java Application Action.

Run-Time Behavior for LaunchAnywhere Files on OS and OS X–Based Systems

Note the following behavior that occurs at run time if your OS or OS X–based installer does not include a bundled VM and an end user uses the LAX_VM parameter for LaunchAnywhere to override the JRE that the installer or uninstaller uses:

- If the end user uses LAX_VM for installation and uninstallation, both the installer and the uninstaller use whatever JVM was passed with LAX_VM.
- If the end user uses LAX_VM for installation but does not use it for uninstallation, both the installer and the uninstaller use whatever JVM was passed with LAX_VM for the installation. If the specified JVM is not found, the system JVM is used.
- If the end user does not use LAX_VM for the installation but does use LAX_VM for uninstallation, the installer uses the system JVM, and the uninstaller uses whatever JVM was passed with LAX_VM.

Note the following behavior that occurs if your OS or OS X–based installer includes a bundled VM:

- If the end user uses LAX_VM for installation and uninstallation, both the installer and the uninstaller use whatever JVM was passed with LAX_VM.

- If the end user uses LAX_VM for installation but does not use it for uninstallation, the installer uses whatever JVM was passed with LAX_VM for the installation. The uninstaller uses the bundled JVM.
- If the end user does not use LAX_VM for the installation but does use LAX_VM for uninstallation, the installer uses the bundled JVM, and the uninstaller uses whatever JVM was passed with LAX_VM.



Note *▪ If the Windows installer or uninstaller is launched with administrative privileges and the LAX_VM parameter points to a JVM located in an unsecured folder, the process will exit. JVMs found in unsecured folders are not permitted for use and are restricted by the launcher with a security-related error message. This behavior can be turned off by using one of the following methods:*

- *Configure the `designer.jvm.securefolder.check` property to `false` in the `com.zerog.ia.Designer.properties` file.*
- *Configure the **Secure JVM Folder Check** setting to **Disable** in the **Project** > **Platforms** >**Windows** area.*
- *Run the installer or uninstaller using the `<installer/uninstaller_name> -Djvm.securefolder.check=<false>` command line, where `<installer/uninstaller_name>` refers to your installer or uninstaller executable.*

Launching Additional Installers

The Execute Target File action enables you to launch an installer from within an installer. Use this action when you need to launch another installer that is not an InstallAnywhere-generated installer. You can also use merge modules to launch other InstallAnywhere-generated installers for maximum optimization and performance.

Working with Source Code Control Software

InstallAnywhere enables you to manage different versions of your project files in source code control software such as ClearCase, Perforce, and CVS. You can also run InstallAnywhere builds without checking out your source files out of your source code control system. InstallAnywhere writes temporary files to the system's temp folder, to your home\InstallAnywhere folder, and to the folder where the InstallAnywhere project file is located. You can optionally place VM packs and plug-ins in your home\InstallAnywhere folder on Windows- or UNIX-based systems. This is helpful if you change or update your VM packs and plug-ins often, but you do not want to check them into source code control.

6

Adding Advanced Functionality to Installers

This section explains how to add advanced functionality to an installer:

Table 6-1 ▪ Adding Advanced Functionality to Installers (Sheet 1 of 3)

Topics	Description
Working with Variables	Includes information on setting InstallAnywhere variables, checking variable values, and encrypting variable values in your installers.
Creating JRE VM Packs	Describes how to create your own JVM packs for InstallAnywhere. (InstallAnywhere provides many prebuilt JVM packs on the Web.)
Controlling the JVM That Your Installers Use	Explains how to customize the criteria that are used for the JVM search that is performed by the installer's launcher. (This determines the JVM under which the installer runs.)
Controlling the Installation of Bundled JVMs	Explains how to control when a JVM that is bundled with your installer is installed and where to install it.
Controlling the JVM that Your Launchers Use	Explains how to use project-level settings for the installer's JVM search as well as launcher-specific settings that can leverage, or operate independently from, the installer's JVM 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 that your end users may use to select a VM for the software that your installer deploys.
JVM Spec Files	Describes the format of JVM spec files—which are JVM search instructions that indicate how to search for an appropriate JVM under which the installer runs—and how to customize and write your own JVM spec files.

Table 6-1 ▪ Adding Advanced Functionality to Installers (cont.) (Sheet 2 of 3)

Topics	Description
Generating Response Files	Explains how to generate response files for use with silent installers.
Getting User Input at Run Time	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 that you can use with InstallAnywhere-generated installers and uninstallers.
Setting Project Version at Build Time	Provides several methods that 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.
Preparing Your Installer for Update Notifications	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-generated 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.
Digitally Signing Windows-Based Installers	Describes how to specify the required information for digitally signing your Windows-based installer at build time.
About Authentication and Code-Signing Support for OS or OS X–Based Installers	Describes how to code sign your OS or OS X–based installers and optionally include authentication support.

Table 6-1 ▪ Adding Advanced Functionality to Installers (cont.) (Sheet 3 of 3)

Topics	Description
Requiring Elevated Privileges for OS or OS X–Based Installers and UninstallersIf you want your OS or OS X–based installers and uninstallers to install files to and remove files from locations where write permissions are restricted for standard users, you can configure your project to require authentication. When authentication is required and standard users who are not root users or administrative users with adequate privileges try to launch your installer or uninstaller, they are prompted to enter an administrator name and password in order to proceed.	Explains how to configure your project to specify that elevated privileges are required for your OS or OS X–based installers and uninstallers.
Determining Whether an Installation Was Successful	Describes how to use the `$INSTALL_SUCCESS$` variable to determine the result of an install process.
Troubleshooting Issues with Installers	Provides information on debugging installers.

Working with Variables

InstallAnywhere variables enable you to control the flow of information and the flow of the installation. Variables let you store information from the system and information input by end users, and create rules to determine operations based on that information. You can even have your installer write that information to configuration files or other resources for the application

In the Advanced Designer, you can set variable values, check the values of existing variables, and define which variables must be encrypted.

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



Task ***To add a Set InstallAnywhere Variable - Single Variable action:***

1. In the Advanced Designer, on the **Sequence** page, click the appropriate sequence: **Pre-Install**, **Post-Install**, **Pre-Uninstall**, or **Post-Uninstall**. The corresponding view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. Click the **Set InstallAnywhere Variable - Single Variable** action, and then click the **Add** button. InstallAnywhere adds the action to the list and displays its actions in the customizer at the bottom of the view.
4. Configure the action's settings as needed:
    a. In the **Variable Name** box, enter a name that identifies this variable.
    b. In the **Set Value To** box, enter the value that you want to set for this variable.
    c. To set this variable to evaluate at assignment, select the **Evaluate any variables at assignment** check box.
    d. To avoid the substitution of unknown variables for this action by instructing InstallAnywhere to resolve only InstallAnywhere variables that are listed in the Variables view on the Project page of the project (the known variables), select the **Do not substitute unknown variables** check box. For more information, see Preventing the Substitution of Unknown Variables.

Adding a Set InstallAnywhere Variable - Multiple Variables Action



Task ***To add a Set InstallAnywhere Variable - Multiple Variables action:***

1. In the Advanced Designer, on the **Sequence** page, click the appropriate sequence: **Pre-Install**, **Post-Install**, **Pre-Uninstall**, or **Post-Uninstall**. The corresponding view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. Click the **Set InstallAnywhere Variable - Multiple Variable** action, and then click the **Add** button. InstallAnywhere adds the action to the list and displays its actions in the customizer at the bottom of the view.
4. Click the **Edit Variables** button. The **Edit Multiple Variables** dialog box opens.
5. Add a new variable:
    a. Click the **Add** button. InstallAnywhere adds a blank row to the list of variables.
    b. In the **Variable Name** column, enter a name for the variable.
    c. In the **Value** column, enter the value you want to set for this variable.
    d. To set this variable to evaluate at assignment, select the **Evaluate any variables at Assignment** check box.
6. To add additional variables, repeat step 5.

7. When you have finished adding variables, click **OK**. The **Edit Multiple Variables** dialog box closes, and InstallAnywhere adds the variables that you added to the variables list.
8. To avoid the substitution of unknown variables for this action by instructing InstallAnywhere to resolve only InstallAnywhere variables that are listed in the Variables view on the Project page of the project (the known variables), select the **Do not substitute unknown variables** check box. For more information, see Preventing the Substitution of Unknown Variables.

Preventing the Substitution of Unknown Variables

If you use any of the following actions to set variables in your installation project, a **Do not substitute unknown variables** check box is available:

- Set InstallAnywhere Variable - Multiple Variables
- Set InstallAnywhere Variable - Single Variable
- Execute Command
- Execute Script/Batch File
- Execute Target File
- Modify Text File - In Archive
- Modify Text File - Multiple Files
- Modify Text File - Single File

If your installation project includes a nested variable that could possibly contain multiple $ characters, this could result in the creation of an unknown variable. Therefore, select the **Do not substitute unknown variables** check box to instruct InstallAnywhere to resolve only InstallAnywhere variables that are listed in the Variables view on the Project page of the project (the known variables).

The following table provides a scenario where it would be beneficial to select this check box.

Table 6-2 ▪ 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** check box is selected, the `$COMMAND_LINE$` variable is set correctly to:
		`-u te$t -p pa$sword`
		However, if the **Do not substitute unknown variables** check box is cleared, 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 installation 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. You 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. An InstallAnywhere-generated installer 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 ▪ *To specify the type of encryption that you want your installers to use, use the Security area in the Variables view on the Project page. Using JCE encryption, along with a FIPS-compliant JCE library and supported algorithm, you can ensure that your installers are FIPS 140-2 compliant. For more information about the encryption options that InstallAnywhere provides, see Variables View.*



Task ***To encrypt a variable value:***

1. In the Advanced Designer, on the **Project** page, click **Variables**. The **Variables** view opens.
2. In the **Configure Variables** area, click the **Configure** button. The **Configure Variables** dialog box opens.
3. Click the **Add** button. InstallAnywhere adds a new row to the list of variables.
4. In the **Variable Name** column of the new row, enter the variable name for the variable value you want to encrypt.
5. In the corresponding **Options** column, select **Encrypt Variable Value**.
6. Click **OK**.



Note ▪ *To remove encryption from a variable in your project, select that variable in the Configure Variables dialog box and click Remove.*

Resolving Variables at Build Time

You can configure your project to use build-time variables, which are variables that have their values set at build time.

Instead of using the dollar symbol ($) as the variable delimiter (which is used for standard variables, such as **var**), use the at symbol (@) as the delimiter for build-time variables, such as **@var@**.



Important ▪ *InstallAnywhere does not have support for resolving build-time variables for any of the properties in the Build Appliances view on the Build page.*

Configuring Build-Time Variables in the Advanced Designer



Task ***To configure build-time variables in the Advanced Designer:***

1. In the Advanced Designer, on the **Project** page, click **Variables**. The **Variables** view opens.
2. Click the **Build Time Variables** button. The **Build Time Variables** dialog box opens.
3. Click the **Edit Variables** button. The **Edit Build Time Variables** dialog box opens.
4. Click the **Add** button. InstallAnywhere adds a new row to the list of variables.
5. In the **Build Time Variable** column of the new row, enter a name for the new variable, such as **@Apple@**.
6. In the corresponding **Value** column, enter the value for the new variable.
7. Click **OK**. The **Edit Build Time Variables** dialog box closes. The new build-time variable is now listed on the **Build Time Variables** dialog box.
8. Click **OK**.

The **Build Time Variables** dialog box closes.

Configuring Build-Time Variables Using a Property File

You can also define build-time variables in a property file (.properties) and specify this property file during the command-line build. This method enables you to update variables just before build by editing the property file.

To specify the properties file that you want to use, pass the full path and file name to `build.exe` through the `-btv` parameter; for example:

```
build.exe "C:\Projects\My_Project.iap_xml" -btv "C:\Properties\mypropertyfile.properties"
```

If the same variable is present in the both the property file and in the Advanced Designer, the variable that is defined in the property file overrides the variable that is defined in the Advanced Designer.

To add a build-time variable named `version` with a value of `8.4.0.1` to a build-time .properties file, add the following line to the .properties file:

```
version=8.4.0.1
```



Note ▪ *When using a .properties file to define Build-Time Variables, make sure not to use the `IA_BTV_ prefix` in the Build-Time Variable name. For example, use `PATH`, not `IA_BTV_PATH`. The `IA_BTV_ prefix` is intended only for setting a Build-Time Variable with an environment variable. Also, when setting a Build-Time Variable to a directory path, make sure to escape a \ backslash with another \ backslash. For example use `PATH=C:\\Users\\JoeSmith` and not `PATH=C:\Users\JoeSmith`.*

Defining Build-Time Variables as Environmental Variables

You can also define environment variables and use them as build-time variables in your project.

To define build-time variables as environmental variables, use the prefix **IA_BTV_** in the name of the environment variables—for example, **IA_BTV_VariableName** (or **@IA_BTV_VariableName@**). InstallAnywhere lists such environment variables in the Build Time Variables dialog box, which opens when you click the Build Time Variables button in the Variables view. This dialog box lists the variable name without the **IA_BTV_** prefix.

Creating JRE VM Packs

A JRE VM pack is the InstallAnywhere term for a compressed ZIP archive that contains the files and directories that constitute a platform-specific Java Runtime Environment (JRE), which includes the Java Virtual Machine (JVM) that runs an InstallAnywhere-generated installer. InstallAnywhere recognizes the .vm file extension for a VM pack, but the VM pack is still just a compressed ZIP archive. A VM pack must include at least a valid JRE, but it can also include a full Java Development Kit (JDK), which implicitly includes a JRE.

Many VM packs are available for download on the InstallAnywhere Web site.

https://www.revenera.com/install/products/installanywhere/installanywhere-files-utilities.html

If none of the VM packs available on the InstallAnywhere Web site meet your requirements, you can create your own VM pack using the Create JRE VM Pack Wizard. This wizard enables you to package existing JREs and JDKs as VM packs that InstallAnywhere can bundle with InstallAnywhere-generated installers.

To learn how to use the Create JRE VM Pack Wizard to create VM packs, review this section of the documentation.

- Using the Create JRE VM Pack Wizard
- Directory Name Requirement for Preparing a JRE VM Pack
- Creating a JRE VM Pack Manually
- JRE VM Pack Structure
- JRE VM Pack Properties
- Making a JRE VM Pack FIPS-Compliant

Using the Create JRE VM Pack Wizard

InstallAnywhere enables you to create a JRE VM pack for any platform.



Note ▪ *You can also use the Create JRE VM Pack Wizard as a standalone tool on a machine where InstallAnywhere is not installed. To learn how, see Running the Create JRE VM Pack Wizard as a Standalone Tool.*



Task ***To create a VM pack using the Create JRE VM Pack Wizard:***

1. In the Advanced Designer, on the **Tools** menu, click **Wizards**, then click **Create JRE VM Pack**. The **Create JRE VM Pack Wizard** opens.
2. Click the **Choose** button next to **Choose directory where VM is installed** and browse to the directory that contains the VM. This directory must be named `jre`. For more information, see Directory Name Requirement for Preparing a JRE VM Pack.
3. Click the **Choose** button next to **Choose destination folder** and browse to the directory where you want to store the new VM pack.
4. In the **Platform** list, select the platform of this new VM pack.

   Note ▪ *If the platform that you select does not match the platform of the selected JRE/JDK, the files that are used to create the VM pack are not cleaned up correctly, and the destination folder will contain extra files that are not part of the VM pack.*
5. In the **Choose a name for VM pack** setting, enter a name to uniquely identify the new VM pack.
6. Click the **Next** button.

InstallAnywhere creates the JRE VM pack.



Task ***To display the VM pack in the InstallAnywhere interface for use:***

1. Close InstallAnywhere.
2. Copy the .vm file into the resource/installer_vms subdirectory of the InstallAnywhere installation directory.
3. Open a project in the InstallAnywhere Advanced Designer.

When you open the Advanced Designer in InstallAnywhere, the added VM pack is now listed in the VM to Bundle with Installer list on the Build Targets tab in the Build Installers view of the Build page.

Directory Name Requirement for Preparing a JRE VM Pack

In order to create a VM Pack using the Create JRE VM Pack Wizard, 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 a directory that 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** setting on the Create JRE VM Pack Wizard.

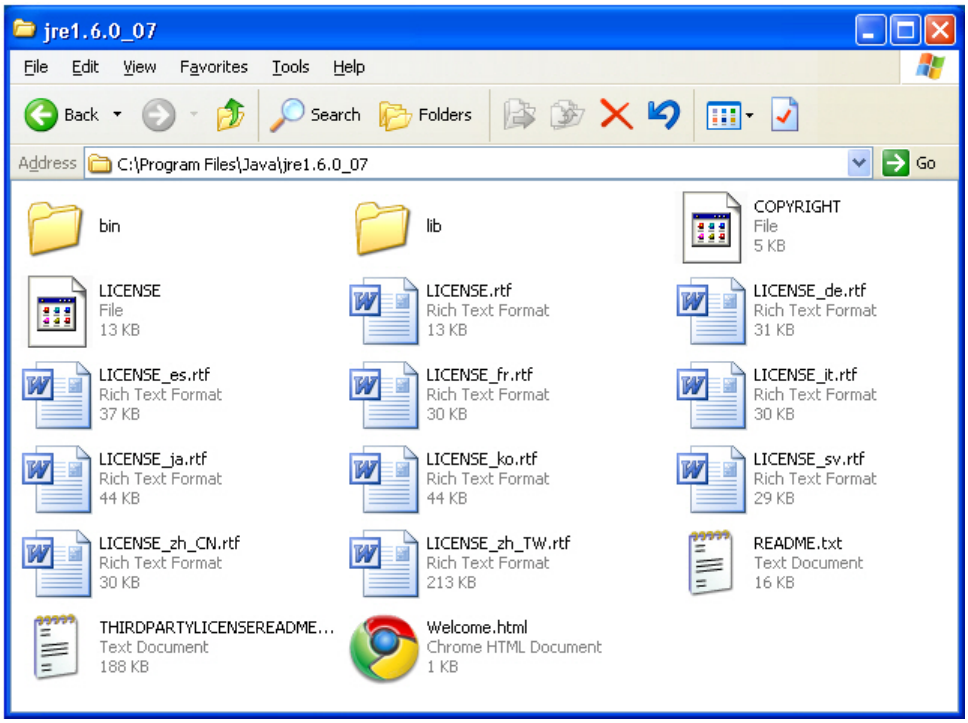


Figure 6-1: VM in Directory Named jre1.6.0_07

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 Create JRE VM Pack Wizard as a Standalone Tool

You can use the Create JRE VM Pack Wizard as a standalone tool on a machine where InstallAnywhere is not installed.

Task ***To run the Create JRE VM Pack Wizard 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. Use the following command line to launch the utility:

```
$ > java -jar Utility.jar
```

The Create JRE VM Pack Wizard opens. Use the wizard to create the JRE VM pack.

Important ▪ *The VM must be located in a directory named* `jre`*. For more information, see Directory Name Requirement for Preparing a JRE VM Pack.*

Creating a JRE VM Pack Manually

As an alternative to using the Create JRE VM Pack Wizard (as described in Using the Create JRE VM Pack Wizard), you can manually create a JRE VM pack.



Task ***To create a JRE VM pack manually:***

1. Create an archive of the JRE directory.
   - For Windows, create a `vm.zip` archive. Do not use compression and do not retain full path info.
   - For OS or OS X, create a `vm.zip` archive. Do not use compression and do not retain full path info.
   - 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. For details, see JRE VM Pack 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
   ```

JRE 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-based, OS or OS X–based, and UNIX-based systems.

Table 6-3 ▪ VM Pack Structure for Different Platforms (Sheet 1 of 2)

Platform	VM Pack Structure

Windows VM Packs	`Windows.vm`
	`  |-vm.properties`
	`  |-vm.zip`
	`  |-bin`
	`  |-java`
	`  |-lib`

OS or OS X VM Packs	`.vm`
	`  |-vm.properties`
	`  |-vm.zip`
	`  |-jre`
	`  |-Contents`
	`  |-home`
	`  |-bin`
	`  |-java`
	`  |-lib`

Table 6-3 ▪ VM Pack Structure for Different Platforms (cont.) (Sheet 2 of 2)

Platform	VM Pack Structure
UNIX VM Packs	unix.vm |-vm.properties |-vm.tar.Z |-jre |-bin |-java |-lib

Support for VM Packs With Nested JRE Folders

In addition to supporting normal VM pack structure, InstallAnywhere also supports VM packs that have nested JRE folders (such as `jre/jre/bin`).

The following table lists the VM pack structure for VM packs that have nested JRE folders.

Table 6-4 ▪ VM Pack Structure for VM Packs with Nested JRE Folders

Platform	VM Pack Structure
Windows VM Packs	Windows.vm |-vm.properties |-vm.zip |-jre |-bin |-java |-lib
UNIX VM Packs	Unix.vm |-vm.properties |-vm.tar.Z |-jre |-jre |-bin |-java |-lib



Important ▪ *Only one level of nesting of JRE folders is supported.*

JRE VM Pack Properties

Every JRE VM pack requires a `vm.properties` file at the root of the JRE VM pack archive.

Table 6-5 ▪ VM Pack Properties (Sheet 1 of 2)

Property	Description
vm.platform	Sets the general platform. Available options are: ● -osx ● unix ● windows
vm.platform.flavor	Identifies the specific platform that the VM supports. Available options are: ● AIX ● HP-UX ● Linux ● solaris ● win32 These values are case-sensitive and must be used exactly as shown. InstallAnywhere uses the vm.platform.flavor value to determine which VMs to list for each platform. If the value that you set does not match one of the expected values, your VM pack will not appear in the VM to Bundle with Installer list on the Build Targets tab in the Build Installers view of the Build page.
vm.name	Records the VM name as it appears in the VM to Bundle with Installer list on the Build Targets tab in the Build Installers view of the Build page. For example: **`SunJRE160_01iWin32.vm`**

Table 6-5 ▪ VM Pack Properties (cont.) (Sheet 2 of 2)

Property	Description

vm.exe.path	Sets the path to the VM pack. Following are several examples of how to set this property for different platforms:
	
	`vm.platform=`**`unix`**
	`vm.platform.flavor=solaris`
	`vm.name=JavaSoft jre1.4 Sparc/Solaris`
	`vm.exe.path=bin/java`
	
	or
	
	`vm.platform=`**`windows`**
	`vm.platform.flavor=win32`
	`vm.name=JavaSoft jre1.4 Win32`
	`vm.exe.path=bin\\java.exe`
	
	or
	
	`vm.platform=`**`-os`**
	`vm.platform.flavor=`
	`vm.name=Sun JRE 1.7.0_25 OSX`
	`vm.exe.path=Contents/Home/jre/bin/java`

Making a JRE VM Pack FIPS-Compliant

Most JRE VM packs are not preconfigured to use FIPS-compliant Java Cryptography Extension (JCE) security providers. To enable JCE encryption and specify an encryption algorithm, use the Variables view on the Project page in the Advanced Designer. Together with a FIPS-compliant JCE library, these settings enable 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 the following:

   ```
   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.IBMJGSSProvider
   security.provider.6=com.ibm.security.cert.IBMCertPath
   ...
   ```

6. Re-create the VM pack archive. For more information, see:

- Using the Create JRE VM Pack Wizard
- Creating a JRE VM Pack Manually



Note ▪ *While modifying the VM pack, you can also add a setting to the* `vm.properties` *file to set the type of algorithm that should be used by default when the Use JCE Encryption check box is selected in the Variables view on the Project page). For example,* `vm.algorithm=DES`.

Controlling the JVM That Your Installers Use

InstallAnywhere lets you customize the criteria that your installer's launcher uses for the JVM search. When the installer's launcher finds a JVM that fits the criteria in the Valid VM List (and no JVM is bundled with the installer), it uses that JVM to run the installer.

By default, the launchers accept version 11 or later of any JVM. However, you can customize the selection criteria for different versions and different types (JDK or JRE).



Tip ▪ *For a detailed discussion of JVM selection criteria settings, see About Java VM Selection Criteria.*



Task ***To customize the installer's VM search:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Installer Settings** tab.
3. In the **Valid VM list** setting, enter your own JVM search criteria. You can optionally change the version, vendor, and type to meet your installer's JVM requirements.



Note ▪ *There is commonly no need to alter these settings for InstallAnywhere-generated 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 that the installer deploys. This does not limit your ability to further customize subsequent VM search criteria or launcher-specific VM settings.*

Controlling the Installation of Bundled JVMs

If you bundled one or more JVMs in your project, you can control when the JVM is deployed and where the installer deploys it.

By default, an installer installs any bundled VM and uninstalls that VM if the end user uninstalls your product. But the VM that you bundle with your installer does not need to be installed in order for it to be used to run the installer. (The installer can extract the JVM to a temporary location during the installation and then delete it when the installation is complete.)

The default location to which InstallAnywhere-generated installers store a bundled VM (when installed) is in a `jre` subdirectory of the product's install location. The location is defined in the VM Install Folder settings on the Installer Settings tab in the JVM Settings view of the Project page. The default location is:

```
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.

Preventing the Installation of the Bundled JVM

InstallAnywhere lets you prevent the installation of a JVM that is bundled with an installer.



Task ***To prevent your bundled JVM from being installed in all cases:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Installer Settings** tab.
3. Clear the **Install Bundled/Downloaded Virtual Machine** check box.



Note ▪ *If you clear the Install Bundled/Downloaded Virtual Machine check box, it will not be possible to allow end 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 JVM When the Installer Includes a Launcher

If your installer includes a LaunchAnywhere launcher, you may want the installer to install the bundled VM.



Task ***To install your bundled VM only when your project includes a LaunchAnywhere launcher:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Installer Settings** tab.
3. Select the **Only when installing a LaunchAnywhere** check box.

Installing the Bundled JVM Only When No Valid JVM Exists

In some cases, you may want the VM that you are bundling 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. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Installer Settings** tab.
3. Select the **Only when a compatible VM is not found in the system** check box.

This approach gives priority to the installer's VM search—the search that the installer performs to find a suitable VM for the launchers that the installer deploys. The criteria for this search depend on how you configure the VM search settings on the subtabs of the Search Panel Settings tab. The Search Panel Settings tab is available in the JVM settings view on the Project page.



Note ▪ *For more information on these settings, see Customizing the VM Search Settings for Your Launchers and Customizing the VM Search Paths and Patterns.*

If the VM search succeeds, the installer does not install the bundled VM. If the search fails to locate a compatible VM, the installer installs the bundled VM.

Preventing the Uninstallation of the Bundled JVM

Typically, if an 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. InstallAnywhere enables you to specify that you want to keep the bundled VM on the end user's system even when the product is uninstalled.



Task ***To prevent your bundled VM from being removed during the uninstallation:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Installer Settings** tab.
3. Select the **Do not remove bundled/downloaded VM when uninstalling** check box.

Changing the Bundled JVM Install Folder

InstallAnywhere lets you override the default VM installation folder of the bundled/downloaded virtual machine.



Task ***To change the VM install folder:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Installer Settings** tab.
3. In the **VM Install Folder** list, select a magic folder.



***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 sequence.*

Changing the Bundled JVM Install Subfolder

InstallAnywhere lets you change the bundled VM installation subfolder for the bundled/downloaded virtual machine.



Task ***To change the subdirectory:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Installer Settings** tab.
3. In the text box next to the **VM Install Folder** setting, 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 that you choose for VM install folder.*

Controlling the JVM that Your Launchers Use

The Java virtual machine that is used by the launcher that your installer deploys depends on the target environment as well as various JVM-related project settings.

Customizing the VM Search Settings for Your Launchers

InstallAnywhere lets you specify how you want your installers to find the valid VMs for your product.



Task ***To specify how you want the installer to search for a valid VM:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **General** subtab.
4. In the **VM Search Settings** area, select the appropriate option:
   - **Use installer's valid VM list**—The installer uses the criteria that the installer's launcher uses in the bootstrap phase. InstallAnywhere displays the currently configured version number next to this setting.
   - **Use the valid VM list of all LaunchAnywhere actions**—The installer combines the lax.nl.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 that require 1.5+ but one that requires SUN 1.6+—the installer's search checks for the vendor SUN and the version 1.5+ for all launchers.

- **Use a specific valid VM list**—The installer uses the criteria that you enter in the box next to this setting.

For more information, see About Java VM Selection Criteria.



***Note** ▪ For information on configuring the* `lax.nl.valid.vm.list` *property for individual launchers, see Customizing Individual Launcher Settings.*

Customizing the VM Search Paths and Patterns

The conventional locations in which InstallAnywhere-generated installers search on Windows-based and UNIX-based systems for VMs are reflected in the default settings on the Windows and UNIX subtabs (Project page > JVM Settings view > Search Panel Settings tab). Likewise, the default Java Executable Patterns settings in these locations reflect the search patterns that InstallAnywhere-generated installers use to identify Java executable files. InstallAnywhere lets you customize both these search paths and the Java executable patterns that 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-based and UNIX-based launchers when the launcher's Advanced Settings (defined for the action on the Sequence page) indicate that it will use either the **VM Selected by the installer or by the end user via Choose VM Panel** option or the **First VM found in the system matching the VM Search Settings defined under Project > JVM Settings** option. However, if a LaunchAnywhere launcher performs an independent JVM search or uses the VM that is used by the installer, these custom search paths and Java executable patterns are not used.*

Customizing Windows Search Paths

InstallAnywhere lets you customize search path settings for installers that target Windows-based systems.

Adding a Search Path for Windows-Based Systems



Task ***To add a search path for Windows-based systems:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **Windows** subtab.
4. Click the **Search Paths** subtab.
5. Do one of the following:
   - To specify an absolute path for the search location:
     a. Click the **Add** button. InstallAnywhere adds a new row to the search paths list.

b. Click the new row and then enter the absolute path to the directory that you want the installer to search for Java executable files.

   Note that you should not use InstallAnywhere variables or magic folders in the path statement.

   By default, new search paths search only the top level of the directory that you specify. No subdirectories are searched.

- To specify a Windows registry search, click the **Add Windows Registry** button. InstallAnywhere adds a new **Windows Registry Search** row to the search paths list.

  The Windows registry search examines standard keys in the Windows registry. It is not possible to customize this type of search.

- To specify a PATH environment variable search, click the **Add Path Environment Variable** button. InstallAnywhere adds a new **Path Environment Variable Search** row to the search paths list.

  The PATH environment variable search scans the top-level of any directories in the PATH environment variable.

Removing a Search Path from Windows Systems



Task ***To remove a search path from Windows systems:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **Windows** subtab.
4. Click the **Search Paths** subtab.
5. Click the search row that you want to remove, and then click the **Remove** button.

Editing an Existing Windows Search Path

You can edit a search path that uses an absolute path.



Task ***To edit an existing Windows search path:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **Windows** subtab.
4. Click the **Search Paths** subtab.
5. Click the search row that you want to edit. This activates the entry and places the cursor at the end of the path statement.
6. Enter your path changes.

Changing the Directory Depth for a Windows Search Path



Task ***To change the directory depth for a Windows search path:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **Windows** subtab.
4. Click the **Search Paths** subtab.
5. In the search paths list, double-click the level that you want to edit, and then select 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.

     Searching all levels results in a very thorough search but can significantly increase the time that it takes to perform the search. However, the installer stops searching in any directory as soon as it finds a VM in that directory.

Expanded Search Functionality for String Values

Now, InstallAnywhere supports expand search functionality for String Value. If the partial search is unchecked, the string search will search for the whole text. If the partial search is checked, then the parts of strings & variables will also be searched.

The “Variable Search” which is a new checkbox has been added in InstallAnywhere.

- If the Variable Search check box is checked, it will enclose with a dollar sign symbol ($) before and after the variable to search for a variable.
- If the Variable Search check box is unchecked, it will not be enclosed with a dollar sign symbol ($) before and after the text to search for a string and variable.

Customizing Java Executable Patterns for Windows-Based Installations

The default search patterns that InstallAnywhere-generated installers use on Windows-based systems include:

- `java.exe`
- `bin\java.exe`
- `jre\bin\java.exe`

You can customize the list of patterns as needed.



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



Task ***To add a new search pattern to the Java Executable Patterns list:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **Windows** subtab.
4. Click the **Java Executable Patterns** subtab.
5. Click the **Add** button. InstallAnywhere adds a new row to the search paths list.
6. Enter the new search pattern.

Removing a Search Pattern



Task ***To remove a search pattern from the Java Executable Patterns list:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **Windows** subtab.
4. Click the **Java Executable Patterns** subtab.
5. Click the search pattern that you want to delete and then click the **Remove** button.

Editing an Existing Search Pattern



Task ***To edit an existing search pattern in the Java Executable Patterns list:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **Windows** subtab.
4. Click the **Java Executable Patterns** subtab.
5. Click the search pattern that you want to edit and then enter your changes.

Customizing UNIX Search Paths

InstallAnywhere lets you customize search path settings for installers that target UNIX-based systems.

Adding a Search Path for UNIX-Based Systems



Task ***To add a search path for UNIX-based systems:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **UNIX** subtab.
4. Click the **Search Paths** subtab.
5. Do one of the following:
   - To specify an absolute path for the search location:
     a. Click the **Add** button. InstallAnywhere adds a new row to the search paths list.
     b. Click the new row and then enter the absolute path to the directory that you want the installer to search for Java executable files.

        Note that you should not use InstallAnywhere variables or magic folders in the path statement.

        By default, new search paths search only the top level of the directory that you specify. No subdirectories are searched.
   - To specify a PATH environment variable search, click the **Add Path Environment Variable** button. InstallAnywhere adds a new **Path Environment Variable Search** row to the search paths list.

     The PATH environment variable search scans the top-level of any directories in the PATH environment variable.

Removing a Search Path from UNIX-Based Systems



Task ***To remove a search path from UNIX-based systems:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **UNIX** subtab.
4. Click the **Search Paths** subtab.
5. Click the search row that you want to remove, and then click the **Remove** button.

Editing an Existing UNIX Search Path

You can edit a search path that uses an absolute path.



Task ***To edit an existing UNIX search path:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **UNIX** subtab.
4. Click the **Search Paths** subtab.
5. Click the search row that you want to edit. This activates the entry and places the cursor at the end of the path statement.
6. Enter your path changes.

Changing the Directory Depth for a UNIX Search Path



Task ***To change the directory depth for a UNIX search path:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **UNIX** subtab.
4. Click the **Search Paths** subtab.
5. In the search paths list, double-click the level that you want to edit, and then select 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.

     Searching all levels results in a very thorough search but can significantly increase the time that 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-Based Installers

The default search patterns that InstallAnywhere-generated installers use on UNIX-based systems include:

- `java`
- `bin\java`
- `jre\bin\java`

You can customize the list of patterns as needed.



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



Task ***To add a new search pattern to the Java Executable Patterns list:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **UNIX** subtab.
4. Click the **Java Executable Patterns** subtab.
5. Click the **Add** button. InstallAnywhere adds a new row to the search paths list.
6. Enter the new search pattern.

Removing a Search Pattern



Task ***To remove a search pattern from the Java Executable Patterns list:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **UNIX** subtab.
4. Click the **Java Executable Patterns** subtab.
5. Click the search pattern that you want to delete and then click the **Remove** button.

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. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Search Panel Settings** tab.
3. Click the **UNIX** subtab.
4. Click the **Java Executable Patterns** subtab.
5. Click the search pattern that you want to edit and then enter your changes.

Customizing Individual Launcher Settings

InstallAnywhere lets you specify which JVM each LaunchAnywhere in your project should use.



Task ***To specify which JVM a launcher in your project should use:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. In the **Visual Tree**, select the **Create LaunchAnywhere** item that you want to configure.
3. Click the **Properties** tab at the bottom of the screen.
4. Click the **Advanced Settings** subtab.
5. In the **Configure LaunchAnywhere with** list, select the appropriate option. Available options are:
   - **VM selected by the installer or by the end user via Choose VM Panel**—The launcher uses the same JVM that the end user selected or that the installer chose.
   - **VM used by the installer**—The launcher uses the same JVM that was used to run the installer.

     When the VM that was bundled with the installer is present, the launcher uses that VM. If no VM was bundled with the installer, the launcher uses the JVM that was specified in the Installer Valid VM List (on the Installer Settings tab, which is available in the JVM Settings view of the Project page) as its value for `lax.nl.valid.vm.list`.
   - **the first VM found in the system matching the VM Search Settings defined under Project -> JVM Settings**—The launcher uses the JVM that was selected during the installer's JVM search.

     The launcher's JVM selection criteria are set at install time. Therefore, the launcher itself does not use any customized search paths or Java executable patterns that are set on the Windows and UNIX subtabs of the Search Panel Settings tab (Project page > JVM Settings view > Search Panel Settings tab).
   - **no specific VM**—The launcher performs its own JVM search at the time that the launcher is run.

     If you select this option, the launcher uses the value in its `lax.nl.valid.vm.list` property as the VM search criteria.



Note ▪ *On the Windows platform, only JVMs located in secure folders are permitted to launch the installer with administrative privileges. JVMs found in unsecured folders are not permitted for use and are restricted by the launcher with a security error message. This behavior can be turned off by using one of the following methods:*

- *Configure the `designer.jvm.securefolder.check` property to `false` in the `com.zerog.ia.Designer.properties` file.*
- *Configure the **Secure JVM Folder Check** setting to **Disable** in the **Project** > **Platforms** >**Windows** area.*
- *Run the installer using the `<installer_name> -Djvm.securefolder.check=<false>` command line, where `<installer_name>` refers to your installer executable.*



Note ▪ *For a detailed discussion of how the launcher's Advanced Settings options affect VM selection, see About the Launcher's JVM Selection Behavior at Run-Time.*

Using the Choose Java VM Panel

If you want end users to be able to choose which JVM is used at run time, you can include the Choose Java VM panel in your installer.



Task ***Adding a Choose Java VM panel:***

1. In the Advanced Designer, on the **Sequence** page, click **Pre-Install**. The **Pre-Install** view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. Click the **Choose Java VM** action, and then click the **Add** button. InstallAnywhere adds the action to the **Pre-Install Action List**. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. Use the arrow keys to move the action up or down in the action list.

There will be an error message when you select a Java VM that is not on the valid VM list which will indicate 'The selected application is not a valid Java Virtual Machine for this installation.' when choosing a Java VM from the **Choose Java VM** panel.

To specify an action's settings, select the action and use its customizers at the bottom of the view. For detailed information on the Choose Java VM action's customizer settings, see Choose Java VM Panel Action.

By default, the Choose Java VM panel action enables end users to search for JVM locations other than the ones that are defined in the project. In addition, the default behavior enables end users to choose a specific Java executable file. To override this default behavior, configure the action's customizer settings as needed.

For Windows-based and UNIX-based installers, you can customize the search paths and search patterns that the JVM search uses. For details, see Customizing the VM Search Paths and Patterns.

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 that cater to different versions of the JVM, as well as belong to different vendors such as IBM, Sun Microsystems, and HP-UX. 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
/:
```

A slash followed immediately by a colon (`/:`) is used to separate keys that can have multiple values.

During the launch of the installer, the installer's launcher reads the JVM spec file and uses the hints in this file to determine the location where the JRE is available on the target system.

Customizing and Creating JVM Spec Files

InstallAnywhere provides JVM spec files that target all the supported platforms. You can customize these files as well as add new files to the existing set.

You can instruct the installer to use more than one JVM spec file at the same time. (Use the JVM Search Settings box by selecting more than one file in the JVM Search Settings list (Build page > Build Installers view > Build Configurations tab > Build Targets subtab). The order in which the JVM spec files are listed in the JVM Search Settings list is important. The installation attempts to find a matching JRE using the first JVM spec file specified in the list, then work its way down the list.

Guidelines for Writing a JVM Spec File

If you write your own JVM spec file, note the following guidelines:

Table 6-6 ▪ Guidelines for Writing a JVM Spec File (Sheet 1 of 3)

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: • **Windows**: `bin\java.exe` • **UNIX**: `bin/java` You should maintain the same format when customizing the spec file.

Table 6-6 ▪ Guidelines for Writing a JVM Spec File (cont.) (Sheet 2 of 3)

Guideline	Description
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: • **Windows**: C:\Program Files\jre1.5.07 • **UNIX**: /usr/java5_64/jre the PATH_HINT should be: • **Windows**: \Program Files\jre1.5.07 • **UNIX**: /usr/java5_64/jre  ***Note ▪*** *For Windows, you should not mention the drive letter (such as C: or D:) in the* PATH_HINT *section.*
Wildcard (*) character can be used in the PATH_HINT section	The wildcard (*) character can be used in the PATH_HINT section.
Regular expressions limited to ending ellipsis or wildcard character	Regular expressions in the JVM_PROPERTIES section are limited to ending ellipsis (...) or wildcard (*) character.

Table 6-6 ▪ Guidelines for Writing a JVM Spec File (cont.) (Sheet 3 of 3)

Guideline	Description

Locations where JVM spec files can be stored	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.
	
	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:
	
	● **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.
	
	Instead of adding each platform-specific directory to this environment variable, such as:
	
	`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….`
	
	you should point this environment variable to the directory which contains all of the platform-specific directories, such as:
	
	`IA_JVM_SPECS_PATH=C:\IDEs\IA2011RC1\resource\jvms`
	
	Note ▪ *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:*
	
	

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

InstallAnywhere enables you to generate a response file. By default, all of the variables that are defined in your installation are recorded.



***Note** ▪ By default, a response file is named* `installer.properties` *or* `[installername].properties` *and it is created in the same directory as the installer.*



***Important** ▪ A response file must be saved with the appropriate encoding.*

For Windows-based target systems, the response file must be saved in one of the following encodings:

- *UTF-8 without a BOM*
- *UTF-16 little endian*

For Linux-based and OS or OS X–based target systems, the response file must be UTF-8 without a BOM.

If an unsupported encoding is used, the installer is unable to read the file properly.

Generating Response Files By Selecting Yes in the Always Generate Response Files Setting



Task ***To automatically generate a response file and exclude variables:***

1. In the Advanced Designer, on the **Project** page, click **General Settings**. The **General Settings** view opens.
2. In the **Project Information** area, click the **Response File** setting.
3. In the **Always Generate Response File** setting, select **Yes**.
4. In the Advanced Designer, on the **Project** page, click **Variables**. The **Variables** view opens.
5. In the **Configure Variables** area, click the **Configure** button. The **Configure Variables** dialog box opens.
6. Click the **Add** button. InstallAnywhere adds a blank row to the list.
7. In the **Variable Name** column, enter the name of the variable you want to exclude.
8. In the **Options** column, double-click the blank box and then select the appropriate option:
   - **Exclude Variable Entirely**—Exclude the variable name and value in the response file.
   - **Exclude Value Only**—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 selecting **Yes** in the **Always Generate Response File** setting by using the `-r` command-line option, and you can use it in combination with other options.



Important ▪ *While using `-r` command-line option, if you do not specify a path and file name for the installer's response file at the command line, the file will be named installer.properties or [installername].properties and it will be created in the same directory as the installer.*

Specifying a Different Response File Location

To override the default location when generating a response file for the installer, specify the location in the command line:

```
install.exe -r DestinationPath
```

Specifying a Different Response File Location and Response File Name

To override the default file name and default location when generating the response file for the installer an, specify the path and new name in the command line:

```
install.exe -r FullPathAndFileName
```

For example, to override the default file name and default location of response file for the installer:

```
install.exe -r C:\IAProjects\ResponseFiles\MyResponseFile.txt
```

In this scenario, a response file named `MyResponseFile.txt` is generated at the specified location, provided the directory or path is accessible to the end user. Also, if a file with the same name exists at the specified location, the end user must have permission to overwrite that file.

Getting User Input at Run Time

InstallAnywhere offers two versions of a Get User Input panel. You can include either version in your installer.

- **Get User Input - Simple panel**—This panel lets you collect basic information from end users during the installation process. The panel includes a panel title, prompt, and one or more text fields, check boxes, radio buttons, popup menus, or lists. Note that only one type of control can be displayed on each Get User Input - Simple panel.

  To learn more, see Using the Get User Input - Simple Panel.

- **Get User Input - Advanced panel**—This panel lets you collect more complex information from end users than the simple version of this panel. The panel includes a panel title, prompt, and one or more types of controls.

  To learn more, see Using the Get User Input - Advanced Panel.

Using the Get User Input - Simple Panel

The Get User Input - Simple Panel action lets 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 on 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. In the Advanced Designer, on the **Sequence** page, click the appropriate sequence (**Pre-Install**, **Post-Install**, **Pre-Uninstall**, or **Post-Uninstall**). The appropriate view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. On the **Panel** tab, click the **Get User Input - Simple** action, and then click the **Add** button. InstallAnywhere adds the action to the action list, and displays its settings in the bottom of the view. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. In the **Input Method** list, select the appropriate type of control that you want to use.
5. To add, edit, or remove controls, click the **Configure** button. The **Configure Input Items** dialog box opens.
6. Configure the settings as needed.
7. Click **OK**.



Tip ▪ *To view the controls that you have added to the panel, click the Preview button.*

Example: Obtaining Server Setup Information

This example demonstrates the use of 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. In the Advanced Designer, on the **Sequence** page, click **Pre-Install**. The **Pre-Install** view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. On the **Panel** tab, click the **Get User Input - Simple** action, and then click the **Add** button. InstallAnywhere adds the action to the action list, and displays its settings in the bottom of the view.
4. In the **Title** setting, enter the following:

   `Server Setup`
5. Click the **Configure** button. The **Configure Input Items** dialog box opens. (The default value for the Input Method setting is Textfields.)
6. Add an IP Address control to your panel:
   a. Click the **Add** button. InstallAnywhere adds a new row to the list of items.
   b. In the **Label** column, enter `IP Address`, and then press the TAB key.
   c. In the **Default Value** column, enter `127.0.0.1`.
7. Add a Port control to your panel:
   a. Click the **Add** button. InstallAnywhere adds a new row to the list of items.
   b. In the **Label** column, enter `Port`, and then press the TAB key.

c. In the **Default Value** column, enter `8080`.

8. Add a Server Contact control to your panel:
   a. Click the **Add** button. InstallAnywhere adds a new row to the list of items.
   b. In the **Label** column, enter `Server Contact`, and then press the TAB key.
   c. In the **Default Value** column, enter `yourname@yourdomain.com`.
9. Click **OK**.

InstallAnywhere adds a row for each control to the Labels and Defaults for Input Items table.

To see the panel with the current controls, click the Preview button.



***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

The Get User Input - Advanced Panel action lets create a custom panel with a variety of configurable controls that you can use to obtain input from end users. You may add an unlimited number of varied input items to a single panel.



Task ***To create the Get User Input - Advanced panel:***

1. In the Advanced Designer, on the **Sequence** page, click the appropriate sequence (**Pre-Install**, **Post-Install**, **Pre-Uninstall**, or **Post-Uninstall**). The appropriate view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. On the **Panel** tab, click the **Get User Input - Advanced** action, and then click the **Add** button. InstallAnywhere adds the action to the action list, and displays its settings in the bottom of the view. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. Click the button for the type of control that you want to add to your panel. Available options are: **Add Textfield**, **Add Choice Group**, **Add Label**, or **Add File Chooser**. InstallAnywhere adds a row for that control to the table next to the buttons.
5. Click the new row and then click the **Configure Selection** button. A dialog box opens, enabling you to configure the settings for the control.
6. Configure the settings as needed.
7. Click **OK**.



***Tip** ▪ To view the controls that you have added to the panel, click the Preview button.*

Using Command-Line Arguments with Installers and Uninstallers

InstallAnywhere-generated installers and uninstallers can be run with command-line arguments that can dictate, alter, or override the function of the installer and uninstaller.

- Setting the Interface Mode
- Setting the Installer Locale
- Generating a Response File
- Using a Response File or installer.properties File
- Setting Custom Variables
- Setting JVM Heap Size
- Showing Installer Help
- Setting the Temp Directory
- Manage the JVM Secure Folder Validation

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/uninstaller_name> -i <mode>`**

For example, to use a specific interface mode with an installer or an uninstaller, use a command such as one of the following:

Table 6-7 ▪ Interface mode -Specific Sample Commands for Installers and Uninstallers

Interface mode	Installer	Uninstaller
gui	`install.exe -i gui`	`uninstall.exe -i gui`
console	`install.exe -i console`	`uninstall.exe -i console`
silent	`install.exe -i silent`	`uninstall.exe -i silent`



Note ▪ *Using `-i` command-line switch without an argument (interface mode) or with an invalid argument (interface mode) for launching the Installer, results to display a usage message including the appropriate available options on 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
```



Note ▪ *Uninstaller does not support the command-line argument for setting locale and will always run with locale set for installer.*

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/uninstaller_name> -r "path_and_file_name"**

For example, to create a response file for the installer and uninstaller, you could enter the following at the command line:

```
install.exe -r "C:\Users\James\myresponse.properties"
```

```
uninstall.exe -r "C:\Users\James\myresponse.properties"
```



Note ▪ *Consider the following to create a response file for the installer and uninstaller using the command-line switch:*

- *For the installer if you do not specify a path and file name for the response file in the command line, the file will be named `installer.properties` or `[installername].properties` and it will be created in the same directory as the installer.*
- *For the uninstaller if you do not specify a path or a file name for the response file at the command line, the response file will not be created. Both a path and a file name should be specified for the response file creation.*



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 John's Desktop directory, enter the following:

```
install.exe -f "C:\Users\John\Desktop\installer.properties"
```



Note ▪ *Uninstaller does not support the command-line argument for using a response file or `installer.properties` file.*

Setting Custom Variables

To set custom variables from the command line, perform the following steps:



Task ***To set custom variables from the command line:***

At the command line, enter **`<installer_name> -D<myvar=myvalue>`**

For example, to set a new value for the install directory (`$USER_INSTALL_DIR$`), use a command such as one of the following ones:

Table 6-8 ▪ Platform-Specific Sample Commands for Defining Custom Variables from the Command Line

Platform	Command
Linux	`install.bin -DUSER_INSTALL_DIR="/home/jane/InstallationDirectory"`
OS or OS X	`install.bin -DUSER_INSTALL_DIR="/Users/jane/InstallationDirectory"`
Windows	`install.exe -DUSER_INSTALL_DIR="C:\Users\jane\Desktop\MyDirectory"`



Note ▪ *Uninstaller does not support the command-line argument for setting custom variables.*

Setting JVM Heap Size

To set JVM heap size from the command line, perform the following steps:



Task ***To set JVM heap size from the command line:***

1. To set the initial JVM heap size, at the command line, enter:

   ```
   <installer_name> -jvmxms <size>
   ```

2. To set the maximum JVM heap size, at the command line, enter:

   ```
   <installer_name> -jvmxmx <size>
   ```

The default size for these values is measured in bytes. Append the letter `k` or `K` to the value to indicate kilobytes, `m` or `M` to indicate megabytes, and `g` or `G` to indicate gigabytes. For example, to set the maximum JVM heap size to 25 megabytes, enter the following:

```
install.exe -jvmxmx 25m
```



Note ▪ *Uninstaller does not support the command-line argument for setting JVM heap size.*

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> -?`**

For example

```
install.exe -?
```



Note ▪ *Uninstaller does not support the command-line argument for showing uninstaller help.*

Setting the Temp Directory

To set the temp directory path from the command line, perform the following steps:



Task ***To set the temp directory from the command line:***

At the command line, enter **`<installer_name/uninstaller_name> -tempdir "Path"`**

You can set the required temp directory path by using the `-tempdir` argument with the installer or uninstaller name as shown above. For instance, to set the temp directory path to `C:\ExampleTemp` for the installer, where the InstallAnywhere launchers are required to be extracted, you could enter the following at the command line:

```
install.exe -tempdir "C:\ExampleTemp"
```



***Note** ▪ If the specified directory path is not accessible or is not a valid path, then the default temp directory path will be utilized by the InstallAnywhere installer/uninstaller.*

Manage the JVM Secure Folder Validation

To enable or disable the validation of the Java Virtual Machine (JVM) secure folder during installation or uninstallation from the command line, perform the following steps:



Task ***To enable or disable JVM secure folder validation from the command line***

At the command line, enter **`<installer/uninstaller_name> -Djvm.securefolder.check=<option>`**

where **`<option>`**:

- **true**—Enables Java Virtual Machine (JVM) secure folder validation, allowing the InstallAnywhere launcher at runtime to verify whether the JVM is located in a secure or administrative folder. By default, this validation is enabled on the Windows platform when running the installer or uninstaller with administrative privileges.
- **false**—Disables Java Virtual Machine (JVM) secure folder validation, allowing the installer or uninstaller to proceed with installation or uninstallation, respectively, even if the Java Virtual Machine is located in an unsecured folder.

When this validation is enabled, the launcher allows the installer or uninstaller to run only if the JVM is located in a secure or administrative folder. JVMs found in unsecured folders are blocked, and the launcher terminates execution with a security-related error message before the installer or uninstaller starts.

For example, to disable Java Virtual Machine (JVM) secure folder validation during installation, you can enter the following at the command line:

```
install.exe -Djvm.securefolder.check=false
```

Setting Project Version at Build Time

InstallAnywhere offers several methods for changing the version number of an InstallAnywhere project at build time.

Using the 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 configure your installer to perform a disk space check at various points in the installation life cycle. You can also optionally have your installer display the available disk space on the Pre-Installation Summary panel at run time.

Task ***To check for disk space during the installation and display the available space:***

1. In the Advanced Designer, on the **Sequence** page, click **Pre-Install**. The **Pre-Install** view opens.
2. In the **Pre-Install Action List**, click the **Panel: Pre-Install Summary** action. InstallAnywhere displays the action's settings at the bottom of the view.
3. On the **General Settings** tab, select the **Disk space information in** check box, and then select the unit of measure that you want the installer to use to display the available disk space.

In the following example, MegaBytes was selected from the list:

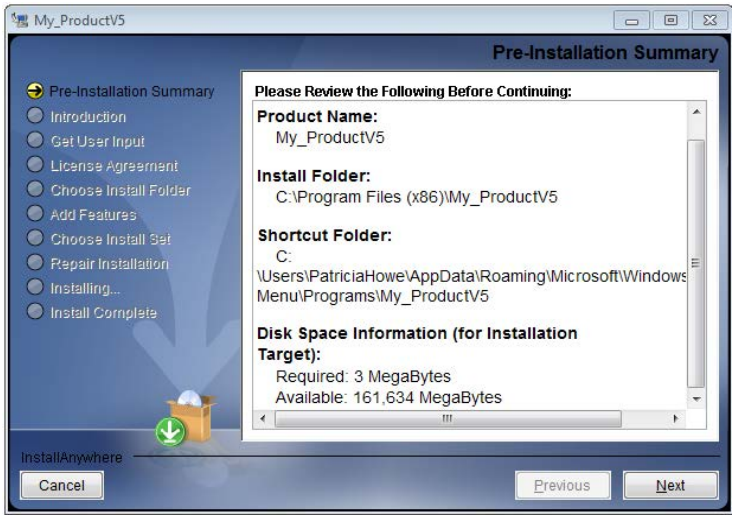


Figure 6-2: Pre-Installation Summary Panel Displaying Disk Space Information

If you want to use more than one unit of measure 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 6-9 ▪ 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 are always displayed in megabytes.*

Considerations Regarding Disk Space Checks

If you configure your installer to perform disk space checks, note the following:

- The Disk Space Check action is always the first action that is executed as part of Pre-Install sequence.
- 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 is also run on demand when there is a change in installation folder or install sets using either actions or variables.

Preparing Your Installer for Update Notifications

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 automatically notifies your Web-connected end users when patches, updates, and product information for your product are available. FlexNet Connect helps you reduce the number of end users running old releases of your products and prevent them from installing the wrong updates from your Web site.

By default, the Enable Update Notifications action also creates a launcher for the Java agent so that your end users have the option of checking for updates on demand.

Adding an Enable Update Notifications Action

You can integrate with FlexNet Connect and enable update notifications by adding an action to the Install sequence of your project.



Task ***To add an Enable Update Notifications action:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. Click the **Enable Update Notifications** action, and then click the **Add** button. InstallAnywhere adds the action to the **Visual Tree**, and displays the action's settings at the bottom of the view.
4. Configure the action's settings as needed.

For information about each of the action's settings, see Enable Update Notifications Action.



Tip ▪ *You can reassign the Enable Update Notifications action 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.*



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 the* ***Click here*** *button on the*

General Settings tab. The FlexNet Connect Publisher site opens. Enter your user name and password and click Sign In. Register your product using the screen prompts. Click the quick help icons for more information.

Registering Your Product with the FlexNet Connect Publisher Site

Before your installed products can receive update notifications, you must make sure that the product code of the product in your installer matches the product code of the product on the FlexNet Connect Publisher site.



Task ***To verify that your product code matches the product code:***

1. In the Advanced Designer, on the **Project** page, click **General Settings**. The **General Settings** view opens.
2. In the **Product Information** area, find the **Product Code** setting, and note its value.
3. In an Internet browser, sign in to the FlexNet Connect Publisher site, and open the **Products** view.
4. Select the product that your installers deploy, and then click the **View/Edit** button. The Edit Product page opens.

   Note ▪ *If your installer project's product does not exist as a product on the FlexNet Connect Publisher site, you must first enter the product there. Remember to use the product code from your installer as the product code on the FlexNet Connect Publisher site.*
5. On the **Edit Product** page, locate the product code and compare it to the product code in the **General Settings** view.

   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 (but 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 FlexNet Connect 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



Task ***To remove an Enable Update Notifications action:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. In the **Visual Tree**, click the **Enable Update Notifications** action that you want to remove.

3. Click the **Remove** button.

InstallAnywhere removes the Enable Update Notifications action from the Visual Tree.

Localizing Projects and Installers

InstallAnywhere offers various features that enable you to customize your installers for global distribution.

Generating Multilanguage Installers

InstallAnywhere enables you to specify the locales that you want your installers to support.

Additional localization may be required for resources, customized panels or consoles, and custom code.



Task ***To generate multilanguage installers:***

1. In the Advanced Designer, on the **Build** page, click **Build Installers**. The **Build Installers** view opens.
2. Click the **Build Configurations** tab.
3. In the **Select Build Configuration** list, select the configuration that you want to configure.
4. Click the **Locales** subtab.
5. Select the check box of each language that you want your installers to support. Clear the check box of each language that you do not want your installers to support.

Localizing Resources

You can localize resources (such as License Agreements, side panels, billboards, and custom icons) for specific different locales. Actions such as the License Agreement Panel and LaunchAnywhere serialize the paths and file names to their resources as well as their dynamic strings to the locale files. You can then change these paths and the file names.



Task ***To localize the License Agreement:***

1. Make sure to include every resource in the Install view on the Sequence page. Installers do not have access to resources that are not specified in this view.
2. Find the line in the locale file that contains the text `LicenseAgr.#.FileName`.
3. Specify the file name of the file that contains the localized license agreement (for instance, `License_fr.html`). Do not type the fully qualified absolute path to the file—just the file name itself.
4. Find the line that contains the text `LicenseAgr.#.Path`.
5. Specify the path 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. Provide translations for these labels in their respective files.

Localizing the Splash Screen

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 command-line option that sets the locale:

```
-l <language code>
```



Note ▪ *To learn more, 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



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 the directory is named to match your project name:

   `Project_name``locales_`*`Build_configuration_name`*

   For example:

   `My_Productlocales_OSConfiguration`

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 the number sign (`#`) represents the reference ID for the element.
3. Provide the translated title text as the value for the `Installer.#.splashScreenGUITitle` key.

Localizing the Splash Screen Image



Task ***To localize the splash screen image:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. Add all localized splash screen image files to the **DO NOT INSTALL** magic folder. This ensures that the image resources for the localized splash screens are available to your installer but not installed to the target system.
3. Locate your project's locales directory. This directory is in the same location as your project file, and the directory is named to match your project name:

   *Project_name*locales_*Build_configuration_name*

   For example:

   `My_Productlocales_OSConfiguration`
4. Open each non-English locale file in a text editor, and find the `splashScreenGUIImageName` and `splashScreenImagePath` elements. These elements appear in the locale files as `Installer.#.splashScreenGUIImageName` and `Installer.#.splashScreenImagePath`, where the number sign (`#`) represents the reference ID for the element.
5. Enter the correct locale-specific file name and path (if necessary) values for the `Installer.#.splashScreenGUIImageName` and `Installer.#.splashScreenImagePath` keys, respectively.

Localizing the Splash Screen Confirmation Button



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 the directory is named to match your project name:

   *Project_name*locales_*Build_configuration_name*

   For example:

   `My_Productlocales_OSConfiguration`
2. Open each non-English locale file in a text editor, and find the `splashScreenGUIConfirm` element. This element typically uses an `Installer.#.splashScreenGUIConfirm` key, where the number sign (`#`) represents the reference ID for the element.
3. Provide the translated confirmation button text as the value for the `Installer.#.splashScreenGUIConfirm` key.



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 on the Get User Input panel is localizable whether it is a label, text field, option button, or check box. Following are some examples that use 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, add a Get User Input - Simple panel that asks the question: “Are you ready?” This panel displays option buttons with Yes and No as possible choices.
2. Build for all intended platforms and locales.
3. Locate your project’s locales directory. This directory is in the same location as your project file, and the directory is named to match your project name:

   Project_name`locales`*_Build_configuration_name*

   For example:

   `My_Productlocales_OSConfiguration`
4. Use a simple text editor (for example, Notepad, TextEdit, VI, or es) to open the `custom_fr` and `custom_en` files.
5. In the `custom_en` file, search for the text that 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= Êtes-vous prêt?

   # CreateDialog.c586fa408eaf.title=Get User Input
   ```

```
CreateDialog.c586fa408eaf.title=Obtenez Entrée de L'Utilisateur

# 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 installer uses the proper entries in the proper locale.

Get User Input - Advanced Example

The process is similar for the Get User Input - Advanced panels. Following 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, add a Get User Input - Advanced panel that asks the question: “Are you ready?” This panel displays option buttons with Yes and No as possible choices.
2. Build for all intended platforms and locales.
3. Locate your project’s locales directory. This directory is in the same location as your project file, and the directory is named to match your project name:

   *Project_name*locales_*Build_configuration_name*

   For example:

   `My_Productlocales_OSConfiguration`
4. Use a simple text editor (for example, Notepad, TextEdit, VI, or es) to open the `custom_fr` and `custom_en` files.
5. In the `custom_en` file, search for the text that you want to customize (the text string used in your Get User Input - Advanced 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=Obtenez Entrée de L'Utilisateur

# GUIGroupData.c8e471658f03.caption=Are you ready?
GUIGroupData.c8e471658f03.caption= Êtes-vous prêt?

# GUIComponentData.c8e477e48f04.label=Yes
GUIComponentData.c8e477e48f04.label=Oui

# GUIComponentData.c8e4c9558f04.label=No
GUIComponentData.c8e4c9558f04.label=Non
```

8. Rebuild your project.

The installer uses the proper entries in the proper locale.

Adding an External Resource Bundle

External resource bundles contain custom locale files (one per locale) that 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 that the installer references, based upon the value of `$INSTALLER_LOCALE$`.



Task ***To add an external resource bundle:***

1. On the **Project** page, click **Locales**. The **Locales** view opens.
2. In the **External Resource Bundle Settings** table, in the **Bundle Name** column, double-click an empty field.
3. Enter a name for the external resource bundle that you want to use.
4. Double-click the row's **Resource Bundle Path** column. The **Choose a File** dialog box opens.
5. Locate the directory in which your custom locale properties file is stored and select one of the locale properties files.
6. Click the **Select** button.



Note ▪ *Verify that the external resource bundle contains a properties file for each locale that you plan to support. Installers that are 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

To reference the keys that you created in the locale properties files of your external resource bundle in your installers, use the following syntax.

`$L{Bundle_Name.Key}`

Bundle_Name represents the name that you provided in the External Resource Bundle Settings table in the Locales view of the Project page. *Key* is the key for the localized value that 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 that the end 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.*

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 looks 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. You can create any name for the key, as long as it does not conflict with previously defined keys. You can even use a pre-existing key to obtain a string that has already been translated in InstallAnywhere's resource files.

However, the prefix **`_ia.`** must be added to the beginning of each key. For example, if the custom code used 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.` prefix and searching for the remainder of the key name.*

Locale key-value pairs are set directly in the project locales files. 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 or edit locale keys for every installer project, update the static text. To create or edit keys only in the current project, update the dynamic text. The dynamic text is regenerated every time that you save your project, so update the files every time that the project is changed.

Default to Base Locale for French-Canadian and Portuguese-Brazilian

For French and Portuguese, InstallAnywhere supports both base and a customized version of the locale:

- French (fr) and French Canadian (fr_CA)
- Portuguese (pt) and Portuguese Brazilian (pt_BR)

If you create a project and build it with both English and French locales (with English set as default) but not with the French Canadian locale, if the installer is run on a machine with a system locale of French-Canadian, the installer will use French instead of English. InstallAnywhere is smart enough to know that the base locale of the missing locale is a better choice than using the actual default locale.

The same is true for Portuguese (pt) and Portuguese-Brazilian (pt_BR).



Note ▪ *This behavior is not applicable for Chinese because there is no base locale for Chinese-simplified and Chinese-traditional.*

Packaging and Executing Custom Code

Custom code can extend the functionality of your installer. The following procedure explains how to ensure that 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 needs to reference the classes in this file when compiling the code. `IAClasses.zip` is located in the root installation directory of InstallAnywhere.
2. Add the action, panel, console, or rule to your project. A good starting point is to use the Java source file templates in the `CustomCode/Templates` folder, which is available in the InstallAnywhere installation directory.
3. Compile the source files.
4. Decide which additional files and resources your custom code requires. For example, your code may require images, text files, or other resources.
5. Create a Java archive file (.jar) that contains the compiled class files and resource files.



***Note** ▪ Make sure that the JAR file has full path information in it. (Each file in the archive should be stored with its proper package path.) Without the path information in the JAR file, Java will not be able to properly find the packaged class files.*

It is critical that the custom code JAR does not include the classes and resources from `IAClasses.zip` since this would seriously affect the behavior of the installer and uninstaller.

6. Add to your project an action or rule that you want to execute custom code. To learn more, see the following:
   - Adding Actions to the Install Sequence
   - Defining Rules and Rule Expressions that Evaluate Conditions on Target Systems
7. For your action or rule, select the JAR file that contains your custom code, and enter the fully qualified package and class name, such as: **`com.acme.MyAction`**.
8. Configure the action's or rule's 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**.*

9. Test and debug the action or panel.

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.

A frequent source of trouble with custom code is incompatibility between the JDK with which you compile your code and the JVM that you bundle with your installer. Ensure your Java compiler is compatible with your installers' bundled JVMs.

Support for Signed JARs as Dependencies

InstallAnywhere supports adding signed JARs as dependencies to custom code actions, custom code panels, custom code consoles, and custom rules. Also, InstallAnywhere supports signing JAR and ZIP files that contain custom code.

If your project contains a signed JAR file with your custom code, InstallAnywhere does not extract the contents of the JAR at build time; the contents are left in the `execute.zip` file. At run time, the signed JARs are extracted to temporary directories and added to the InstallAnywhere classpath. This makes the JAR contents ready to be consumed for the custom code or any relevant actions that require the contents of the signed JAR.

You can designate that a signed JAR is a dependency in various scenarios—for example, a signed JAR can be a dependency for an Execute Custom Code action in your project.

Execute Custom Code Action, Panel, or Console

If you configure 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 that are required for execution, including the fully qualified class name to execute.

In addition, specify the location of the signed JAR archive files on which the custom code depends. When InstallAnywhere builds the installer, InstallAnywhere leaves the signed jar and its signing information intact.

At run time, the installer places the dependant signed JARs on the InstallAnywhere classpath to be consumed while the relevant action execution occurs.

Evaluate Custom Rule

If you configure an Evaluate Custom Rule, you can click Configure Dependencies and select a signed JAR on the Custom Rules Dependencies dialog box.

Maintenance Mode, Instance Management, Plug-Ins

You can also include signed JARs for some maintenance mode options (such as add or repair) and also for instance management. Also, custom code files 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) that you can import into a custom code action. The following service interfaces are available:

Table 6-10 ▪ 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.
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.



Task ***To call InstallShield MultiPlatform APIs in an installer:***

1. To access to the InstallShield MultiPlatform APIs, 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
   ```

When compiling code that uses these services, all JAR files in the *IA_HOME*/resource/services directory and the *IA_HOME*/resource/services/ppk directory should be in the compiler's classpath.

2. Add service support for your custom code to ensure that the services are built into the installer and made available at run time. To do so:
   a. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
   b. Click the **General Settings** tab.
   c. Select the **Add service support for custom code** check box.

For additional information on custom code, see the following resources:

Table 6-11 ▪ Additional Information on Custom Code

Subject	Location in InstallAnywhere's Installation Directory
Java Docs	/javadocs/index.html
Templates	/CustomCode/Templates
Samples	/CustomCode/Samples

Packaging Custom Code as a Plug-in

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. This properties file needs to contain all of the information that 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).
3. Include the following properties in the properties file:
   - **plugin.main.class=<classname>**—This property defines 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>**—This property identifies the name that InstallAnywhere uses for the plug-in on the Plug-Ins tab of the Advanced Designer's Choose an Action dialog box.
   - **plugin.type=<action | panel | console>**—This property identifies when the plug-in can be used and which icon to use to represent it in the Advanced Designer.

   Optionally include one or more of the following properties:

   - **property.<propertname>=<propertydefault>**—This property tells the plug-in to populate the action's customizer with a property named <propertyname> and set to the default value of <propertydefault>.

- **plugin.icon.path=<relative path to .png or .jpg file in JAR>**—This property sets a custom 32x32 icon for the custom code plug-in of the Advanced Designer.
- **plugin.available=<preinstall | install | postinstall | preuninstall | postuninstall>**—This property identifies a comma-separated value list that defines the sequences 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
```

4. 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. InstallAnywhere displays the Help button if a help file is provided in the plug-in. To include installer help for your plug-in:
   a. Create a file called `help.htm`, and instructions to this file.
   b. Package it in the plug-in JAR (without stored path information).
5. Place the properly packaged JAR (with the custom code, its resources, and the properties file) in *IA_HOME*`/plugins`.

The next time that you launch InstallAnywhere, the plug-in will be visible in the InstallAnywhere Choose an Action dialog box.

Digitally Signing Windows-Based Installers

InstallAnywhere includes support for digitally signing your Windows-based installers (installer .exe file, installer launcher, and uninstaller launcher) during build time. Digitally signing your Windows-based installers assures end users that your installers have not been tampered with or altered since release. End users are presented with a digital certificate when they run your installers.

If you have not digitally signed an installer, launching your installer on Windows XP SP2 and later results to display an unknown publisher warning to end users.

The ability to digitally sign Windows-based installers during build time requires a personal information exchange file (.pfx) type of digital certificate or an external tool that is configured to access the required certificate, depending on the signing method opted. In addition, it requires that you are using InstallAnywhere on a Windows-based system. If you try to build a Windows-based installer on a non-Windows system, InstallAnywhere does not sign the resulting installer.



Tip ▪ *If the certificate authority that issued you a digital certificate provided you with a private key file (.pvk) and a software publishing credentials file (.spc) instead of a .pfx file, you can use PVK2PFX.exe to create a .pfx file from the files that were provided to you. PVK2PFX.exe is part of the Microsoft Windows Platform SDK.*

Certification Authorities

A certification authority is an organization such as VeriSign that issues and manages digital certificates (also known as digital IDs). The certification authority validates the requester's identity according to prescribed criteria and issues a digital certificate. Obtaining a digital certificate requires providing the certificate authority with specific information about your company and your product.

For a list of certification authorities, see the Windows Root Certificate Program member list on the MSDN Web site.

SHA-1 vs. SHA-2 Certificates

InstallAnywhere enables you to use digital certificates that use the SHA-256 or SHA-1 hashing algorithm for signing your installations and files at build time.

SHA-2 is favored over SHA-1, which is being deprecated because of the potential for security vulnerabilities. Microsoft announced that Windows will stop trusting items that were signed and timestamped with SHA-1 certificates after January 1, 2016. In addition, certification authorities—the organizations that issue certificates—are phasing out the creation of SHA-1 certificates. Thus, it is recommended that you replace any SHA-1 certificates in your InstallAnywhere projects with SHA-2 certificates. For the latest information and more specific details, check with your certification authority.

If your project is configured to sign with a SHA-2 certificate, InstallAnywhere uses a SHA-2 hash in the signature of the files that it signs at build time. If your project is configured to sign with a SHA-1 certificate, InstallAnywhere generates a build error: either 813 (a SHA-1 certificate is configured in the project) or 814 (a SHA-1 certificate is configured in the project and a timestamp server is not being used for signing).

Support for Digital Signature in Standard Signing

In InstallAnywhere 2026 R1, when you are configuring digital signature information in standard signing approach, use the Certificate Selection dialog box to specify which certificate you want to use to sign your files. InstallAnywhere lets you choose between the following options:

- You can specify the .pfx certificate file on your machine that you want to use for signing.
- You can reference a certificate store that contains the certificate that you want to use for signing.

Accessing the Certificate Selection Dialog Box

Instructions on how to access the Certificate Selection dialog box depend on whether you are specifying certificate information for a release, a patch, or a QuickPatch package.



Task ***To access the Certificate Selection dialog box for a release:***

1. In the Advanced Designer, on the **Project** page, click on the **Platforms**. The **Platforms** view opens.
2. In the **Windows** area, under the **Digital Signing** setting, select the **standard** option in the **Signing Method** setting field.
3. Then in the **Certificate Information** setting field, click the ellipsis button (**...**).

Certificate Selection Dialog Box Settings

Table 6-12 ▪ Certificate Selection Dialog Box Settings

Setting	Description
Use a file (.pfx)	To use a .pfx file to digitally sign your release at build time, select this option. Then specify the location of your .pfx. You can type the path to the file or use the ellipsis button (...) to browse to the file location.
Use a certificate store	To reference a certificate store that contains the certificate that you want to use to digitally sign your release at build time, select this option and then enter values in the subsettings under this option.
Certificate Store Name	Select the name of the certificate store that contains the certificate that you want to use. Available options are: ● Personal ● Trusted Root Certification Authorities ● Enterprise Trust ● Intermediate Certification Authorities This setting is available if you select the **Use a certificate store** option.
Certificate Store Location	Select the location of the certificate store that contains the certificate that you want to use. Available options are: ● User ● Machine This setting is available if you select the **Use a certificate store** option.
Certificate Subject	Enter the subject of the certificate that you want to use, or select from the list of certificates that are available on your machine. This setting is available if you select the **Use a certificate store** option.
Signature Digest	Choose the signature digest hashing algorithm (or choose to let InstallShield specify it automatically based on the certificate hash). Available options are: ● Based on certificate hash ● SHA-1 ● SHA-256 ● Dual (SHA-1 & SHA-256)  ***Important ▪*** *Only when choosing Dual (SHA-1 & SHA-256), the Timestamp Server is Mandatory.*



Task ***To specify digital signature information in standard signing approach:***

1. In the Advanced Designer, on the **Project** page, click on the **Platforms**. The **Platforms** view opens.
2. In the **Windows** area, under the **Digital Signing** setting, select the **standard** option in the **Signing Method** setting field.
3. Configure each of the enabled settings as required in the standard signing approach.

As an alternative, to specify the actual certificate file, password, and timestamp server in the required subsettings of the **Digital Signing** setting on the **Platforms** view, you can use build-time variables in these subsettings (that is, enclose the name of each variable within at symbols: `@VariableName@`). You can set build-time variables in the **Variables** view on the **Project** page, through a .properties file, or through environment variables. For more information, see Resolving Variables at Build Time.

Support for Digital Signature in Custom Signing

When you are configuring digital signature information in custom signing approach, use the **Custom Sign Tool Location** and **Sign Tool Arguments** settings field to specify the external sign tool's location and argument respectively, which helps you to digitally sign your Windows-based installers via an external sign tool.



Task ***To specify digital signature information in custom signing approach:***

1. In the Advanced Designer, on the **Project** page, click on the **Platforms**. The **Platforms** view opens.
2. In the **Windows** area, under the **Digital Signing** setting, select the **Custom** option in the **Signing Method** setting field.
3. Specify the external sign tool's location and valid argument in the **Custom Sign Tool Location** and **Sign Tool Arguments** settings respectively, which are required in custom signing approach.

As an alternative, to specify the external sign tool's location and valid argument in the required subsettings of the **Digital Signing** setting, you can use build-time variables in these subsettings (that is, enclose the name of each variable within at symbols: `@VariableName@`). You can set build-time variables in the **Variables** view on the **Project** page using a .properties file, or as environment variables. For more information, see Resolving Variables at Build Time.

Verifying that Your Output Files Are Digitally Signed on Windows–Based Target Systems

Regardless of which method you use to prepare your digital-signed installers, at various points in the process, it is important to verify whether the target file has been properly digital signed before proceeding with the next step of the process.

It is also crucial to test the final output on a clean machine.

Command for Digital Signing in Windows

The command for executing IAWinDigiSign.exe in InstallAnywhere is given below:-

```
IAWinDigiSign.exe /f <pfx file path> /o <Name of signer> /p <password>
/t <timestamp> /algo <algorith> <executable to sign>
```

where:

- /f parameter is the path of Certificate file to be used for signing
- /o parameter is Signing Certificate Subject/certificate owner name
- /p parameter is the Password for encoded certificate
- /t parameter is the URL of time Stamp Server
- /algo parameter is the algorithm used for signing

followed by the Executable which is to be signed.



Important ▪ *If and only the executable is digitally signed properly, you will be able to view the Digital Signatures tab on the Properties dialog box. If not, an error message will be displayed.*

The following instructions explain how to verify digital signing output files.



Task ***To verify your output files are digitally signed:***

1. Right-click the product and click **Properties** to view the Product Properties.
2. Click **Digital Signatures** tab on the product **Properties** dialog box to view the Name of Signer, Digest Algorithm, and Timestamp.
3. Click **Details** on the **Digital Signatures** tab of the product **Properties** dialog box to verify that your digital signing output files are working properly on Windows-based Target Systems.

The **Digital Signature Details** dialog box lists out the below Signer information.

- Name of the Signer
- Email of the Signer
- The Signing time of the digitally signed output file.

You can also view the certificate information by clicking **View Certificate** on the **Digital Signature Details** dialog box.

About Authentication and Code-Signing Support for OS or OS X–Based Installers

InstallAnywhere has support for signing OS or OS X–based installers with a Developer ID Application certificate at build time. Signing an installer with this type of certificate enables you to distribute it outside the App Store. When an end user downloads and runs your installer, Gatekeeper allows the installer to run.

If an end user tries to launch an unsigned, downloaded installer on an OS X Mountain Lion or later (including OS) system on which Gatekeeper is turned on, the system displays an alert and blocks the installer from running.

If you want your OS or OS X–based installers and uninstallers to install files to and remove files from locations where write permissions are restricted for standard users, you can configure your project to require authentication. When authentication is required and standard users who are not root users or administrative users with adequate privileges try to launch your installer or uninstaller, they are prompted to enter an administrator name and password in order to proceed.

If your installer and uninstaller installs files to and removes files from unrestricted locations such as the user's home directory, authentication is not required on OS or OS X–based target systems.

Note that in order to require authentication for OS or OS X–based installers and uninstallers, they must be code signed.

Run-Time Behavior for a Code-Signed Installer that Includes Authentication Support

An installer that includes authentication support consists of the following primary files:

- An authentication wrapper
- A helper tool-the file that is used to launch the installer or uninstaller with elevated privileges
- The installer application-that is, the Java installer
- Optionally, the uninstaller application. Note that InstallAnywhere generates the uninstaller and bundles it into the installer at build time.

The authentication wrapper, helper tool, installer, and uninstaller must all be code signed with the same Developer ID Application certificate.

At run time on target systems, the following process occurs for a properly code-signed installer or uninstaller that includes authentication support:

1. The end user launch the authentication wrapper on a OS or OS X–based system.
2. The authentication wrapper prompts for elevation (if needed for a standard user) and installs the helper tool with root privileges.
3. The authentication wrapper requests that the helper tool launch the LaunchAnywhere.
4. The helper tool launches the LaunchAnywhere, which inherits the elevated privileges from the helper tool.
5. The LaunchAnywhere launches the installer or uninstaller, which inherits the elevated privileges.

At the end of the installation or uninstallation, the authentication wrapper shuts down the helper tool and uninstalls it. If any part of the process is not signed with a matching Developer ID Application certificate, the process fails.

Overview of the Process of Code Signing Installers and Including Authentication Support

Preparing Your Machines

The following steps outline the process of preparing your machines to build and code sign installers with authentication support.

1. Determine whether you will be performing the code-signing step at build time on your InstallAnywhere build machine, or on a designated code-signing machine. To learn more, see Code-Signing Methods for OS or OS X–Based Installers.
2. Ensure that your build and code-signing machines meet the requirements for code-signing support. To learn more, see Requirements for Code-Signing Support for OS or OS X–Based Installers.
3. If you have not already done so, obtain your Developer ID Application certificate. To learn more, see Obtaining a Developer ID Application Certificate for Code Signing OS or OS X–Based Installers.
4. Prepare your code-signing machine. This involves importing your Developer ID Application certificate, ensuring that the latest Xcode IDE and its default SDKs are present, building and code signing the helper tool, verifying the code-signed helper tool, and copying the code-signed helper to all of your InstallAnywhere build machines. To learn more, see Adding the Code-Signing Capability to Your InstallAnywhere Build Machines or Code-Signing Machines.

Configuring Your InstallAnywhere Projects

Once you have competed the aforementioned preparation steps, perform the following steps for generating code-signed installers with authentication support:

1. Configure your projects for code signing and authentication. To learn more, see Code Signing Your OS or OS X–Based Installers and Including Authentication Support.
2. Verify that the resulting installers were successfully code signed. To learn more, see Verifying that Your Code-Signing Output Files Are Working as Expected on OS or OS X–Based Target Systems.

Code-Signing Methods for OS or OS X–Based Installers

InstallAnywhere supports two different methods of code signing:

- **InstallAnywhere performs the code-signing step at build time on the InstallAnywhere build machine**—The InstallAnywhere user has access to the Developer ID Application certificate and, in the InstallAnywhere project, configures the Code Signing settings (the certificate location and the keystore password). At build time, InstallAnywhere code signs the appropriate output files.
- **The InstallAnywhere build output is code signed on a designated code-signing machine**—The InstallAnywhere user builds an unsigned installer on the build machine. The build output is then code signed on a secure code-signing machine that has the Developer ID Application certificate.



Note ▪ *A limitation of this second method—where code signing is performed on a designated code-signing machine— is that it is not possible to sign the uninstaller; therefore, the resulting uninstaller cannot be used to uninstall the product. The only way to remove a product whose installer was code signed using this method is to drag it to the Trash.*

Requirements for Code-Signing Support for OS or OS X–Based Installers

The following requirements must be met for code signing OS or OS X–based installers:

- The OS X–based installer must be built on a OS or OS X–based system.
- All code signing must be done on systems that are running OS X 10.9 or later (including OS), since these versions can create version 2 signatures. Version 1 signatures, which are created by earlier versions of OS X, are not recognized by Gatekeeper on systems with OS X 10.9 and later (including OS) and are considered obsolete. Files that are signed with version 2 signatures will work on OS X 10.8 and later (including OS). To learn more, see Technical Note TN2206: OS X Code Signing in Depth in the Developer Library.
- A Developer ID Application certificate must be used to sign the files. The certificate should be added to the login keychain—not the system keychain—on the machine that is going to be used for code signing, and the same user account that was used to add the certificate to the login keychain should be used to sign files.
- If you plan on performing builds through the command-line console, ensure that the certificate has been granted access to be used by all applications.
- Ensure that the latest Xcode IDE and all of its default SDKs are installed on the machine that is going to be used for code signing.
- The build target for an installer that requires authentication must be OS or OS X; generic UNIX–based build targets do not support authentication on OS or OS X–based systems.
- Merge modules cannot use authentication independently. To deploy a merge module that requires authentication, you must authenticate the parent installer.

Obtaining a Developer ID Application Certificate for Code Signing OS or OS X–Based Installers

The following instructions explain how to obtain the appropriate certificate so that you can code sign your OS or OS X–based installers.



Task ***To obtain the Developer ID Application certificate:***

1. Enroll in the Developer Program.
2. On an OS X 10.9 or later (including OS) system, sign in to the Member Center by clicking the Member Center link on the Apple Developer site and providing your credentials.
3. Use the Developer Certificate Utility on the site and the Keychain Access application on the OS or OS X–based system to create a certificate signing request (CSR).
4. Use the Developer Certificate Utility to generate a Developer ID Application certificate.
5. Install the certificate on the OS or OS X–based machine.
6. In Keychain Access, locate the installed certificate and export it to the PKCS #12 certificate file format (.p12). When you are prompted for the certificate password, enter it.

You can use the .p12 certificate file when setting up code signing for your helper tool and your OS or OS X–based installers.

Adding the Code-Signing Capability to Your InstallAnywhere Build Machines or Code-Signing Machines

The following instructions explain how to prepare your InstallAnywhere build machine or your code-signing machine so that you can code sign your OS or OS X–based installers and include authentication support. The process involves adding your Developer ID Application certificate to the Keychain Access utility on all of your machines that you will be using for code signing. The process also involves creating a code-signed helper tool that you also add to all of your code-signing machines.



Note ▪ *If you want to prepare your InstallAnywhere build machine or your code-signing machine for code signing your installers, but you do not need to be able to include authentication support in your installers, perform step 1 only. It is not necessary to generate a code-signed helper tool on your code-signing machines unless you plan to support authentication.*



Task ***To add the code-signing capability to your machines:***

1. Set up the Keychain Access utility on all of your machines that you will be using for code signing (either InstallAnywhere build machines or separate designated signing machines):
    a. Add your Developer ID Application certificate to the login keychain. For information on adding a certificate to a keychain, see Adding certificates to a keychain in the Developer Library.
    b. *If you plan on performing builds through the command-line console:* Ensure that the certificate has been granted access to be used by all applications.
        i. In **Keychain Access**, right-click the certificate and then click **Get Info**.
        ii. On the **Access Control** tab, click the **Allow all applications to access this item** option.
2. Ensure that the latest Xcode IDE and all of its default SDKs are installed.
3. *If you are using your InstallAnywhere build machine to code sign, perform the following step in InstallAnywhere to generate a code-signed helper tool:* On the **File** menu, click **Prepare Helper Tool**. The **Prepare the Helper Tool** dialog box opens. Use this dialog box to specify code-signing information and code sign the helper tool. To learn more, see Prepare the Helper Tool Dialog Box.
4. *If you are using a separate designated signing machine, perform the following steps to generate a code-signed helper tool:*
    a. Copy the ht-signer folder in the `IA_INSTALL_DIR/resource/nativetools/osx` folder to the designated signing machine.
    b. In the ht-signer folder that you copied to the designated signing machine, find the `build-helper-tool.sh` file, open it in an editor, and customize the following entries:

- **CERTIFICATE_ID**—Specify the common name of the certificate. This name is displayed in Keychain Access (see screen shot below). The common name must use syntax such as this:

  "Developer ID Application: AAA Software LLC"

- **CERTIFICATE_DEV_ID**—Specify the user ID of the certificate. This ID is displayed in Keychain Access.

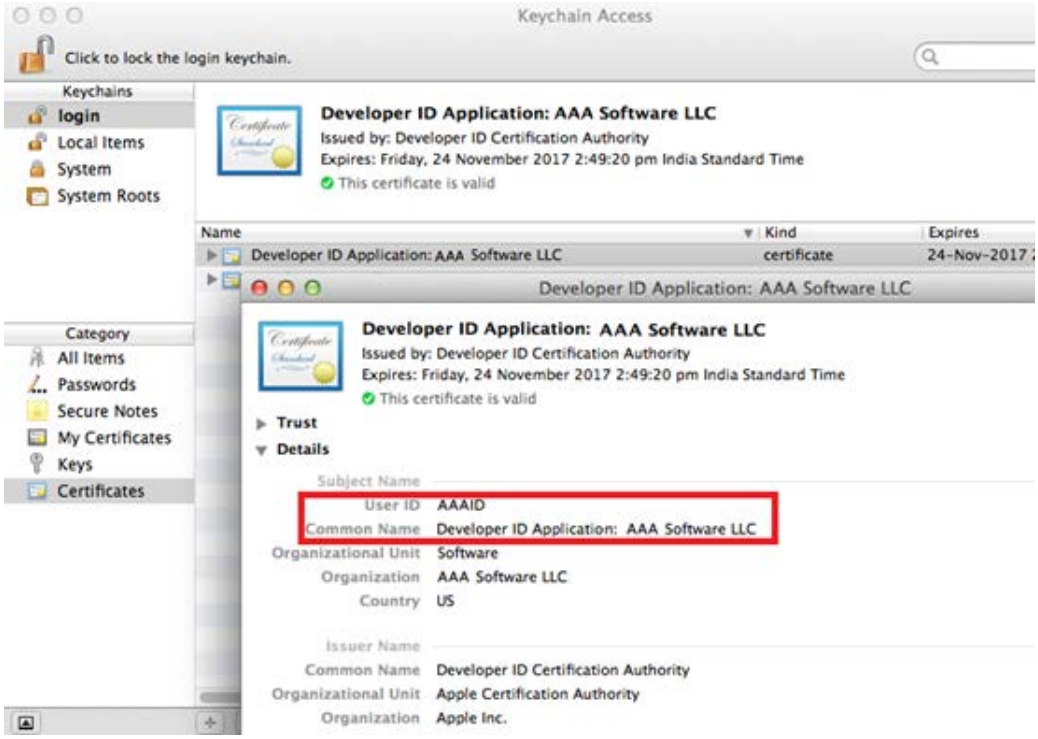


Figure 6-3: User ID and Common Name in Keychain Access

- **OUTPUT_DIR**—Specify the output directory where the helper tool will be copied.
- **SDK_PATH**—Specify the OS or OS X SDK directory. If the OS or OS X SDK is available in the default path, you can leave this blank. If you are logged in on an OS X 10.9–based machine, SDK 10.9 is the default SDK. If you want to specify a separate SDK, or if the SDK is not installed, provide the absolute path of the SDK. For example:

```
SDK_PATH="/Applications/Xcode.app/Contents/Developer/Platforms/OSX.platform/Developer/SDKs/
    OSX10.9.sdk"
```

c. Change the current directory to the ht-signer directory, and execute the revised `build-helper-tool.sh` script in that directory. The signed helper tool—called `com.flexera.ia.helper`—is created in the output directory that you specified for OUTPUT_DIR.

5. Verify that the helper tool was successfully signed. To learn how, see Verifying that Your Code-Signing Output Files Are Working as Expected on OS or OS X–Based Target Systems.

6. Make the verified, signed helper tool available to all of your InstallAnywhere build machines: Copy the `com.flexera.ia.helper` tool to the `IA_INSTALL_DIR/resource/nativetools/osx` folder on each InstallAnywhere build machine.

Your machines are now ready to code sign your installers. The next time that you build an installer that includes authentication support, InstallAnywhere includes your signed helper tool with your installer.

If you install a new version of InstallAnywhere, or if you are replacing a Developer ID Application certificate with a new one, the aforementioned process needs to be repeated.

Code Signing Your OS or OS X–Based Installers and Including Authentication Support

The process of code signing your authentication wrappers, your installers, and (if applicable) your uninstallers varies, depending on whether you are performing the code-signing step at build time on the InstallAnywhere build machine or on a separate designated code-signing machine.



Caution ▪ *Before you can code sign your installer, you must ensure that your helper tool is code signed and available on your InstallAnywhere build machine. In addition, your Developer ID Application certificate must be in Keychain Access on your code-signing machine. For more information, see* *Adding the Code-Signing Capability to Your InstallAnywhere Build Machines or Code-Signing Machines*.

Option 1: Code Signing the Authentication Wrapper, Installer, and Uninstaller as Part of the Build Process on the InstallAnywhere Build Machine

Once you have added the code-signing capability to one of your InstallAnywhere build machines, you can use that machine to build authenticated installers for OS or OS X–based target systems.



Task ***To configure your InstallAnywhere project to code sign your build output and include authentication support:***

1. In the Advanced Designer, on the **Project** page, click **Platforms**. The **Platforms** view opens.
2. In the **OS X** area, in the **Code Signing** setting, select the **Code Sign the Generated Installer** check box.
3. Specify the location and password of the certificate. The use of build-time variables for the certificate location and password is highly recommended for security purposes.
   a. For the **PKCS #12 File** setting, specify the fully qualified path for your PKCS #12 file (.p12).
   b. In the **Keystore Password** setting, specify the certificate's password.

   Note ▪ *The certificate that you specify must be the same Developer ID Application certificate that was used to add the code-signing capability to your machines, as described in* *Adding the Code-Signing Capability to Your InstallAnywhere Build Machines or Code-Signing Machines*.
4. Under the **Authentication** category, in the **Requires an Administrator Name and Password to Install** setting, select **Yes**.
5. Optionally change the value of the **Always Show GUI** setting.

When you build OS or OS X–based installers, they are code signed and include authentication support. Monitor the stderr and stdout streams for any code-signing errors that occur at build time. The code-sign command should not exit with a non-zero exit code.

Before you release these installers, test them and verify that they are properly built. To learn how, see Verifying that Your Code-Signing Output Files Are Working as Expected on OS or OS X–Based Target Systems.

Option 2: Building an Installer with Authentication Support on the InstallAnywhere Build Machine and then Code Signing the Authentication Wrapper and Installer on a Designated Code-Signing Machine

Once you have added your verified, signed helper tool to an InstallAnywhere build machine and prepared your separate designated code-signing machine, you can build and code sign authenticated installers for OS or OS X–based target systems.



Task ***To configure your InstallAnywhere project to include authentication support in your OS or OS X–based installers:***

1. In the Advanced Designer, on the **Project** page, click **Platforms**. The **Platforms** view opens.
2. In the **OS X** area, under the **Authentication** category, in the **Requires an Administrator Name and Password to Install** setting, select **Yes**.
3. Optionally change the value of the **Always Show GUI** setting.
4. In the **Code Signing** setting, ensure that the **Code Sign the Generated Installer** check box is cleared.

When you build OS or OS X–based installers, they include authentication support and are ready to be code signed on your designated code-signing machine.



Task ***To code sign the authentication wrapper and the installer on a separate code-signing machine:***

1. Customize the two information property list files (one for the installer and one for the authentication wrapper) in the build output so that they include the user ID from your Developer ID Application certificate:
   a. Extract the `install.zip` file to a folder on your code-signing machine.
   b. Grant write-execute permission to the extracted application so that you can edit values in its `Info.plist` files. The following command demonstrates this:

      ```
      chmod -R 755 <absolute path of install.app>
      ```

   c. Find the installer's `Info.plist` file in the `install.app\Contents\Resources\install.app\Contents` folder, and open it in an editor.
   d. Under the **SMAuthorizedClients** key, replace the **XXXXXXXXXX** string with the user ID from your Developer ID Application certificate. This ID is displayed in the User ID field when you select your certificate in the Keychain Access utility.
   e. Find the authentication wrapper's `Info.plist` file in the `install.app/Contents` folder, and open it in an editor.
   f. Under the **SMPrivilegedExecutables** key, replace the **XXXXXXXXXX** string with the user ID from your Developer ID Application certificate. This ID is displayed in the User ID field when you select your certificate in the Keychain Access utility.
2. Code sign and check the installer (which is in a subfolder of the authentication wrapper; that is, the `install.app\Contents\Resources\install.app` folder):

a. Use the following command line to code sign the installer:

```
codesign --force --deep --timestamp=none --sign <CERTIFICATE_ID> <install.app absolute path>/
    Contents/Resources/install.app
```

where *<CERTIFICATE_ID>* is the value without the **Developer ID Application:** string that is displayed in the Common Name field when you select your certificate in the Keychain Access utility. That is, if the Common Name field displays **Developer ID Application: ABC Software Inc**, use a certificate ID of **ABC Software Inc** in your command line.

b. Verify that your signed installer has been properly built and signed. To learn how, see Verifying that Your Code-Signing Output Files Are Working as Expected on OS or OS X–Based Target Systems.

3. Code sign and check the authentication wrapper (that is, the top-level install.app folder):

a. Use the following command line to code sign the authentication wrapper:

```
codesign --force --deep --timestamp=none --sign <CERTIFICATE_ID> <install.app absolute path>
```

where *<CERTIFICATE_ID>* is the value without the **Developer ID Application:** string that is displayed in the Common Name field when you select your certificate in the Keychain Access utility. That is, if the Common Name field displays **Developer ID Application: ABC Software Inc**, use a certificate ID of **ABC Software Inc** in your command line.

b. Verify that your signed authentication wrapper has been properly built and signed. To learn how, see Verifying that Your Code-Signing Output Files Are Working as Expected on OS or OS X–Based Target Systems.

4. Compress the top-level install.app folder back into an `install.zip` file.

The resulting install.app file includes authentication support for the installer. Note that the only way to remove a product whose installer was code signed using this method is to drag it to the Trash.

Verifying that Your Code-Signing Output Files Are Working as Expected on OS or OS X–Based Target Systems

Regardless of which method you use to prepare your code-signed installers, at various points in the process, it is important to verify whether the target file has been properly code signed before proceeding with the next step of the process.

It is also crucial to test the final output on a clean machine.

If you encounter issues while testing your output files, you can verify that the trust relationship between the helper tool and the authentication wrapper are intact.

Verifying that the Helper Tool, the Authentication Wrapper, and the Installer Have Been Successfully Signed

Regardless of which method you use to prepare your code-signed installers, at various points in the process, it is important to verify whether the target file has been properly code signed before proceeding with the next step of the process.

To verify that the helper tool, the authenticated wrapper, or the installer was signed properly, use the following commands in Terminal:

Table 6-13 ▪ Commands for Verifying that Various Files Were Signed Properly

Command	Expected Result
$> codesign -vvv <path_to_signed_file>	This should specify that all designated requirements are satisfied.
$> codesign -dvvv <path_to_signed_file>	This should specify the code-signing information, including details about the certificate that was used.
$> spctl -a <path_to_signed_file>	If this command exits without any output, Gatekeeper will accept the file.

Testing Your Installer

Before you release your product, it is important to test the final output on a clean machine—one that does not have your Developer ID Application certificate available in the login or system keychains in the Keychain Access utility. Ensure that you are able to successfully install your product when you are logged on to the test machine as a standard user who is not a root user or an administrative user.

Verifying that the Trust Relationship Between the Helper Tool and the Authentication Wrapper Are Intact

In some cases, it is possible that the trust relationship between the helper tool and the authentication wrapper may be broken. Such cases are difficult to identify and debug. One easy way might be to first print out the code-signing requirements of both the helper tool and the authentication wrapper, and compare them. To do so, use the following command:

```
$> codesign -d -r- <path-to-app>
```

where *<path-to-app>* is the path to your helper tool or your authentication wrapper

Run this for both the helper and the authentication wrapper. This produces output on Terminal. Compare the helper tool output with the authentication wrapper output, and look for any mismatches. The identifier portion can differ, but the other portions should match.

The following output samples demonstrate an example of a working relationship.

Helper Tool output:

```
iadevs--mini:Build iadev$ codesign -d -r- com.flexera.ia.helper
designated => anchor apple generic and
identifier "com.flexera.ia.helper" and
(certificate leaf[field.1.2.840.113635.100.6.1.9] /* exists */ or
certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and
certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and
certificate leaf[subject.OU] = XXXXXXXXXX)
```

Authentication wrapper output:

```
iadevs--mini:NoVM iadev$ codesign -d -r- install.app

designated => identifier "com.flexera.authenticator" and

anchor apple generic and

certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and

certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */

and certificate leaf[subject.OU] = XXXXXXXXXX
```

You can also optionally use the following command to obtain additional debugging information:

```
$> otool -X -s __TEXT __info_plist com.flexera.ia.helper | xxd -r
```

This assumes that the helper tool (`com.flexera.ia.helper`) is present in the current directory, and that it can be used to verify that the `info.plist` file that is injected into the helper tool was customized properly.

Troubleshooting Tips for Code-Signing and Authentication Support for OS or OS X–Based Target Systems

The following table provides tips for various issues that you may encounter when preparing or testing code-signed installers with authentication support.

Table 6-14 ▪ Troubleshooting Tips for Code-Signing and Authentication Support (Sheet 1 of 2)

Issue	Tip
While you are code signing one of your files, you encounter an error about the code-signing identity not being found.	This error indicates that the certificate is missing from the login keychain, or that the spelling of the specified common name is incorrect.
When you are building a code signed installer with authentication support from within InstallAnywhere (option 1), the stderr/stdout messages indicate that your code-sign command exited with a non-zero exit code.	Check to ensure that the certificate is found in the login keychain on your InstallAnywhere build machine. Also verify that you are not logged in as the root user on the machine when you are building and signing; you should be logged in as the same user that was logged in when the certificate was added to the login keychain.

Table 6-14 ▪ Troubleshooting Tips for Code-Signing and Authentication Support (cont.) (Sheet 2 of 2)

Issue	Tip
After performing all of the building and code-signing steps, your installer does not launch when you try to run it.	This issue may occur in the following scenarios: • One or more of the code-signing requirements are not met on the machine that you are using for code signing (either your InstallAnywhere build machine or a designated signing machine). To learn more, see Requirements for Code-Signing Support for OS or OS X–Based Installers. • The helper tool, the authentication wrapper, or the installer was not properly signed. Ensure that you perform the verification steps after signing each file. • The code signature identities of the helper tool and the authentication wrapper do not match, or the trust relationship between them is not intact. • Your login keychain is locked. Ensure that it is not locked. To learn more, see Verifying that Your Code-Signing Output Files Are Working as Expected on OS or OS X–Based Target Systems. The Console utility that is built into OS or OS X may provide relevant clues. (This is different from Terminal; the Console utility is the OS-level log-reporting application.) When you try to launch the installer but it fails to launch, check the Console for errors. If you still cannot identify the problem, try removing the certificate and corresponding private key from your keychain, re-adding them, and then rebuilding and re-signing as needed.
When the code-signing step occurs, a dialog box opens, prompting for administrative credentials.	On some machines, when the certificate is added to the login keychain, only certain applications have access to it. If you try to code sign in this case, the prompt for credentials is displayed. To avoid this prompt, ensure that the certificate has been granted access to be used by all applications: 1. In Keychain Access, right-click the certificate and then click **Get Info**. 2. On the **Access Control** tab, click the **Allow all applications to access this item** option.
Your installer successfully installs your product, but your uninstaller does not uninstall it.	Being unable to uninstall the product is a limitation of the option 2 method of code signing, where signing is performed on a separate designated code-signing server. The only way to remove a product whose installer was code signed using this method is to drag it to the Trash.

OS X Notarization

The Apple notary service is a robotized framework that outputs your product for malign content, checks for code-signing issues, and returns the outcomes to you rapidly.

Starting in OS 10.14.5, all new or updated kernel extensions and all products from developers new to dispersing with Developer ID must be notarized so as to run. Starting in OS 10.15, by default, the notarization is required for all products.

Notarization likewise ensures your users if your Developer ID signing key is uncovered. The notary service keeps up a review trail of the product appropriated utilizing your signing key.

InstallAnywhere now supports notarizing OS or OS X–based installers with a Developer ID Application certificate during the build time.

Introduction

In InstallAnywhere 2026 R1, you can automatically notarize your application with ease.

The process of notarizing your authentication wrappers, your installers, and your uninstallers varies, depending on whether you are performing the notarizing step at build time on the InstallAnywhere build machine or on a separate designated notarization machine.

You can specify whether and how you want InstallAnywhere to notarize your OS X–based installer at build time. If you notarize the installer, end users can download your installer from outside the App Store and install the product without being blocked by the Gatekeeper. The settings in this area are:

- **Notarize the Generated Installer**
- **Developer Username**
- **Developer App Specific Password**
- **Notarization Response Timeout (min)**

Prerequisites

Apple's notary service requires the following:

- Notarization requires Xcode 10 or later.
- Building an application for notarization requires OS 10.13.6 or later.
- Stapling an application requires OS 10.12 or later.
- Code-signing is mandatory for all the executables.
- Link against the OS 10.9 or later SDK.
- Using any new certificate, requires Helper Tool.



Important ▪ *Only when using a new certificate, you are required to use the Helper Tool. For more information, see Prepare the Helper Tool Dialog Box.*

Notarizing your OS or OS X–Based Installers and Including Authentication Support

The process of notarizing your authentication wrappers, your installers, and your uninstallers varies, depending on whether you are performing the notarizing step at build time on the InstallAnywhere build machine or on a separate designated notarization machine.



Important ▪ *If you want to notarize the application, you must code sign your installer.*



Task ***To configure your InstallAnywhere project to notarize your build output and include authentication support:***

1. In the Advanced Designer, on the **Project** page, click **Platforms**. The **Platforms** view opens.
2. In the **OS X** area, in the **Code Signing** setting, select the **Code Sign the Generated Installer** check box.
3. Specify the location and password of the certificate.
    a. For the **PKCS #12 File** setting, specify the fully qualified path for your PKCS #12 file (.p12).
    b. In the **Keystore Password** setting, specify the certificate's password.
4. In the **OS X** area, in the **App Notarization** setting, select the **Notarize the Generated Installer** check box.
5. Specify the username and password for your notarization.
    a. For the **Developer Username** setting, specify the apple's username for your notarization.
    b. In the **Developer App Specific Password** setting, specify the app specific password of your application.
    c. In the **Team Identifier** setting, specify the team identifier Information from the Apple Developer certificate.
    d. In the **Notarization Response Timeout(min)** setting, specify the notarization response timeout for your application.
    e. In the **Notarization Process Delay Interval(sec)** setting, specify the notarization process delay wait time for your application.

*For more information on **App Notarization** setting, see App Notarization.*

Command for Notarization

The following are examples of notarization:

- Commands for Notarization (Xcode 13 and Above)
- Commands for Notarization (Below Xcode 13)

Commands for Notarization (Xcode 13 and Above)

The following are examples of commands for notarization for Xcode 13 and above:

Table 6-15 ▪ Commands for Notarization (Xcode 13 and Above)

Command	Example

Send notarization	`xcrun notarytool submit <”installer zip”> --keychain-profile <”keychain profile”> --wait`
	or
	`xcrun notarytool submit <”installer zip”> --wait --apple-id <"username"> --password <”app specific password”> --team-id <”Team Identifier”>`

Get notarization logs	`xcrun notarytool log <RequestUUID> --keychain-profile <”keychain profile”> <logfile.json>`
	or
	`xcrun notarytool log <RequestUUID> --apple-id <"username"> --password <”app specific password”> --team-id <”Team Identifier”> <logfile.json>`

Commands for Notarization (Below Xcode 13)

The following are examples of commands for notarization for below Xcode13:

Table 6-16 ▪ Commands for Notarization (Below Xcode 13)

Command	Example
Send notarization	`xcrun altool --notarize-app --primary-bundle-id <"test.zip"> --username <"username"> --password <"password123"> --file <"filename.zip">`
Check status	`xcrun altool --notarization-info <RequestUUID> -u <"username"> -p <password>`
Staple ticket to app	`$ xcrun stapler staple "Notepad.app"`
Check notarization status	`spctl -v -a <app name>`

Requiring Elevated Privileges for OS or OS X–Based Installers and Uninstallers

If you want your OS or OS X–based installers and uninstallers to install files to and remove files from locations where write permissions are restricted for standard users, you can configure your project to require authentication. When authentication is required and standard users who are not root users or administrative users with adequate privileges try to launch your installer or uninstaller, they are prompted to enter an administrator name and password in order to proceed.

Note that in order to require authentication for OS or OS X–based installers and uninstallers, they must be code signed. To learn more, see About Authentication and Code-Signing Support for OS or OS X–Based Installers.

Determining Whether an Installation Was Successful

You can use InstallAnywhere's built-in variable `$INSTALL_SUCCESS$` to receive confirmation that an installation was successful. There are four possible values for the `$INSTALL_SUCCESS$` variable:

- **SUCCESS**—Installation was successful.
- **WARNING**—An error that doesn't prevent installation from happening but might not be the desired behavior. If an action is trying to perform an activity and, due to restrictions or incorrect input parameters, the action is unable to perform successfully, a `WARNING` results. For example, if an action is trying to delete a protected file, or if the Expand Archive action is provided with an archive of an unknown format, shortcuts or an alias may be created because of existing ones of the same name or are named to be the same as the directory name. These actions report `WARNING` in the log files as a way of providing additional information.
- **NONFATAL_ERROR**—A non-fatal error that prevents some part of the installation from completing, but does not make the entire installation fail. In particular, a non-fatal error deals with specific actions that failed to perform due to some critical errors originating during execution. Examples of non-fatal errors include: an execute script action failure, a create shortcut action failing on a different OS, or, in case of a `RUNSQL` script action, the server type or a missing delimiter causes the action to fail. In such cases, the offending actions log the `ERROR` in log entries.
- **FATAL_ERROR**—An error that causes the entire installation to fail. This state represents the abort status of the installers. For example, if upgrades fail, a severe thread exception causes the installer to shut down or initiates cancellation of the installers. The rollback feature is tied to this error type per the settings available in the designer. The custom code allows the user to set the `FATAL_ERROR` using the `FatalInstallationException` class available in the API.

Similarly, you can use the variable `$UNINSTALL_SUCCESS$` to determine the success of the uninstaller.

Troubleshooting Issues with Installers

There are several methods available to debug InstallAnywhere-generated installers. Deciding 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-generated installers use LaunchAnywhere technology. Along with many convenient features for end users (double-clickable launchers, built-in-like user experience), LaunchAnywhere launchers provide various built-in debugging features.

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.

- Select the Optimize Installer Size by Platform and Tags check box (Advanced Designer > Build page > Build Installers view > Build Configurations tab > Distribution subtab).
- 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-Based Installers

InstallAnywhere creates a single UNIX-based installer without a bundled VM that can be run on most UNIX-based platforms. The installers with bundled VMs are built specifically for each platform and may be optimized for the platform. The UNIX-based installer without a bundled VM, however, contains all the resources that are needed for every UNIX-based platform. Having all of the resources for the various UNIX-based platforms can make this UNIX-based installer without VM larger than the optimized installers for individual UNIX-based platforms that actually include a VM. However, the UNIX-based installer without VM will be smaller than the UNIX-based 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.

Debugging During Installer Development

When you are debugging an installer at design time, you can use various debugging features. InstallAnywhere provides two primary types of debug output:

- **Installer Debug Output**—This output provides information that is relevant to debugging issues with the installer run time on various platforms. To obtain this output, define values for the **Send stderr to** and **Send stdout to** settings in your project (Advanced Designer > Project page > General Settings view > Logs area). These settings let you direct debug output to the console or to a file or by setting `lax.stderr` and `lax.stdout` properties for a launcher. To learn how to obtain the debug output after you have created your installer, see Debugging During Post-Development.
- **Output from the Output Debug Information Action**—The data that this action returns is useful for debugging the logic of your installer. For example, you can use this feedback to determine the values of InstallAnywhere variables, magic folders, and Java properties. This can help ascertain the cause of unexpected file deployment/panel flow events. It also lets you choose what type of debug output you receive.

Directing Installer Debug Output

At the project level, InstallAnywhere enables you 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 enables you to select the output data that you want to include. Use an Output Debug Information action to obtain feedback on the logic of your installer. Configure the log-related settings in your project to debug errors and failures in the installer's run time that may be dependent on the environment of the target system.



Task ***To direct debug output to the console:***

1. In the Advanced Designer, on the **Project** page, click **General Settings**. The **General Settings** view opens.
2. In the **Log Settings** area, in the **Send stderr to** and **Send stdout to** settings, enter the following value:

   `console`



Task ***To direct debug output to a file:***

1. In the Advanced Designer, on the **Project** page, click **General Settings**. The **General Settings** view opens.
2. In the **Log Settings** area, in the **Send stderr to** and **Send stdout to** settings, enter a file name or a limited set of InstallAnywhere variables.



Note ▪ *The default location for debug output that is sent to a file is the directory in which the installer resides.*

The values that you enter in the **Send stderr to** and **Send stdout to** settings 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.

Note the following information when you are working with debug output settings on the Project page:

- 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 that are permitted, they resolve to empty strings at run time.
- OS or 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** settings before the final build of the product installer. As an alternative, leave the output intact to make post-development debugging easier.
- If 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 enables you to select the output data that you want to include; it also lets you set destination of the output (console or file). The values that are produced in the debug output depend in part on when the action is performed as part of the installation.

The debug information is useful for investigating and resolving problems with your installer. If errors that are associated with 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 that are used by the InstallAnywhere rules. For errors that are related to the run-time launcher or installer, configure the **Send stderr to** and **Send stdout to** settings in your project. For more information, see Directing Installer Debug Output.



***Note** ▪ The **Send stderr to** and **Send stdout to** settings also generate debug information; however, the debug information obtained in that manner is different, in content, from the information that the Output Debug Information action provides.*



Task ***To use the Output Debug Information action:***

1. In the Advanced Designer, on the **Sequence** page, click **Post-Install**. The **Post-Install** view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. On the **General** tab, click the **Output Debug Information** action, and then click the **Add** button. InstallAnywhere adds the action to the Pre-Install Action List.
4. In the customizer at the bottom of the view, on the **Properties** tab, select the output data items that you want to include in the debug information output.
5. Configure the **Output destination** settings:
   - To send the debug output to the console, select the **Output to the console (stderr)** check box.
   - To send the debug output to a file, select the **Output to a file** check box, and in the in the **Path** box, enter the path for the file.

     The value for the path can be a file name, a relative path, or an absolute path. You can optionally include InstallAnywhere variables.

     If you specify a file name without a path, the debug information is written to the location that is defined by *$USER_INSTALL_DIR$*.

Debugging Using Display Message Panel

Often, it is desirable to debug some portions of an installation during installer development. InstallAnywhere lets you to add a Display Message panel that can display specific InstallAnywhere variable values at run time; you can use this Display Message panel to debug your installer.

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. Rebuild and 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 Executable Files

Because InstallAnywhere-generated installers use LaunchAnywhere executable files, 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 file:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. In the **Visual Tree**, select LaunchAnywhere action that you want to configure.
3. In the customizer at the bottom of the view, click the **Edit Properties** button.
4. In the lax.stderr.redirect and lax.stdout.redirect rows, double-click the field in the Value column, and enter the following value:

   ```
   output.txt
   ```

When you are done troubleshooting your installer, edit the .lax file and remove the debugging behavior.

Debugging During Post-Development

InstallAnywhere includes support for debugging installers on various target platforms without access to the Advanced Designer (post-development debugging). These methods override the debug settings of the current installer (not including the data from Output Debug Information actions). That is, if you configured stderr or stdout settings in the General Settings view on the Project page or you are using `lax.stderr.redirect`/`lax.stdout.redirect` settings for your LaunchAnywhere launcher, those settings are supplanted by the override setting.



Note ▪ *None of these methods has any effect on the data that an Output Debug Information action generates.*

Debugging a Windows-Based Installer

To view or capture the debug output from a Windows-based installer, hold down the CTRL key immediately after launching the installer until a console window opens. Before exiting the installer, copy the console output to a text file for later review.

On some Windows-based systems, run the installer once while holding the CTRL key down, resetting the scroll-back buffer for the console window, and then quit and run the installation again.

If you encounter problems while capturing the console output, try a slightly more convoluted method. (This often is the case on Windows 9x systems 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 panel, when given the opportunity to choose a language or to go to the Windows `temp` directory, look for a temp folder that starts with a vertical bar (|), followed by many numeric digits (for example—**I1063988642**). Be sure it is the most recent directory by sorting the directories by their modified date. Open the directory; it should contain a file called `sea_loc`. Delete this file. Now return to the installer, click OK. 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 in 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, the same directory that contains the .lax file should contain an `output.txt` file. The `output.txt` file contains the same information that was 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 is being used) 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 that are 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 OS or OS X–Based Installers

InstallAnywhere-generated installers use the standard output layers on OS or OS X target systems to display output. To gather debugging output from a OS or OS X–based 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, press Control while clicking the installer (or right-click the installer) and then click Show Package Contents. Inside the Contents folder, you will see an XML file named `Info.plist`. You'll need to change the contents of the file. Initially, it contains the following:

```
<key>lax.stderr.redirect</key>
<string></string>
<key>lax.stdout.redirect</key>
<string></string>
```

Change the file to this:

```
<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.

Reviewing Debug Information

InstallAnywhere produces debug information in two distinct ways:

- Installer debug output is generated when you direct debug output from the installer (that is, when you configure the stderr or stdout settings in the General Settings view on the Project page).
- 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 2013
Version: 14.0

-------------------------------------------------------------------------

Fri Jun 27 17:13:04 CDT 2013

Free Memory: 24575 kB
Total Memory: 22581 kB

No arguments
```

```
java.class.path:

C:\Program Files\InstallAnywhere 2026 R1\resource
C:\Program Files\InstallAnywhere 2026 R1\resource\swingall.jar
C:\Program Files\InstallAnywhere 2026 R1\resource\compiler.zip
C:\Program Files\InstallAnywhere 2026 R1\IAClasses.zip
C:\Program Files\InstallAnywhere 2026 R1\lax.jar
C:\program files\InstallAnywhere 2026 R1\jre\lib\rt.jar

ZGUtil.CLASS_PATH:

C:\Program Files\InstallAnywhere 2026 R1\resource
C:\Program Files\InstallAnywhere 2026 R1\resource\swingall.jar
C:\Program Files\InstallAnywhere 2026 R1\resource\compiler.zip
C:\Program Files\InstallAnywhere 2026 R1\IAClasses.zip
C:\Program Files\InstallAnywhere 2026 R1\lax.jar
...
java.version        = 1.5.0
java.vm.vendor      = Sun Microsystems Inc.
java.class.version  = 45.3
java.home           = c:\program files\InstallAnywhere 2026 R1\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 that you select in the action's customizer.

InstallAnywhere Variables

The content in the InstallAnywhere Variables section changes depending on settings that were made within the project. The output is sorted alphabetically and shows the contents of all of the available InstallAnywhere variables, including special variables (such as `$USER_INSTALL_DIR$`), properties that read from the end user's environment, Java properties (denoted by variable names that begin with the prefix **prop.**), InstallAnywhere installer properties (denoted by variable names that begin with the prefix **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

The Magic Folders 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 ($X_USER_APPLICATIONS$)=/Users/
...
```

Java Properties

The output includes InstallAnywhere installer properties (denoted by variable names that start with the prefix **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.nl.env.openwinhome=/usr/openwin
java.vendor=Sun Microsystems Inc.
lax.nl.env.dtend-usersession=raman-bay-0
lax.nl.env.StartDtscreenPyro=StartDtscreenPyro
...
```

Visual Tree

The Visual Tree section lists the names of all of 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

The Component Tree 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

The Pre-install Actions 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

The Post-install Actions 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 ▪ *For reference information on the Output Debug Information action, see General Actions.*

Troubleshooting Issues with OS or OS X–Based Magic Folders

The following table describes the behavior that is associated with both authenticated end users and regular end users on OS or OS X–based systems.

Table 6-17 ▪ OS or OS X Behavior (Sheet 1 of 2)

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
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

Table 6-17 ▪ OS or OS X Behavior (cont.) (Sheet 2 of 2)

Magic Folders	User Without Administrative Privileges	User Without Administrative Privileges Post- Authentication
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. The OS or OS X Choose Folder panels have aliases to such locations as Desktop or Home. When an end user selects these from a built-in OS or 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.*

Troubleshooting EBCDIC Encoding Issues

InstallAnywhere-generated installers work with EBCDIC-built-in systems. You can build installers for these systems with InstallAnywhere on an ASCII-built-in 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-built-in target system on a Windows-based 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 that is 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 built-in encoding. To make ASCII-format changes to a text file on an EBCDIC-built-in 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.

Exit Codes

This section of the documentation describes exit codes for the build and for InstallAnywhere-generated installers.

Table 6-18 ▪ Exit Codes

Section	Description
Build Exit Codes	Lists the exit codes that you might encounter at build time.
Installer Exit Codes	Lists the exit codes for InstallAnywhere-generated installers with descriptions for each code.

Build Exit Codes

The command-line build tool returns an exit code based upon the results of the build as follows.

Table 6-19 ▪ Build Exit Codes (Sheet 1 of 5)

Code	Status
Resource Related	
cancelled1	Missing resources, build abort canceled 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

Table 6-19 ▪ Build Exit Codes (cont.) (Sheet 2 of 5)

Code	Status
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
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
Errors with Digitally Signing Windows-Based Installers	
801	To resolve this error, check the values that you entered in the Digital Signing area in the Platforms view on the Project page, and make corrections as needed. To learn more, see Digitally Signing Windows-Based Installers.
802	To resolve this error, check the values that you entered in the Digital Signing area in the Platforms view on the Project page, and make corrections as needed. To learn more, see Digitally Signing Windows-Based Installers.
803	To resolve this error, check the values that you entered in the Digital Signing area in the Platforms view on the Project page, and make corrections as needed. To learn more, see Digitally Signing Windows-Based Installers.

Table 6-19 ▪ Build Exit Codes (cont.) (Sheet 3 of 5)

Code	Status
804	To resolve this error, check the values that you entered in the Digital Signing area in the Platforms view on the Project page, and make corrections as needed. To learn more, see Digitally Signing Windows-Based Installers.
805	To resolve this error, check the values that you entered in the Digital Signing area in the Platforms view on the Project page, and make corrections as needed. To learn more, see Digitally Signing Windows-Based Installers.
806	This build error occurs if InstallAnywhere encounters an issue reading the certificate file that is configured for digitally signing Windows-based installers. To resolve this error, check the values that you entered in the Digital Signing area in the Platforms view on the Project page, and make corrections as needed. To learn more, see Digitally Signing Windows-Based Installers.
807	To resolve this error, check the values that you entered in the Digital Signing area in the Platforms view on the Project page, and make corrections as needed. To learn more, see Digitally Signing Windows-Based Installers.
808	This build error may occur if the digital certificate password that was specified in InstallAnywhere does not match the password that was set up for the certificate file. To resolve this error, check the values that you entered in the Password setting (Project page > Platforms view > Windows area > Digital Signing area), and make corrections as needed. If the values in those settings are correct, try regenerating the certificate file (.pfx). To learn more, see Digitally Signing Windows-Based Installers.
809	To resolve this build error, ensure that the password that you entered in the Password setting (Project page > Platforms view > Windows area > Digital Signing area) matches the password that was set up for the certificate file. Also ensure that the build machine can access the timestamp server that is specified in the Digital Signing area. If the settings are configured correctly and the build machine can access the timestamp server, try regenerating the certificate file (.pfx). To learn more, see Digitally Signing Windows-Based Installers.

Table 6-19 ▪ Build Exit Codes (cont.) (Sheet 4 of 5)

Code	Status
810	This build error occurs if the timestamp server is not available to the build machine, and InstallAnywhere cannot timestamp the digitally signed Windows-based installers. To resolve this error, ensure that the build machine can access the timestamp server that is specified in the Digital Signing area. If it cannot, try a different timestamp server. To learn more, see Digitally Signing Windows-Based Installers.
811	This build error may occur if the build machine cannot access the certificate file that was specified in the Certificate File setting (Project page > Platforms view > Windows area > Digital Signing area). To resolve this error, ensure that the specified path and file name are correct, and that all users have read permission for the file. To learn more, see Digitally Signing Windows-Based Installers.
812	This build error occurs if InstallAnywhere is able to digitally sign the Windows-based installers but cannot timestamp the signed files. Skipping timestamping may affect how long your digital signature is considered to be valid. To resolve this error, ensure that a valid and available timestamp server is specified in the Timestamp Server setting (Project page > Platforms view > Windows area > Digital Signing area). If necessary, try a different timestamp server. To learn more, see Digitally Signing Windows-Based Installers.
813	This build error occurs if a SHA-1 digital certificate was configured to be used to digitally sign the Windows-based installers at build time. Windows will not trust files that were signed with SHA-1 certificates if they were signed after January 1, 2016. SHA-2 is favored over SHA-1, which is being deprecated because of the potential for security vulnerabilities. Thus, it is recommended that you replace any SHA-1 certificates in your InstallShield projects with SHA-2 certificates. To replace the SHA-1 certificate with a SHA-2 certificate, specify a SHA-2 certificate in the Certificate File setting (Project page > Platforms view > Windows area > Digital Signing area). To learn more, see Digitally Signing Windows-Based Installers.

Table 6-19 ▪ Build Exit Codes (cont.) (Sheet 5 of 5)

Code	Status
814	This build error occurs if the following conditions are true: ● A SHA-1 digital certificate was configured to be used to digitally sign the Windows-based installers at build time. ● The build cannot timestamp the installers. Skipping timestamping may affect how long your digital signature is considered to be valid. Windows will not trust files that were signed with SHA-1 certificates if they were signed after January 1, 2016. SHA-2 is favored over SHA-1, which is being deprecated because of the potential for security vulnerabilities. Thus, it is recommended that you replace any SHA-1 certificates in your InstallShield projects with SHA-2 certificates. To resolve this error: ● To replace the SHA-1 certificate with a SHA-2 certificate, specify a SHA-2 certificate in the Certificate File setting (Project page > Platforms view > Windows area > Digital Signing area). ● Ensure that a valid and available timestamp server is specified in the Timestamp Server setting (Project page > Platforms view > Windows area > Digital Signing area). If necessary, try a different timestamp server. To learn more, see Digitally Signing Windows-Based Installers.
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

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 following table describes the possible exit codes that may be returned during an installation.

Table 6-20 ▪ Installer Exit Codes (Sheet 1 of 2)

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.
18	The temporary directory `/tmp` is mounted with the 'noexec' option. The installer cannot execute the bundled Java VM. Please use `IATEMPDIR` environment variable or `-tempdir` command line option to specify an alternate directory.
106	Could not find a valid Java Virtual machine to load. You may need to reinstall a supported java virtual machine.
107	The JVM used for this installer seems to have been installed in an unsecured location (non-admin user folder). Installer would now exit. Please reach out to your vendor for more details.
1000	The installation was canceled by the end 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 64-bit Windows system, but installation did not include support for 64-bit Windows systems.
2006	The installation failed because it was launched in a UI mode that is not supported by this installer.
2009	Indicates that the user attempted to launch multiple instances of an installer at the same time even though the installer was configured to prevent multiple launches; the **Prevent multiple launches of an installer at a given time** check box was selected in the General Settings view on the Project page of this installer's InstallAnywhere project.

Table 6-20 ▪ Installer Exit Codes (cont.) (Sheet 2 of 2)

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 file could not be found at the directory that was 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 product registry has been corrupted.
7000	The installation was rolled back due to a fatal exception.
8000	The upgrade was canceled because a newer version of the product is already installed on the target system.
8001	The end user canceled the upgrade.
8002	The upgrade exited because the earlier version of the product could not be successfully uninstalled.



Note ▪ *For information on exit codes from the command-line builder, see Build Exit Codes.*

7

Updating Applications

Installing updates for applications is a more common operation than installing the original release of an application. This makes creating effective, reliable upgrades an important task.

InstallAnywhere enables you to create an upgrade that is packaged as a full installation. End users can run this upgrade in both of the following scenarios:

- If the target system has an earlier version of the product, the earlier version is uninstalled, and then the new version is installed.
- The target system does not have an earlier version of the product, the upgrade installer behaves as a first-time installation and installs the new version of the product.

To learn how to create updates, refer to this section of the documentation.

Requirements for Upgrade Support

To create an upgrade that updates earlier versions of a product (referred to as the *base versions*), the following conditions are required:

- The uninstaller for the base version of the product must have been created in InstallAnywhere.
- The installer for the base version of the product must have had silent mode enabled. That is, the uninstaller must be capable of silently uninstalling the earlier version. This requirement does not apply to OS or OS X–based installers, which do not have support for enabling silent mode.
- The installer for the base version of the product must have been configured to update the InstallAnywhere product registry on target systems. This enables the installer to determine whether any earlier versions of the product exist on the target system. Thus, Yes must be selected for the Update the Product Registry setting in the installer project of the base version (Project page > General Settings view > Product Information area).

  If No is selected for the Update the Product Registry setting in project for the base version of the product, the upgrade behaves as a normal first-time installation.
- The product version that you specify in your project must be updated. The version number for the upgrade must be higher than the version number of any of the base versions.

InstallAnywhere does not have built-in support for downgrades; that is, it does not have support for installing the current version of your product over a future version of your product.

Creating an Upgrade



Task ***The process of creating an upgrade for your product can be broken down into the following tasks:***

1. If you have not already done so, back up the latest version of your InstallAnywhere project in source code control software, or by other methods.
2. Open the latest version of your InstallAnywhere project and change the product version to a newer version number.

   To learn how to specify the version, see Specifying the Product Version.
3. Make the necessary changes to your project, such as adding files, components, and features as needed.
4. Configure the upgrade settings for your project. This includes enabling upgrade support, as well as defining details about the earlier version or versions of your product that need to be updated. For instructions, see the following topics:
   - Configuring Upgrade Settings
   - Enabling Upgrade Support
   - Detecting Installed Earlier Versions that Need to Be Updated
   - Adding Upgrade-Specific Actions to Sequences
5. Build the installers for your project.

Configuring Upgrade Settings

When you are managing a project in InstallAnywhere, you can optionally enable upgrade support so that your installers can remove one or more earlier versions of your product before installing the new version. You can also configure other upgrade-related information, such as which earlier versions should be removed.



Task ***To configure upgrade settings in your project:***

1. In the Advanced Designer, on the **Project** page, click **Upgrades**. The **Upgrades** view opens.
2. Configure the settings in this view as needed.

For information on each of the settings in the Upgrades view, see Upgrades View

Enabling Upgrade Support

In order to include support for upgrades in an installer, you must enable upgrade support in your project. If this is not enabled, the installer does not check target systems at run time to verify whether an earlier version of the product needs to be uninstalled before installing the new version.



Task ***To enable upgrade support in your project:***

1. In the Advanced Designer, on the **Project** page, click **Upgrades**. The **Upgrades** view opens.
2. In the **Upgrades** area, set the **Enable Upgrade Support** setting to **Yes**.

Detecting Installed Earlier Versions that Need to Be Updated

When you are configuring the upgrade settings in your project, you need to specify details about the earlier version or versions of your product that you want to be able to update with the upgrade that you are creating.



Task ***To define which earlier version or versions of your product need to be updated:***

1. In the Advanced Designer, on the **Project** page, click **Upgrades**. The **Upgrades** view opens.
2. Find the **Upgrade Configuration** setting.
3. In the **Upgrade Configuration** setting, click the **Manage Upgrade Configurations** button.The **Manage Upgrade Configurations** dialog box opens.
4. Use the **List of Upgrade Configurations** box to define one or more configurations that specify how your upgrade should detect earlier versions of the product that need to be updated.

   By default, this list shows a **Default Upgrade Configuration** item.

   Perform each of the following tasks as needed:

   - To add a new configuration, click the **Add** button next to the list.
   - To view or edit the settings for a particular configuration, select the configuration and then review and modify the settings below the list as needed.
   - To remove an existing configuration, select it in the list, and then click the **Delete** button.
5. Configure the settings in the **Criterion for Detecting Which Products to Upgrade** area and the **Range of Product Versions to Upgrade** area as needed.
6. Click the **OK** button.

For more information, see Manage Upgrade Configurations Dialog Box.

If upgrade support is enabled in your installer, the installer checks the InstallAnywhere registry on target systems for one or more earlier versions of the product that meet the upgrade configuration conditions that you configured on the Manage Upgrade Configurations dialog box.

Tips on Specifying Details about the Earlier Versions of Your Product

Upgrades that are built in InstallAnywhere need to be able to identify earlier versions of the product that are present on target systems and that are eligible to be updated. You can specify these sorts of details by defining one or more upgrade configurations in your project. An upgrade configuration consists of two main types of data:

Table 7-1 ▪ Managing Upgrade Configurations

Criteria	Notes
Upgrade code or product code	The upgrade code should be consistent across different versions and languages of a family of related products. For this reason, this identifier is often preferred rather than the product code for detecting earlier versions. The product code is sometimes updated for new versions of a product; in other scenarios, it is kept the same across versions. If you want to use the product code instead of the upgrade code for detecting earlier versions, you can specify whether you want to check for the product code that matches the one in the current project, or you can specify the specific product code from an earlier version of the product. Note that if the base installer was created in a version of InstallAnywhere that did not include an upgrade code (that is, if the base installer was created in InstallAnywhere 2013 or earlier), use the product code for detecting that earlier version.
Range of product versions	The version number for the upgrade must be higher than the version number of any of the base versions. That is, the version that you specify in the General Settings view of the project must be higher than the range that you specify for the upgrade configuration. To target a specific earlier version of your product, you can enter the same version number for the minimum and maximum version numbers.

Thus, depending on whether each earlier version of your product has an upgrade code or uses the same or a different product code, you may want to define multiple upgrade configurations on the Manage Upgrade Configurations dialog box.

The following table shows sample upgrade configurations for upgrading to version 5.0.0 of a product. The upgrade is intended to target all of the earlier versions of the product; that is, if any of the earlier versions of the product are present, the upgrade has support for detecting these earlier versions and uninstalling them before installing the new version.

Table 7-2 ▪ Example Upgrade Configurations for an Upgrade Installer

Upgrade Configuration	Detection Criterion	Product Version Range
Default Upgrade Configuration The installer for this version of the product did not have an upgrade code.	Different product code than that of the current project	1.0.0–1.99.999

Table 7-2 ▪ Example Upgrade Configurations for an Upgrade Installer (cont.)

Upgrade Configuration	Detection Criterion	Product Version Range
Upgrade Configuration 1 The installer for this version of the product did not have an upgrade code. However, the same product code has been used for this version and all of the newer versions of the product.	Same product code that is in the current project	2.0.0–2.99.999
Upgrade Configuration 2 The installer for this version of the product had an upgrade code.	Same upgrade code that is in the current project	3.0.0–4.99.999

Adding Upgrade-Specific Actions to Sequences

InstallAnywhere enables you to add actions that perform upgrade-related tasks to the Pre-Install and Post-Install sequences of your project. In addition, InstallAnywhere lets you add Check Running Mode rules to actions and other elements of sequences.

Adding an Upgrade-Specific File or Folder Action to a Sequence

When an end user launches an upgrade a target system that has an earlier version of the product that needs to be updated, the uninstallation of the earlier version of the product occurs after the Pre-Install sequence. Therefore, if you need to perform any file or folder operations such as copying files before the uninstallation occurs, you can use the following upgrade-specific actions in the Pre-Install sequence:

- Copy File Action
- Copy Folder Action
- Delete File Action
- Delete Folder Action
- Move File Action
- Move Folder Action

Note that if you use these Pre-Install actions in installation mode and the target destination is $USER_INSTALL_DIR$, the user's installation directory is created before the Pre-Install sequence.

Similarly, you can use these same actions in the Post-Install sequence.



Task ***To add an upgrade-specific file or folder action for an upgrade to the Pre-Install or Post-Install sequence:***

1. In the Advanced Designer, on the **Sequence** page, click **Pre-Install** or **Post-Install**.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.

3. Click the action you want to add, and then click the **Add** button. InstallAnywhere adds the action to the **Pre-Install Action List**. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. Use the arrow keys to move the action up or down in the action list.
5. Add the Check Running Mode rule:
    a. On the action's customizer, click the **Rules** tab.
    b. Click the **Add Rule** button. The **Choose a Rule** dialog box opens.
    c. Select the **Check Running Mode** rule, and then click the **Add** button.
    d. Select the **Upgrade** option.
    e. Configure the rule as needed.

To specify an action's settings, select the action and use its customizers at the bottom of the view. For detailed information on each action's customizer settings, see Actions.

Adding an Upgrade-Specific Check Running Mode Rule to an Element in a Sequence

You can add a Check Running Mode rule to an action group, action, or other element that runs only in upgrade mode.



Task ***To add an upgrade-specific Check Running Mode rule to an action or other element in a Pre-Install, Install, or Post-Install sequence:***

1. In the Advanced Designer, on the **Sequence** page, click **Pre-Install**, **Install**, or **Post-Install**.
2. Find the action or other element that you want to have the Check Running Mode rule.
3. On the element's customizer, click the **Rules** tab.
4. Click the **Add Rule** button. The **Choose a Rule** dialog box opens.
5. Select the **Check Running Mode** rule, and then click the **Add** button.
6. Select the **Upgrade** option.
7. Configure the rule as needed.

To learn more about configuring a rule, see Building Complex Rule Expressions.

Special Considerations for Upgrade Support

If you are adding upgrade support to your project, note the following details.

Upgrade-Related Variables

The following upgrade-related standard variables are available:

- $IA_UPGRADE_BASE_VERSION$
- $IA_UPGRADE_BASE_LOCATION$

These variables specify version and location information about the earlier version of the product that is being updated during an upgrade.

To learn more, see Standard InstallAnywhere Variables.

Upgrades and the Uninstaller

If an end user runs an upgrade on a target system that has an earlier version of the product that needs to be updated, but its uninstaller is missing from the target system or damaged, the upgrade exits without uninstalling the earlier version or installing the new version.

Upgrades and Rollbacks

A rollback that occurs during an upgrade cannot restore a target system to the state that it was in before the upgrade was installed. That is, if rollback is enabled in an upgrade and the upgrade encounters the FatalInstallException type of exception while installing the new version of the product, the earlier version of the product that the upgrade removed cannot be restored.

Upgrades and File Locking

The upgrade process cannot detect whether the earlier version of the product that is being updated or its uninstaller is locked by another process. Therefore, you may want to add text to the Introduction panel to recommend that users exit any open instances of the product before launching the installer.

Upgrades and Shared Components

If you configure your upgrade to update a shared component of a product whose earlier version is present on a target system, note the following behavior about the upgrade process:

1. The upgrade initiates the uninstallation of the product:
    - As part of this process, the uninstaller removes the standard components.
    - If no other products on the target system reference the shared component, the uninstaller also removes that shared component.
    - If one or more other products on the target system reference the shared component, the uninstaller leaves the shared component on the target system.
2. The upgrade installs the new version of the product. As part of this process, the upgrade overwrites the shared components that were not removed. As a result of the upgrade of the shared component, unexpected results may occur in one or more of the products that reference the shared component.

Upgrades and Instance Management

It is possible to have an installer that includes both upgrade support and support for installing multiple instances of the product on the same system. If an end user has multiple earlier versions of the product installed when launching an upgrade installer for a newer version of the product, the installer performs an upgrade; it does not launch maintenance mode to maintain the existing versions.

Depending on how you configure the upgrade support in your project, you can either have the upgrade update all of the earlier instances that are installed on target systems, or enable end users to choose which instance to update.

Upgrades and Merge Modules

If your project includes one or more merge modules as well as upgrade support, the installer can directly handle upgrades of the merge modules—the uninstallation of the earlier versions of merge modules as well as the installation of the new version. To support this behavior, the parent project must be configured to uninstall the merge modules. That is, if you are using a dynamic merge module, the **Uninstall Merge Module when parent is uninstalled** check box must be selected. This setting is on the Install tab of the Dynamic Merge Module customizer in the parent project (Organization page > Modules view). You can also select this check box on the Install Merge Module action customizer in the Install sequence.

Note the following behavior about upgrades and merge modules:

- If a parent project is configured to uninstall a merge module and if the resulting installer detects an earlier version of the product that needs to be updated, the upgrade installer removes the earlier version of the merge module and also installs the new version.
- If the parent project is not configured to uninstall a merge module, the parent installer leaves the earlier version of the merge module as is on target systems during upgrades. In addition, it does not install the new version of the merge module.

Note that the aforementioned behavior occurs regardless of whether upgrade support is enabled in the merge module project. In addition, it occurs regardless of whether upgrade configurations are configured in the merge module project.

8

Creating Virtual Appliances

Virtual appliances are prebuilt, preconfigured, ready-to-run application solutions that are packaged along with an optimized operating system called a just enough operating system (JeOS).

InstallAnywhere enables you to leverage your existing investments in building installers for your software and to use the same InstallAnywhere project to build virtual appliances for cloud platforms such as Amazon EC2 and VMware vSphere.



Important ▪ *InstallAnywhere does not have support for resolving build-time variables for any of the properties in the Build Appliances view on the Build page.*

About Virtual Appliances

InstallAnywhere enables you to leverage your existing investments in building installers for your software and to use the same InstallAnywhere project to build virtual appliances for cloud platforms such as Amazon EC2 and VMware vSphere.

Virtual appliances are prebuilt, preconfigured, ready-to-run application solutions that are packaged along with an optimized operating system called a just enough operating system (JeOS). Only necessary operating system components are included. A virtual appliance is deployed as a subset of a virtual machine running on virtualization technology, such as VMware vSphere or Amazon EC2.

Components of a Virtual Appliance

The following diagram presents the components of a typical virtual appliance (a single-VM-based virtual appliance).

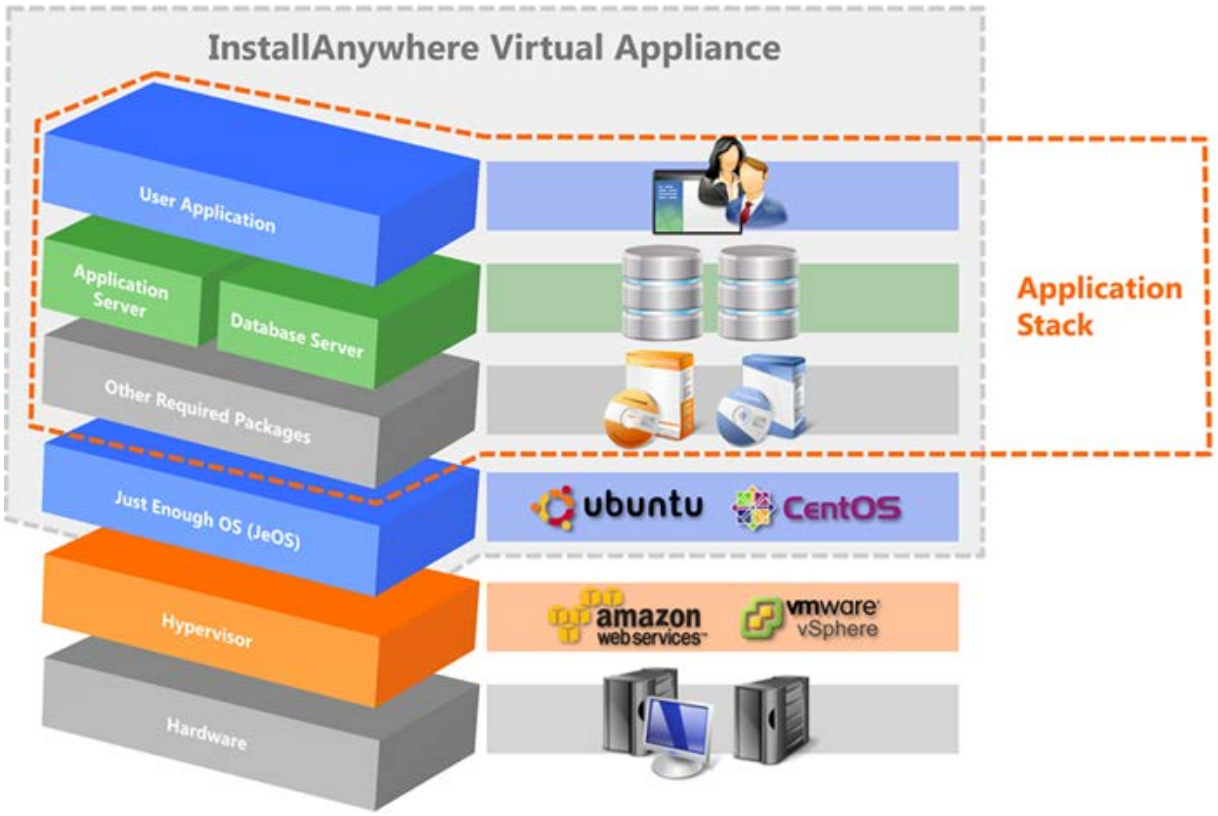


Figure 8-1: Components of a Virtual Appliance

The virtual appliance includes a thin operating system layer called just enough operating system (JeOS). It also includes the required operating system packages and the final application. The self-contained virtual appliance can run directly on hypervisors such as VMware vSphere or Amazon EC2.

InstallAnywhere also includes support for a virtual appliance that has multiple virtual machines; a multi-tier virtual appliance can be deployed to VMware vCenter servers.

Advantages of Using Virtual Appliances

There are many advantages of using virtual appliances instead of regular installers to deploy software. Using virtual appliances provides the following benefits:

- Using virtual appliances enables you to deploy fully configured and ready-to-run software.
- Using virtual appliances eliminates problems with installation and configuration, such as software or driver compatibility issues, or application and library conflicts. End users simply download a single file and run the application.
- Using virtual appliances reduces the testing matrix of operating system platforms.
- Using virtual appliances reduces the resources that are required for maintenance.
- Virtual appliances are usually built to host a single application; virtual appliances are therefore useful in deploying applications on a network.

- Deploying your products as virtual appliances helps you reach new customer segments.

Advantages of Using InstallAnywhere to Create Virtual Appliances

Using InstallAnywhere to create virtual appliances provides additional advantages.

- With the increasing popularity of virtual appliances, you can leverage your existing investment in your installers and use the same project, with almost no change, to create virtual appliances.
- There is no learning curve for developers who are new to virtual appliances.
- You can create customized VM templates that can be used for virtual appliances.
- It enables you to quickly and easily provide demos of your enterprise-class products to your customers.
- You can produce OVF 1.1 compliant appliances for VMware.

Supported Hypervisors and Platforms for Virtual Appliances

InstallAnywhere supports the creation of virtual appliances on VMware vSphere 5 and Amazon EC2.

- **VMware vSphere/vCenter 5**—VMware's cloud computing virtualization operating system. Use this hypervisor to create a virtual appliance for an enterprise/private cloud.
- **Amazon EC2**—Part of Amazon.com's cloud computing platform. Use this hypervisor to create a virtual appliance for the public cloud.

The following table shows the supported operating systems on these hypervisors.

Table 8-1 ▪ Supported Platforms and Hypervisors

Supported Hypervisors/Cloud Infrastructure	Supported Operating Systems
VMware vSphere 5/vCenter	• CentOS 6.2 and 6.3 (32-bit and 64-bit) • Red Hat Enterprise Linux 6.4 (32-bit and 64-bit) • Red Hat Enterprise Linux 6.3 (64-bit) • Ubuntu 13.04 (32-bit and 64-bit) • Ubuntu 12.10 (32-bit and 64-bit) • Ubuntu 12.04 (32-bit and 64-bit) • Ubuntu 11.10 (32-bit and 64-bit)
Amazon EC2	• Ubuntu 11.10 32-bit • Ubuntu 12.04 32-bit

Note that multi-tier virtual appliances can be deployed to VMware vCenter servers.



Note ▪ *You can obtain the ISO files for these operating systems from the vendors' Web sites:*

http://www.centos.org/

http://www.redhat.com/

http://www.ubuntu.com/download/server

Virtual Appliance Creation Workflow

When using InstallAnywhere to create a virtual appliance for either VMware vSphere or Amazon EC2, you need to perform the steps listed in the following topics:

- Creating a VMware vSphere 5 Virtual Appliance
- Creating an Amazon EC2 Virtual Appliance

The following table lists the main steps in the virtual appliance creation workflow and provides links to topics that provide additional information on how to perform each step.

Table 8-2 ▪ Virtual Appliance Creation Workflow

#	Step	Related Topics
1.	**Open an InstallAnywhere Project**	Open an existing InstallAnywhere project that contains the virtual application files and an installer. ● Opening an Existing Project
2.	**Choose a VM Template**	Choose a VM template on the VM Configuration tab of the Build Appliances view on the Build page. VM templates are listed here only if you have obtained them and copied them to the `$IA_HOME$/resource/baseline_vms` directory. ● Obtaining VM Templates for Virtual Appliances ● Downloading VM Templates ● Creating Custom VM Templates

3.	**Customize the Virtual Appliance Configuration**	Create an appliance configuration and customize its settings on the Appliance Configuration and VM Configuration tabs of the Build Appliances view on the Build page.
		● Creating a VMware vSphere 5 Virtual Appliance
		  ● Configuring the Appropriate Appliance Output and Type for VMware-Based Virtual Appliances
		● Creating an Amazon EC2 Virtual Appliance
		  ● Support for Elastic Block Store (EBS) for Amazon
		  ● About EC2 and S3 Regions of Deployment for Amazon
		● Working with Virtual Appliance Configurations

4.	**Add Custom Scripts**	● Using Custom Boot and Login Scripts for a Virtual Appliance

Table 8-2 ▪ Virtual Appliance Creation Workflow (cont.)

#	Step	Related Topics
5.	**Add Installers**	• Adding Installers to Your Virtual Appliance
6.	**Add OS Packages**	• Installing Operating System Packages for a Virtual Appliance
7.	**Build the Virtual Appliance**	• Building a Virtual Appliance
8.	**Deploy the Virtual Appliance**	• Deploying a VMware vSphere 5 Virtual Appliance on Demand • Deploying an Amazon EC2 Virtual Appliance on Demand

Obtaining VM Templates for Virtual Appliances

To create virtual appliances, you can start with a VM template. A VM template is a preinstalled virtual machine disk that contains the operating system and critical operating system packages.

Using a preconfigured VM template can speed up the development of a virtual appliance. VM templates also help in streamlining the process of moving multiple applications to the cloud and in ensuring configuration consistency. You can create a VM template once and use it for creating multiple virtual appliances.

You can either download VM templates from the Revenera Web site, or create your own VM templates using the Create InstallAnywhere VM Template Wizard.



Tip ▪ *If you are creating a virtual appliance for VMware vSphere/vCenter, you can use an existing VM, or a VM snapshot that is running on an existing server, for creating a virtual appliance, instead of obtaining a VM template. For more information, see Selecting a VM for a VMware Virtual Appliance.*

Downloading VM Templates

You can download several VM templates as part of the InstallAnywhere installation, or you can download VM templates at any time from the Revenera Web site.

Downloading VM Templates During Installation

You can download several VM templates as part of the InstallAnywhere installation. If you select the Download VM Templates option on the Download VM Templates panel of the InstallAnywhere installer, the VM templates required for creating appliances will be automatically downloaded during installation. By default, this option is not selected.

Downloading VM Templates from the Revenera Web Site

You can download VM templates for both Amazon EC2 and VMware vSphere from the Revenera Web site:

https://www.revenera.com/install/products/installanywhere/installanywhere-files-utilities.html

The following table lists the files that comprise an InstallAnywhere VM template for VMware vSphere and Amazon EC2 hypervisors.

Table 8-3 ▪ InstallAnywhere VM Template Files

Hypervisor	VM Template Files

Amazon EC2	`Ubuntu_11.10_Oneiric_Ocelot.ami`
	`Ubuntu_11.10_Oneiric_Ocelot_bvi.xml`
	Copy these files to the `$IA_HOME$/resource/baseline_vms/amazon_ec2/` directory.

VMware vSphere	`Ubuntu12.04_i386_BVM_2007.zip`
	`Ubuntu12.04_i386_BVM_2007_bvi.xml`
	Extract these files to the `$IA_HOME$/resource/baseline_vms/vmware_esxi/` directory.



Important ▪ *Be sure to copy both the Amazon `.ami` file or VMware `.zip` file and its associated `.xml` file to the specified directory. If you fail to copy the `.xml` file, the VM template will not be listed in the Select InstallAnywhere VM Template list on the VM Configuration tab in the Build Appliances view.*

Creating Custom VM Templates

InstallAnywhere offers a few prebuilt VM templates that you can use to create virtual appliances. If none of the available VM templates matches the environment that you need to support for your virtual appliances, you can create your own VM templates.

About Creating VM Templates

InstallAnywhere provides several InstallAnywhere VM templates, but you can also create your own VM templates using the Create InstallAnywhere VM Template Wizard.

Launching the Create InstallAnywhere VM Template Wizard

The executable file for the Create InstallAnywhere VM Template Wizard is in the following directory:

`$IA_HOME$/VM_Template_CreationUtility`

To launch the Create InstallAnywhere VM Template Wizard, double-click one of the following files:

Table 8-4 ▪ Create InstallAnywhere VM Template Wizard Executable Files

User	Executable
User with administrative privileges on the build system	`CreateVMTemplate.exe`
User without administrative privileges on the build system	`CreateVMTemplate-AsInvoker.exe`

You can use this wizard to create InstallAnywhere VM templates for both VMware vSphere and Amazon EC2.



Note ▪ *You can also use this wizard's command-line interface to create these templates, as described in Creating VM Templates Using the Command Line.*

Default Credentials for the VM Templates that You Create

The following are the default credentials for the root user for the VM templates that are created using the Create InstallAnywhere VM Template Wizard:

Table 8-5 ▪ Default Credentials for User-Created VM Templates

Hypervisor	Amazon EC2	VMware vSphere
User Name	`root`	`root`
Password	`root123`	`root2012Vapp!`

Creating VM Templates Using the Create InstallAnywhere VM Template Wizard

To create a VMware vCenter/vSphere Server template, you need to first obtain VMware Tools, and then you need to use the Create InstallAnywhere VM Template Wizard to create the VM template.

Obtaining VMware Tools

Before you can create a VMware vCenter/vSphere Server template in InstallAnywhere, you need to obtain VMware Tools in *.tar.gz format. VMware Tools is a pack of utilities that enhance the functioning of the guest operating system on the virtual machine and that helps in managing it.

To obtain VMware Tools, you can either use a virtual machine that is running on a vSphere Server, or you can download VMware Tools.

Obtaining VMware Tools from a Virtual machine that Is Running on a vSphere Server



Task **To obtain VMware Tools from a virtual machine that is running on a vSphere Server:**

1. Launch an existing CentOS or Red Hat virtual machine (with GUI mode) running on a vSphere Server.

   Note ▪ *Ensure that the CD-ROM drive is enabled for the VM.*

2. Launch the vSphere Client.
3. Right-click the virtual machine, point to **Guest**, and then click **Install/Upgrade VMware Tools**. VMware Tools is available in the CD-ROM drive of the virtual machine as a `*.tar.gz` file.
4. Download the `*.tar.gz` file to your local machine.

Downloading VMware Tools



Task ***To download the VMware Tools:***

1. Follow the instructions in the following KB article to download the Linux i386 version of the VMware Tools for Console and GUI VMs:

   http://kb.vmware.com/selfservice/microsites/search.do?cmd=displayKC&externalId=1022525

2. Using the instructions in the KB article, GZIP the downloaded VMware Tools files so that they are in *.tar.gz format on an existing virtual machine with or without GUI.

   - **GUI**—On a GUI virtual machine, the *.tar.gz file is displayed as an icon on the desktop.
   - **Console**—On a console virtual machine, follow the steps described in the KB to copy the *.tar.gz file onto a virtual CD drive and then download it.

3. Download the `*.tar.gz` file to the InstallAnywhere build machine and use it as input for building the VM templates.

Creating a VM Template



Task ***To create a VMware vCenter/vSphere Server VM template:***

1. In the Advanced Designer, on the **Tools** menu, click **Wizards**, and then click **Create VM Template**. The **Create InstallAnywhere VM Template** wizard opens.
2. Complete each of the panels in the wizard as needed.

For information about each of the panels in the wizard, see Create InstallAnywhere VM Template Wizard.

Creating VM Templates Using the Command Line

The Create InstallAnywhere VM Template Wizard has command-line support, making it possible to include template creation in your automation scripts.

The command line takes in two `.properties` files as inputs:

```
createiso.properties
baselinevm.properties
```



Note ▪ *All the path entries in the two properties in Windows should use "\\" as a separator, such as:*

```
C:\\Program Files\\InstallAnywhere\\resource\\baseline_vms\\createiso.properties
```

To create a VM template through the command line, enter the CreateVMTemplate command using the following syntax:

```
CreateVMTemplate <mode_option> <hypervisor_option> <other_options>
```

The following categories of command-line options are available for the CreateVMTemplate command:

- Mode Options

- Hypervisor Options
- Other Options

Mode Options

The `<mode_options>` for the `CreateVMTemplate` command are as follows:

Table 8-6 ▪ Mode Options for the CreateVMTemplate Command

Option	Description
`-gui`	Opens up the GUI mode.
`-console`	Opens up the Console mode.

If you leave out the `<mode_option>` or specify the `<mode_option>` as `-gui`, the wizard opens in GUI mode.

Hypervisor Options

The `<hypervisor_options>` for the CreateVMTemplate command are as follows:

Table 8-7 ▪ Hypervisor Options for the CreateVMTemplate Command

Option	Description
`-amazon`	To build baseline VM for Amazon.
`-vmware`	To build baseline VM for VMware.

Other Options

The `<other_options>` for the CreateVMTemplate command are as follows:

Table 8-8 ▪ Other Options for the CreateVMTemplate Command

Option	Description
`-?`	Show this help text.
`-h`	Show this help text.
`-help`	Show this help text.
`-locale <locale>`	Set the locale to the provided language.
`-modifyISO propertiesFilePath`	Use specified ISO creation properties file.  ***Note*** ▪ *This parameter is applicable to VMware vSphere.*

Table 8-8 ▪ Other Options for the CreateVMTemplate Command (cont.)

Option	Description
`-createVM propertiesFilePath`	Use specified baseline VM properties file.  ***Note ▪*** *This parameter is applicable to VMware vSphere.*
`-useCloudImage propertiesFilePath`	Use specified baseline VM properties file.  ***Note ▪*** *This parameter is applicable to VMware vSphere.*
`-acceptAllEULA`	When creating a VM template using the command-line utility, a console panel opens and you are prompted to accept a Ubuntu or CentOS end user license agreement. When using automated builds and to accept all EULA agreements in console mode only, you can use the `-acceptAllEULA` argument.

Examples of the CreateVMTemplate Command

The following are examples of using the CreateVMTemplate command to create a VM template:

Table 8-9 ▪ Examples of the CreateVMTemplate Command

Example	Code
Create a baseline VM/ ISO modification for VMware	`% CreateVMTemplate -console -vmware -modifyISO <path_of_ISO_creation_properties_file> -createVM <path_of_baseline_vm_properties_file>`
Modify ISO	`% CreateVMTemplate -console -vmware -modifyISO <path_of_ISO_creation_properties_file>`  ***Note ▪*** *The* `modifyISO` *option is used to make an ISO (which is not capable of silent installation) capable of silent installation.*
Create a baseline VM with existing ISO (quiet install enabled)	`% CreateVMTemplate -console -vmware -createVM <path_of_baseline_vm_properties_file>`
Create baseline VM for Amazon	`% CreateVMTemplate -console -amazon -createVM <path_of_baseline_vm_properties_file>`
Create baseline VM using cloud image	`% CreateVMTemplate -console -amazon -useCloudImage <path_of_baseline_vm_properties_file>`

Specifying the Protocol in the baselinevm.properties File

A VMware vSphere/vCenter server may be configured to use either `http` or `https` protocol. (Refer to the VMware vSphere/vCenter documentation for more information.) The protocol configured for VMware vSphere/vCenter is set in the `protocol` section of the `baselinevm.properties` file:

```
#Protocol to access the host. Set either as "https" or "http". Mandatory
protocol=
```

By default, InstallAnywhere sets this to `https`. This change can only be made for console-based builds for the VM Template. The GUI will always try to use `https` as the protocol.

Sample Files

Sample files for `<path_of_baseline_vm_properties_files>` can be found in:

```
$IA_HOME$/resource/build/baselinevm.properties
```

Sample files for `<path_of_ISO_creation_properties_file>` can be found in:

```
$IA_HOME$/resource/build/createiso.properties
```

Creating a VMware vSphere 5 Virtual Appliance

InstallAnywhere enables you to create and deploy a VMware vSphere 5 virtual appliance that you can deploy to enterprise customers for use in a private cloud.

To create a VMware vSphere 5 virtual appliance, start with your InstallAnywhere installer project.



Task ***To create a VMware vSphere 5 virtual appliance:***

1. Open an existing InstallAnywhere project in the Advanced Designer.
2. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
3. Open the **Appliance Configuration** tab.
4. In the **Target Hypervisor** list, select **VMware vSphere 5**. The options that are displayed on the **Appliance Configuration** tab are now customized to VMware vSphere 5.
5. Specify the appliance configuration that you are editing by performing one of the following:
   - To edit an existing appliance configuration, such as the **Default_Appliance_Configuration**, select it from the **Select Appliance Configuration** list.
   - To create a new appliance configuration, click **Add** and perform the steps in Creating a New Appliance Configuration.
   - To copy an existing appliance configuration to edit, click **Copy** and perform the steps in Copying an Appliance Configuration.

When you add a new appliance configuration, an alphanumeric string is automatically entered in the **Appliance UUID** field to uniquely identify this virtual appliance. The virtual appliances and the VM within the virtual appliances are identified using the UUID and the version. It is important to define them uniquely so that this information can be used for maintaining the virtual appliance properly. To generate a new ID, click the **Generate UUID** button.

6. Next to the **EULA** field, click **Choose EULA File** and select a EULA text (`.txt`) file that contains the end user license agreement information for the virtual appliance. The EULA must be a text-only file and will not support markup of any kind. Your customers will be prompted to accept this EULA before they will be permitted to use the virtual appliance.



***Note** ▪ Specifying a EULA is mandatory for the creation of a virtual appliance.*

7. In the **Description** field, enter text to describe the purpose of this virtual appliance.
8. In the **Appliance URL** field, enter the start URL by which the appliance can be accessed post deployment. Enter the **Appliance URL** using the following format:

   `<protocol>#hostname#<context_root>`

   where:

   - `<protocol>` is of type `http://`
   - `#hostname#` will be replaced by the IP of the appliance obtained after deployment of the virtual appliance
   - `<context_root>` is the context root of the virtual appliance

   For more information, see Specifying the Appliance URL.
9. In the **Connection Settings** area of this tab, click the **Add New Credential** button. The **Add New Connection / VMware vSphere 5** dialog box opens.
10. Configure the connection settings as needed, and then click the **OK** button.



***Note** ▪ InstallAnywhere creates a credential store file (`*.credstore`) in the project folder for the project. Using a credential store helps you securely store all of connection settings, and also enables you to define some connection settings globally and reuse them in all of your projects.*

11. Under **OVF Appliance Type**, select one of the following options:

    - **VMware vCenter compatible Appliance**—This type of appliance is deployed in OVF 1.1 format on a licensed VMware vSphere server that is managed by a licensed VMware vCenter. This appliance is deployed as a pure virtual appliance and has its own resource pool. Such virtual appliances are also capable of having multiple virtual machines under the appliance.
    - **VMware vSphere compatible Appliance**—This type of appliance is also deployed in the OVF 1.1 format on a licensed/free version of VMware vSphere, which may or may not be managed by a VMware vCenter. This appliance can be deployed only as a single virtual machine.
12. Under **Appliance Output Options**, select one of the following options:

    - **OVF (Open Virtualization Format) 1.1**—Contains the VM disk files and the OVF descriptor in a single directory.

- **OVA (OVF Archive)**—The OVF 1.1 format output is archived as a `tar` file, which enables easier distribution of the virtual appliances.
- **Both**—The virtual appliance output is provided in both OVF 1.1 and OVA formats. This option will consume a larger amount of disk space.



Note *▪ For more information, see Configuring the Appropriate Appliance Output and Type for VMware-Based Virtual Appliances.*

13. To enable automatic deployment of this virtual appliance, select the **Deploy this Appliance** check box.
14. To deploy the virtual appliance to the same machine on which you are building the virtual appliance, select the **Same as the build credentials** check box.

    To deploy the virtual appliance to a different server—not the one on which you are building the virtual appliance—clear the **Same as the build credentials** check box, and then select the appropriate machine in the box under this check box. The box contains all of the machines that are defined in the **Credentials** list.
15. Click the **VM Configuration** tab.
16. In the **Select InstallAnywhere VM Tier** list, select the VM tier that you want to configure, or click the **Add VM Tier** button next to this setting to add a new tier.
17. Click the **Choose a VM** button, which is next to the **Chosen VM** list. The Choose a VM Wizard opens.
18. Complete the panels in the wizard to specify which virtual machine or template you want to use for your virtual appliance.



Note *▪ For information on obtaining and configuring VM templates, see Obtaining VM Templates for Virtual Appliances.*

19. Open the **Installers** subtab of the **VM Configuration** tab.
20. To add your installer to the **Build Time** tab list, click the **Add Installer** button. The **Add Installer** dialog box opens.



Note *▪ For information on the purpose of **Build Time** vs. **First Boot Time** installers, see Using Custom Boot and Login Scripts for a Virtual Appliance.*

21. On the **Add Installer** dialog box, perform the following steps:
    a. Select the **An Installer out of the current InstallAnywhere project** option.
    b. In the **Choose Installer Build Configuration** list, select the Build Configuration of the installer that you are adding.



Note *▪ This option is displayed only if the **An installer out of the current InstallAnywhere project** option is selected.*

    c. In the **Choose Build Target** list, select the build target of the installer that you are adding.



***Note** ▪ This option is displayed only if the **An installer out of the current InstallAnywhere project** option is selected.*



***Note** ▪ If you do not have any Linux-based build targets enabled in your project, this field will be disabled.*

d. Click **OK**. The **Edit Installer Properties Table** dialog box opens, displaying properties (or response options) that need to be supplied to the installer in order for it to silently install on the virtual appliance.

e. In the **Value** field for each product property, specify a default value. When you enter values on this dialog box, you are creating the response file. When InstallAnywhere installs this installer in the virtual machine, it will pass on these customized values.



***Important** ▪ It is important to enter the default values in the **Value** column.*



***Note** ▪ If you want to add properties that were not automatically discovered by InstallAnywhere, click **Add** to add a new property. This is a required step when using third party installers because InstallAnywhere is unable to automatically discover the properties in a third party installer.*



***Tip** ▪ To edit values previously entered on the **Edit Installers Properties Table** dialog box, select the installer name on the **Properties** subtab of the **VM Configuration** tab and click **View/Edit**.*

f. Click **OK**. The specified installer is now listed on the **Installers** subtab of the **VM Configuration** tab.



***Note** ▪ For more information on adding installers, see Adding Installers to Your Virtual Appliance.*

22. Click the **Build Appliance** button.

Building begins and the Building dialog box shows the progress. The appliance should be ready in a few minutes. The speed depends on your network bandwidth and the configuration of your build system.



***Important** ▪ If you cancel an appliance build by clicking the Cancel button on the Building dialog box, InstallAnywhere will wait for the current step to complete, and then clean up the files and exit the build process. However, if you cancel when building via command line using Ctrl+C, no file clean up will take place.*

The output of the Build Appliance process is one of the following, depending upon the selection you made under **Appliance output options** on the Appliance Configuration tab:

- **OVF (Open Virtualization Format) 1.1**—A set of disks in VMDK format with an .ovf file.
- **OVA (OVF Archive)**—An .ova archive file
- **Both**—A set of disks in VMDK format with an .ovf file, and an .ova archive file.

This output will be found in the Appliance Configuration directory inside the build output folder for your project.

If you selected the Deploy this Appliance check box on the Appliance Configuration tab, your appliance will be automatically deployed in the vCenter/vSphere server.

To manually deploy a VMware vSphere 5 virtual appliance, see Deploying a VMware vSphere 5 Virtual Appliance on Demand.



Tip ▪ *For a sample project, see Using the IABookstore Sample for Creating Virtual Appliances.*

Creating an Amazon EC2 Virtual Appliance

InstallAnywhere enables you to create and deploy an Amazon EC2 virtual appliance that you can deploy to a public cloud.

To create an Amazon EC2 virtual appliance, start with your InstallAnywhere installer project.



Task ***To create an Amazon EC2 virtual appliance:***

1. Open an existing InstallAnywhere project in the Advanced Designer.
2. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
3. Open the **Appliance Configuration** tab.
4. In the **Target Hypervisor** list, select **Amazon EC2**. The options that are displayed on the **Appliance Configuration** tab are now customized to Amazon EC2.
5. Specify the appliance configuration that you are editing by performing one of the following:
   - To edit an existing appliance configuration, such as the **Default_Appliance_Configuration**, select it from the **Select Appliance Configuration** list.
   - To create a new appliance configuration, click **Add** and perform the steps in Creating a New Appliance Configuration.
   - To copy an existing appliance configuration to edit, click **Copy** and perform the steps in Copying an Appliance Configuration.
6. When you add a new appliance configuration, an alphanumeric string is automatically entered in the **Appliance UUID** field to uniquely identify this virtual appliance. The virtual appliances and the VM within the virtual appliances are identified using the UUID and the version. It is important to define them uniquely so that this information can be used for maintaining the virtual appliance properly. Click **Generate UUID** to generate a new ID.
7. Next to the **EULA** field, click **Choose EULA File** and select a EULA text (`.txt`) file that contains the end user license agreement information for the virtual appliance. The EULA must be a text-only file and will not support markup of any kind. Your customers will be prompted to accept this EULA before they will be permitted to use the virtual appliance.



Note ▪ *Specifying a EULA is mandatory for the creation of a virtual appliance.*

8. In the **Description** field, enter text to describe the purpose of this virtual appliance.
9. In the **Appliance URL** field, enter the start URL by which the appliance can be accessed post deployment. Enter the **Appliance URL** using the following format:

   `<protocol>#hostname#<context_root>`

   where:

   - `<protocol>` is of type `http://`
   - `#hostname#` will be replaced by the IP of the appliance obtained after deployment of the virtual appliance
   - `<context_root>` is the context root of the virtual appliance

   For more information, see Specifying the Appliance URL.
10. In the **Connection Settings** area of this tab, click the **Add New Credential** button. The **Add New Connection / Amazon EC2** dialog box opens.
11. Configure the connection settings as needed, and then click the **OK** button.



Note ▪ *InstallAnywhere creates a credential store file (`*.credstore`) in the project folder for the project. Using a credential store helps you securely store all of connection settings, and also enables you to define some connection settings globally and reuse them in all of your projects.*

For information on Amazon EC2 cloud computing and obtaining an Amazon Web Services account (including Account Number, Serial Key, Access Key, Private Key, and x509 Certificate), see:

http://aws.amazon.com/ec2/

http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/using-credentials.html#using-credentials-certificate

http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/using-credentials.html

12. To enable automatic deployment of this virtual appliance, select the **Deploy this Appliance** check box.
13. Under **Root Device Type**, select one of the following options to specify the type of disk storage that your AMI image will be stored in:

    - **EC2 instance store backed**—Temporary block level storage (also known as an “instance store” volume). An instance store is dedicated to a particular instance and the data on it persists only during the lifetime of its associated EC2 instance. Data on such instance store volumes are lost when the instances are stopped, fail, or are terminated, and cannot be restored. It is ideal to use such instance-store-backed instances for temporary storage of information or content.
    - **Elastic block store backed**—Reliable storage volumes that can be attached to a running instance in the same availability zone. EBS volumes attached to an instance persist independently from the life of the instance and are recommended to be used when data changes frequently and requires long term persistence.



Note ▪ *For more information, see Support for Elastic Block Store (EBS) for Amazon.*

14. If you selected **Elastic block store backed** option, specify the following additional options:
    a. In the **EBS Volume size (GiB)** field, specify the size in gigabytes of your EBS volume. The limit is up to 1 TiB depending on your AWS account limits.
    b. In the **Device** field, enter the name of the root device that the volumes will be attached to. For example, the recommended connection is:

       `/dev/sdf` through `/dev/sdp`

    c. In the **Image Name** field enter the name of the EBS backed image created from the EBS snapshot.



Note ▪ *For more information, see Support for Elastic Block Store (EBS) for Amazon.*

15. In the **Select the region for deployment** list, select one of the following regions to upload and run your virtual appliance:
    - **US East (Northern Virginia) Region**
    - **US West (Northern California) Region**
    - **US West (Oregon) Region**
    - **South America (Sao Paulo) Region**
    - **EU (Ireland) Region**
    - **Asia Pacific (Tokyo) Region**
    - **Asia Pacific (Singapore) Region**



Note ▪ *It is always beneficial to deploy the AMI to the region closest to your customer base. For example, if your business runs in South America only, then it is better to base your deployed AMI in the Sao Paulo region. This will result in faster response times. Also, both the S3 bucket and AMI instance would be deployed to the corresponding region, which would help customers reduce costs.*

16. Click the **VM Configuration** tab.
17. To identify the operating system of the VM that you are including in this virtual appliance, in the **Select OS** list, select the appropriate option.
18. In the **Select InstallAnywhere VM Template** list, select a VM template to use to build this virtual appliance.



Note ▪ *For information on obtaining and configuring VM templates, see Obtaining VM Templates for Virtual Appliances.*

19. Open the **Installers** subtab of the **VM Configuration** tab.
20. To add your installer to the **Build Time** tab list, click the **Add Installer** button. The **Add Installer** dialog box opens.



Note ▪ *For information on the purpose of **Build Time** vs. **First Boot Time** installers, see Using Custom Boot and Login Scripts for a Virtual Appliance.*

21. On the **Add Installer** dialog box, perform the following steps:
    a. Select the **An Installer out of the current InstallAnywhere project** option.
    b. In the **Choose Installer Build Configuration** list, select the Build Configuration of the installer that you are adding.
    c. In the **Choose Build Target** list, select the build target of the installer that you are adding.
    d. Click **OK**. The **Edit Installer Properties Table** dialog box opens, displaying properties (or response options) that need to be supplied to the installer in order for it to silently install on the virtual appliance.
    e. In the **Value** field for each product property, specify a default value. When you enter values on this dialog box, you are creating the response file. When InstallAnywhere installs this installer in the virtual machine, it will pass on these customized values.



Important ▪ *It is important to enter the default values in the **Value** column.*



Note ▪ *If you want to add properties that were not automatically discovered by InstallAnywhere, click **Add** to add a new property. This is a required step when using third party installers because InstallAnywhere is unable to automatically discover the properties in a third party installer.*



Tip ▪ *To edit values previously entered on the **Edit Installers Properties Table** dialog box, select the installer name on the **Properties** subtab of the **VM Configuration** tab and click **View/Edit**.*

    f. Click **OK**. The specified installer is now listed on the **Installers** subtab of the **VM Configuration** tab.



Note ▪ *For more information on adding installers, see Adding Installers to Your Virtual Appliance.*

22. Click the **Build Appliance** button.

Building begins and the Building dialog box shows the progress. The appliance should be ready in a few minutes. The speed depends on your network bandwidth and the configuration of your build system. An AMI image will be produced.



Important ▪ *If you cancel an appliance build by clicking the Cancel button on the Building dialog box, InstallAnywhere will wait for the current step to complete, and then clean up the files and exit the build process. However, if you cancel when building via command line using Ctrl+C, no file clean up will take place.*

If you selected the Deploy this Appliance check box on the Appliance Configuration tab, your image will be automatically deployed to the Amazon EC2 infrastructure.

For information on manually deploying an Amazon EC2 virtual appliance, see Deploying an Amazon EC2 Virtual Appliance on Demand.



Tip ▪ *For a sample project, see Using the IABookstore Sample for Creating Virtual Appliances.*

Working with Virtual Appliance Configurations

In some scenarios, you may need to generate multiple virtual appliances for the same application, each with a slightly different configuration. For example, for a particular application, you may want to build a virtual appliance for both a VMware vSphere 5 and an Amazon EC2 hypervisor or using a different VM template. Here are some other examples of appliance configuration scenarios:

- VMware vSphere appliance using a Ubuntu VM template with low hardware requirements
- VMware vSphere appliance using a CentOS VM template with high hardware requirements
- VMware vSphere appliance using a CentOS VM template with low hardware requirements
- Amazon EC2 appliance

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 define different appliance configurations in the same project in InstallAnywhere. Doing so enables you to build various types of appliances (such as Amazon EC2 and VMware vSphere) from the same InstallAnywhere project, and to be able to customize your offering of appliances according to your customer segments.

Each InstallAnywhere project can have multiple virtual appliance configurations, each representing how the virtual appliance will be built for a particular target hypervisor, VM template, set of hardware requirements, or other settings. To create and modify appliance configurations, use Build Appliances view on the Build page of the Advanced Designer. The Appliance Configuration tab in this view is where you can add, rename, copy, or delete an appliance configuration. This section of the documentation explains how.

Creating a New Appliance Configuration

You can create multiple appliance configurations in the same InstallAnywhere project, each with different settings. Each appliance configuration defines how InstallAnywhere builds a virtual appliance for a particular target hypervisor, VM template, set of hardware requirements, and other settings. You can store as many build configurations with a project as necessary.



Task ***To add a new appliance configuration to a project:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. Click the **Add Appliance Configuration** button. The **Add Configuration** dialog box opens, prompting you to enter a name for the new appliance configuration.
4. Enter a name and click **OK**.

The name must be 60 characters or fewer, and it must not contain any of the following characters: asterisk (*), question mark (?), period (.), backslash (\), or forward slash (/).

InstallAnywhere adds the new appliance configuration and selects it in the Select Build Configuration list. Configure the build configuration's settings as needed. To learn more, see:

- Creating a VMware vSphere 5 Virtual Appliance
- Creating an Amazon EC2 Virtual Appliance
- Working with Virtual Appliance Configurations

Renaming an Appliance Configuration

InstallAnywhere lets you change the name of appliance configurations in your projects.



Task ***To rename an appliance configuration:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to rename.
4. Click the **Rename Appliance Configuration** button. The **Rename Appliance Configuration** dialog box opens.
5. Enter a new name for this appliance configuration and click **OK**.

   The name must be 60 characters or fewer, and it must not contain any of the following characters: asterisk (*), question mark (?), period (.), backslash (\), or forward slash (/).

InstallAnywhere updates the name of the appliance configuration.

Copying an Appliance Configuration

InstallAnywhere provides the ability to copy an appliance configuration in a project. Copying creates a new appliance configuration with the same settings as the original appliance configuration. You can modify the copied appliance configuration as needed without modifying the original appliance configuration.

You may want to create one appliance configuration with the appropriate default settings, and to use it as a template to create additional appliance configurations that have slight changes from the default values.



Task ***To copy an appliance configuration:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Build Appliance** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to copy.
4. Click the **Copy Appliance Configuration** button. The **Copy Configuration** dialog box opens.
5. Enter a name for the new appliance configuration and click **OK**.

The name must be 60 characters or fewer, and it must not contain any of the following characters: asterisk (*), question mark (?), period (.), backslash (\), or forward slash (/).

InstallAnywhere adds the new appliance configuration and selects it in the Select Appliance Configuration list. Configure the appliance configuration's settings as needed. To learn more, see:

- Creating a VMware vSphere 5 Virtual Appliance
- Creating an Amazon EC2 Virtual Appliance
- Working with Virtual Appliance Configurations

Removing an Appliance Configuration

If you are no longer using a particular appliance configuration in your project, you may want to delete it from your project.



Task ***To delete an appliance configuration:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to delete.
4. Click the **Remove Appliance Configuration** button. InstallAnywhere displays a message box, prompting you to confirm that you want to delete the selected appliance configuration.
5. Click **Yes**.

InstallAnywhere removes the appliance configuration from your project and from the Select Appliance Configuration list.



Note ▪ *A project must have at least one appliance configuration. If a project contains only one appliance configuration, the Remove Appliance Configuration button on the Appliance Configuration tab is disabled.*

Adding an Appliance Configuration to the Project Build

You can add an appliance configuration to the project build by selecting its **Add to project build** option. If an appliance configuration is added to a project build, that appliance configuration is built each time that one of the following occurs:

- You click the Build Appliance or Build Selected buttons on the Appliance Configuration tab in the Build Appliances view (on the Build page).
- You build the project from the command line without specifying any appliance configurations, such as:

  ```
  build.exe MyProduct.iap_xml -applianceMode
  ```



Task ***To add an appliance configuration to the project build:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to add to the build.
4. Select the **Add to project build** check box.



Note ▪ *If you click the Build All button in the Build Appliances view (or use the* `-all` *command-line argument), all existing build configurations for that project are built, regardless of whether their* ***Add to project build*** *check box has been selected.*

Specifying Connection Settings for Building Virtual Appliances

When you are defining a virtual appliance configuration in InstallAnywhere, you need to specify information such as the user name and password that should be used to connect to the machine that will be used to create the VM for your virtual appliance. You also need to specify other information that varies, depending on whether the target hypervisor is VMware vSphere or Amazon EC2.

InstallAnywhere considers this type of connection information to be credentials. InstallAnywhere does not store the credentials in the project file, but instead stores these credentials using a credential store. The credential store is encrypted and available for use in all of the InstallAnywhere projects that are accessed on the machine. That is, once you define a set of credentials in one project on a machine, those credentials are available for selection in all of the other projects that you open on the same machine.

In order for InstallAnywhere to properly retrieve the credentials at build time, the credential store must be stored in your source code control system.



Task ***To define a set of credentials to an appliance configuration:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to customize.
4. Click the **Add New Credential** button. The **Add New Connection** dialog box opens.
5. Configure the connection settings as needed.
6. Click **OK**.

InstallAnywhere adds the connection settings that you configured to the credential store, and selects this set in the Connection Settings list on the Appliance Configuration tab.



Task ***To edit an existing set of credentials:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to customize.
4. In the **Connection Settings** list, select the name of the credential set that you want to customize.
5. Click the **Manage Credential** button. The **Manage Connection** dialog box opens.
6. Configure the connection settings as needed.
7. Click **OK**.

InstallAnywhere updates the credential store.

Configuring the Appropriate Appliance Output and Type for VMware-Based Virtual Appliances

For VMware-based virtual appliances, there are three output options:

- **OVF 1.1 format**—Contains the VM disk files and the OVF descriptor in a single directory.
- **OVA format**—The OVF 1.1 format output is archived as a `tar` file, which enables easier distribution of the virtual appliance.
- **Both**—The virtual appliance output is provided in both OVF 1.1 and OVA formats. This option will consume a larger amount of disk space.



Note ▪ *InstallAnywhere does not have support for using manifest files and for signing OVA packages.*

For VMware-based virtual appliances, there are two appliance types:

- **VMware vCenter and vSphere-compatible appliances**—Appliance is deployed in OVF 1.1 format on a licensed VMware vSphere server which is managed by a licensed VMware vCenter. This appliance will get deployed as a pure virtual appliance and will have its own resource pool. Such virtual appliances are also capable of having multiple virtual machines under the appliance.
- **VMware vSphere-compatible appliances**—Appliance is also deployed in OVF 1.1 format on a licensed or free version of VMware vSphere, which may or may not be managed by a VMware vCenter. This appliance will be deployed as a single virtual machine.



Note ▪ *In some cases, you may want to host the disk files in OVF format on the Internet and just supply the OVF descriptor file to your customers. In this scenario, you will need to manually update the OVF with the hosted location of the disk files. The VMware client that deploys this OVF will automatically download the required disks from the Internet and proceed to deploy the appliance.*

Specifying the Appliance URL

In the Appliance URL field on the Appliance Configuration tab of the Build Appliances view, you need to enter the start URL by which the appliance can be accessed post deployment. The Appliance URL uses the following format:

```
<protocol>#hostname#<context_root>
```

where:

- `<protocol>` is of type `http://`
- `#hostname#` will be replaced by the IP of the appliance obtained after deployment of the virtual appliance.
- `<context_root>` is the context root of the virtual appliance.

A typical virtual appliance URL can be specified by the appliance author in the InstallAnywhere GUI as:

```
http://#hostname#/sample/demo.html
```

If the IP of the deployed appliance is `x.y.a.b`, then this URL will be resolved as:

```
http://x.y.a.b/sample/demo.html
```

and will be presented as the start URL of the appliance to the end user.



***Note** ▪ The finally resolved appliance URL will also be available to the user in the `ApplianceBuildResults.txt` file, if automatic deployment is selected.*

Support for Elastic Block Store (EBS) for Amazon

When you launch an Amazon EC2 instance, the root device volume contains the image used to boot the instance. Amazon EC2 provides you with several data storage options for your instances. There are mainly two types of root device storage backed instances: instance store backed instances and EBS backed instances.

Amazon EC2 Instance Store Backed Instances

Instance store backed instances are temporary block level storage volumes (also known as instance stores) that are used by Amazon EC2 instances. If the root device is an instance store volume that was created from a template stored in Amazon S3, any Amazon EC2 instances that are launched from this AMI are known as instance store backed instances.

An instance store is dedicated to a particular instance and the data on it persists only during the lifetime of its associated EC2 instance. Data on such instance store volumes is lost when the instances are stopped, fail, or are terminated and cannot be restored. Therefore, such instance store backed instances should be used for temporary storage of information or content.

Amazon EBS Backed Instances

Amazon Elastic Block Storage (EBS) are reliable block level storage volumes for use with Amazon EC2 instances. Amazon EBS volumes are highly available and reliable storage volumes that can be attached to any running instance in the same availability zone. If the root device is an Amazon EBS volume created from an Amazon EBS snapshot, then instances launched from such AMIs are known to as EBS backed instances.

EBS volumes attached to an instance persist independently from the life of the instance and are recommended when data changes frequently and requires long term persistence. Amazon EBS volumes provides certain key features like data availability across availability zones, data persistence independent of the instance life, snapshots and incremental backups.

EBS volumes remain attached throughout the stop-start life of an instance, and are automatically detached when the instance terminates, with the data left intact. The data persists on the volume until it is explicitly deleted.

Elastic Block Store Backed Settings on the Appliance Configuration Tab

You can specify that your Amazon virtual appliance will be an EBS backed instance by making selections on the Appliance Configuration tab of the Build Appliances view on the Build page.

When the Deploy this Appliance check box is selected and the Root Device Type is set to **Elastic block store backed**, the following EBS backed settings must also be specified:

- **EBS Volume size (GiB)**—Specify the size in gigabytes of your EBS volume. The limit is up to 1 TiB depending on your AWS account limits.
- **Device**—Enter the name of the root device that the volumes will be attached to.
- **Image Name**—Enter the name of the EBS backed image created from the EBS snapshot.



Note *▪ For more information on Amazon Elastic Block Store and the Amazon EC2 Instance Store, see the following:*

- *http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/AmazonEBS.html*
- *http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/InstanceStorage.html*

About EC2 and S3 Regions of Deployment for Amazon

The creation of an S3 bucket (which you upload files to) on the AWS Management Console is region-specific. Amazon has given you the flexibility to create data buckets at seven different regions across the world:

- US East (Northern Virginia) Region
- US West (Northern California) Region
- US West (Oregon) Region
- South America (Sao Paulo) Region
- EU (Ireland) Region
- Asia Pacific (Tokyo) Region
- Asia Pacific (Singapore) Region



Note *▪ It is always beneficial to deploy the AMI to the region that is closest to your customer base. For example, if your business runs in South America only, it is best to base your deployed AMI in the Sao Paulo region. This results in faster response times. Also, both the S3 bucket and AMI instance would be deployed to the corresponding region, which would help customers reduce costs.*

You can create an S3 bucket in any of these regions. Amazon guarantees that the S3 bucket will be created in the respective region and that the user-uploaded files will be stored in the specified region.

The same region specification applies to EC2 as well. The launching and running of EC2 instances can be done in any of these regions.



Tip ▪ *To avoid cost overhead of data transmission from S3 bucket to instance launch region and also to make instance launching more efficient by avoiding data transmission time-overhead, it is advised that the S3 bucket creation and running of EC2 instances should be done in a same region.*

Specifying the Amazon Instance Region

You can specify your S3 and EC2 region settings by making a selection in the **Select the region for deployment** list on the Appliance Configuration tab of the Build Appliances view on the Build page.

Working with VM Configurations for a Virtual Appliance

A virtual appliance that is configured and built in InstallAnywhere can be deployed to a single server or a group of servers. One example of a multi-tier architecture that can be set up as a virtual appliance is a shopping Web site. The site may consist of various tiers: the Web site tier, the database tier, and one or more application tiers. Each tier depends on the other tiers in various ways. For example, the application tier needs to be able to write to and read from the databases in the database tier. The Web tier needs to be able to contact the application tier. InstallAnywhere lets you configure the settings for each of these tiers; when the virtual appliance is deployed to the cloud, the settings are configured in each of the tiers as needed.

To define one or more tiers and to configure their settings, use the VM Configuration tab in the Build Appliances view on the Build page. This tab is also where you configure information such as the operating system that you want to be present on your VMs, the installers that need to be run on those VMs, and more.

This section of the documentation describes how to configure tiers and select VMs for your virtual appliance.

Selecting a VM for a VMware Virtual Appliance

If you are creating a virtual appliance for VMware vSphere/vCenter, you can use an existing VM, or a VM snapshot that is running on an existing server, for creating a virtual appliance.



Task ***To select a VM for your VMware virtual appliance:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to configure.
4. Click the **VM Configuration** tab.
5. In the **Select VM Tier** list, select the VM tier that you want to configure.

6. Click the **Choose a VM** button, which is next to the **Chosen VM** list. The **Choose a VM Wizard** opens.
7. Complete the panels in the wizard as needed.

To learn more, see Choose a VM Wizard.

Specifying Hardware Requirements for a Virtual Appliance

Instead of offering different product editions (such as Basic, Deluxe, and) of your virtual appliance with different sets of functionality for each edition, you may want to offer virtual appliances with varying sets of hardware requirements or compute capacity.

To specify the hardware requirements for a target hypervisor, use the Hardware subtab of the VM Configuration tab in the Build Appliances view on the Build page. The options that are listed on this subtab depend on which target hypervisor is selected on the Appliance Configuration tab in the Build Appliances view.

For a virtual appliance with a target hypervisor of VMware vSphere 5, you can specify basic hardware requirements such as number of virtual CPUs, number of virtual CDROMs, number of virtual Ethernet adapters, the default virtual hard disk size, and the default virtual RAM size. InstallAnywhere will create the various controllers—such as SCSI and IDE controllers—for your hard drives and CDROMs respectively. The network adapters are all set to default PCNET32 type.

When you are creating Amazon EC2 virtual appliances, you can use the Hardware subtab to help you choose the instance type to be associated with the Amazon instance, such as a small instance, high-CPU medium instance, or medium instance. The hardware requirements for each of these instance types is constantly being adjusted and updated by Amazon. For the most up-to-date information, see http://aws.amazon.com/ec2/instance-types/.



Task ***To specify hardware requirements:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration whose hardware requirements you want to specify.
4. Click the **VM Configuration** tab.
5. For VMware vSphere appliances: In the **Select VM Tier** list, select the VM tier whose hardware you want to configure.
6. Click the **Hardware** subtab.
7. Specify the hardware requirements.

For information on each of the settings that you can configure, see Hardware Subtab.

Installing Operating System Packages for a Virtual Appliance

The prebuilt VM templates that are available for InstallAnywhere include several operating system packages, such as `SSH`, `wget`, `perl`, and `openssl`. These packages are used by the virtual appliance creation process.

However, there might be a need to install additional operating system packages for your specific virtual appliance to work. You can simply choose the ones for your virtual appliance in the Build Appliances view on the Build page, and these will be installed for you during the creation of the virtual appliance.



Task ***To customize the OS packages that are installed with your virtual appliance:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to customize.
4. Click the **VM Configuration** tab.
5. For VMware vSphere appliances: In the **Select VM Tier** list, select the VM tier that you want to configure.
6. Click the **OS Packages** subtab. All of the OS packages that are available to install are listed in the **Packages Available to Install** list.
7. In the **Packages Available to Install** list, select the package that you want to add and click the **Add** button.

InstallAnywhere adds the OS package to the Packages Selected to Install list.

For more information, see OS Packages Subtab.



Note ▪ *For information on defining your own custom repositories for the installation of operating system packages, see Specifying Which OS Package Repository to Use for Virtual Appliances.*

About Using Ubuntu OS Packages

On Ubuntu, while OS packages are being installed, the Package Manager of Ubuntu tries to fix any pre-existing problems with the packages that are installed on the machine. Therefore, problems may occur when a package installation has failed before the appliance build runs.

For example, if prior to the appliance build and deployment, the end user installed a package named Package_A, and within the appliance the appliance author has bundled another package named Package_X, the installation of OS packages on Ubuntu causes the Ubuntu Package Manager to retry the installation of Package_A before it tries to install Package_X. If Package_A continues to have problems with installation, these errors cause the installation of Package_X to fail, causing the entire appliance build and deployment to fail.

In order to avoid such errors, you are advised to run the following command and ensure that all errors are resolved:

```
dpkg-reconfigure -a
```

Proprietary Operating Systems

If you create OS packages that are proprietary and not part of the standard Ubuntu or CentOS ecosystem, you can include them in your virtual appliance. For more information, see Using RPM/Debian Packages for Proprietary Operating Systems on Virtual Appliances.

Specifying Which OS Package Repository to Use for Virtual Appliances

The virtual appliance creation process requires that the build system be connected to the Internet. However, in some cases, because of security reasons or low Internet bandwidth, such a connection might not be possible. In these cases, you can choose to set up an OS (Ubuntu or Centos) repository of OS packages and OS-related components on the build machine, and have InstallAnywhere use this repository at build time. The VM Configuration tab in the Build Appliances view on the Build page is where you specify whether you want to use Internet-based repositories or local repositories. Note that the repositories need to be accessible to the virtual appliance build system in order to be able to resolve the IP addresses and install the OS packages.



Task ***To specify which repository you want to use:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to customize.
4. Click the **VM Configuration** tab.
5. For VMware vSphere appliances: In the **Select VM Tier** list, select the VM tier that you want to configure.
6. Click the **Repository Settings** subtab.
7. Select the repository type that you want to use.

   If you select the **Local Repositories** option, use the **Location of Local Repository Specification File** setting to specify the appropriate path. Note the following information:

   - As part of the specification, you can point to multiple repositories, and include repositories that list your proprietary packages in the specification file.
   - For baseline VM creation, the repository settings serve as an advanced option that can be pointed to by using the `Baseline.vm.local.repository.location` inside the appropriate property files.



Note ▪ *The only reason a specification file is recommended instead of a pointer to a single server is to give you the ability to point to multiple repository servers. In a distributed development system, each contributing team may publish their OS packages on a custom repository server, and the appliance author might need to consolidate all of these custom repository servers in a single file.*

About Using Ubuntu OS Packages in the Repository

On Ubuntu, while OS packages are being installed, the Package Manager of Ubuntu tries to fix any pre-existing problems with the packages that are installed on the machine. Therefore, problems may occur when a package installation has failed before the appliance build runs.

For example, if prior to the appliance build and deployment, the end user installed a package named Package_A, and within the appliance the appliance author has bundled another package named Package_X, the installation of OS packages on Ubuntu causes the Ubuntu Package Manager to retry the installation of Package_A before it tries to install Package_X. If Package_A continues to have problems with installation, these errors cause the installation of Package_X to fail, causing the entire appliance build and deployment to fail.

In order to avoid such errors, you are advised to run the following command and ensure that all errors are resolved:

```
dpkg-reconfigure -a
```

Adding Installers to Your Virtual Appliance

When you are configuring a VM for a virtual appliance, you need to specify the installers that need to be run on the VM. You can include the installer that is created as part of the same InstallAnywhere project that you are using to create the virtual appliance. You can also optionally include existing built executable files from other InstallAnywhere projects. In addition, you can include third-party installers. When you build the virtual appliance in InstallAnywhere, InstallAnywhere automatically runs all of the installers that are configured to be part of the VM.



Note ▪ *All installers that you specify must be capable of unattended silent installations; otherwise, the appliance build might stop at the point at which these installers prompt for user interaction.*



Task ***To add installers to a VM configuration:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to customize.
4. Click the **VM Configuration** tab.
5. For VMware vSphere appliances: In the **Select VM Tier** list, select the VM tier that you want to configure.
6. Click the **Installers** subtab.
7. Click the appropriate tab to indicate when you want the installer to be executed. Available options are:
   - **Build Time**—Build-time installers are mostly used to push payload for the software appliance.
   - **First Boot Time**— If some configuration needs to be performed before installation, use the First Boot Time installer tab. First-boot-time installers are executed the first time that the virtual appliance is booted.
8. Click the **Add Installer** button. The **Add Installer** dialog box opens.

9. Select the type of installer that you want to add to the VM configuration. Available options are:

   - **A pre-built InstallAnywhere installer**—If you select this option, specify the location of the installer.
   - **An installer out of the current InstallAnywhere project**—If you select this option, specify the build configuration and the build target that you want to use.
   - **Other installers**—If you select this option, specify the executable file, as well as the command line that should be passed to the executable file. The format for entering the command line is:

     `@@executable@@ /i /s /r @@responsefile@@`

     `@@executable@@` refers to the installation executable file, and `@@responsefile@@` refers to the response file. Following are examples:

     - **Windows**—If on a Windows machine, the name of executable is `setup.exe` and the response file name is `resp.rsp`, then the above installation command is executed as:

       ```
       $> setup.exe /i /s /r resp.rsp
       ```

     - **Linux**—If on a Linux box, the name of the executable is `abc.rpm` and the installation command is specified as:

       ```
       rpm -ivh @@executable@@
       ```

       Then the installation command is executed as:

       ```
       $> rpm -ivh abc.rpm
       ```

10. If appropriate, specify the location of the response file that you want to use. The response file should provide default values for the variables that end users are required to provide during installation. InstallAnywhere uses the response file when building the appliance.

    If you do not specify the location of the response file, InstallAnywhere checks the installer and then prompts you to enter default values for the end user–defined variables. Note that the larger the installers is, the longer this step may take. Therefore, for a larger installer, you may opt to include a response file.

11. Click **OK**.

InstallAnywhere adds the installer to the list on the Build Time tab or the First Boot Time tab.

Using RPM/Debian Packages for Proprietary Operating Systems on Virtual Appliances

If you create OS packages that are proprietary and not part of the standard Ubuntu or CentOS ecosystem, you can include them in your virtual appliance. To do so, bundle your OS package as an RPM or Debian package. InstallAnywhere offers two techniques for including standalone RPM/Debian packages as part of your appliance build: as OS packages or as installers.

Including a Standalone RPM/Debian Package as an OS Packages



Task ***To include a standalone RPM/Debian package as an OS package:***

1. Prepare an XML file:
   a. Open the following schema file:

      `$IA_HOME$/resource/ospackages/ospackages.xsd`

   b. Write a separate XML file that conforms to this .xsd for your RPM package. For example, for an RPM file named `abc.rpm`, the XML file might look like this:

      ```
      <?xml version="1.0" encoding="UTF-8"?>
      <ospackages>
          <package>
              <displayName>My ABC RPM package</displayName>
              <version>1.0</version>
              <internalName>abc.rpm</internalName>
              <description>Installs the ABC software</description>
              <operatingSystem>centos</operatingSystem>
              <packageType>rpm</packageType>
              <packageLocationType>file</packageLocationType>
              <packageFileLocation>xyz/abc.rpm</packageFileLocation>
          </package>
      </ospackages>
      ```

      Configure the content for the packageLocationType and PackageFileLocation elements.

   c. Copy your RPM file into the following directory:

      `$IA_HOME$/resource/ospackages/xyz`

   d. Restart InstallAnywhere.
2. Add the OS package to the VM tier for your VM configuration:
   a. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
   b. Click the **Appliance Configuration** tab.
   c. In the **Select Appliance Configuration** list, select the appliance configuration that you want to customize.
   d. Click the **VM Configuration** tab.
   e. For VMware vSphere appliances: In the **Select VM Tier** list, select the VM tier that you want to configure.
   f. Click the **OS Packages** subtab. All of the OS packages that are available to install are listed in the **Packages Available to Install** list.
   g. In the **Packages Available to Install** list, select the RPM/Debian package that you want to add and click the **Add** button.

InstallAnywhere adds the OS package to the Packages Selected to Install list.

To install a Debian package, use the following syntax:

`dpkg -i` *`internalNameOfPackage`*

To install an RPM package, use the following syntax:

```
rpm -i --force --quiet internalNameOfPackage
```

Including a Standalone RPM/Debian Package as an Installer



Task ***To include a standalone RPM/Debian package as an installer:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to customize.
4. Click the **VM Configuration** tab.
5. For VMware vSphere appliances: In the **Select VM Tier** list, select the VM tier that you want to configure.
6. Click the **Installers** subtab.
7. Click the appropriate tab to indicate when you want the installer to be executed. Available options are:
   - **Build Time**—Build-time installers are mostly used to push payload for the software appliance.
   - **First Boot Time**— If some configuration needs to be performed before installation, use the First Boot Time installer tab. First-boot-time installers are executed the first time that the virtual appliance is booted.
8. Click the **Add Installer** button. The **Add Installer** dialog box opens.
9. Select the **Other installers** option.
10. Click the **Choose** button next to the **Choose executable** box, and then select the RPM or Debian file.
11. In the **Installation Command** setting, enter one of the following:
    - For a Debian package:

      dpkg -I @@executable@@
    - For an RPM package:

      rpm -i --force --quiet @@executable@@
12. Click **OK**.

Using Custom Boot and Login Scripts for a Virtual Appliance

Virtual appliances built by InstallAnywhere have the capability of automatically calling scripts at specific lifecycle stage: first boot, first login, subsequent boot, and subsequent login of the virtual appliance.

For example, if you have installed an Apache Web server in your virtual appliance, and this needs to be started every time your appliance boots up, you need to write a small shell script that starts your Apache web server each time that it boots up.



Note ▪ *InstallAnywhere appliances also run some internal scripts at each of these lifecycle stages. These internal scripts help safeguard the security of your appliances by regenerating the different SSL keystores and helping the end user set up a default password for the virtual appliance during deployment.*

Because user interaction is not supported during boot time, custom boot scripts (first boot and subsequent boot) should not have any sections that have user interaction. Also, the display of output user messages from the first/subsequent boot are not visible on the console during boot time.

Custom boot scripts (first boot and subsequent boot) support only shell script.



Task ***To specify the custom scripts that you want to use:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to customize.
4. Click the **VM Configuration** tab.
5. For VMware vSphere appliances: In the **Select VM Tier** list, select the VM tier that you want to configure.
6. Click the **Script Info** subtab.
7. Select the **Choose** button next to any of the following settings and select the script file that you want to use:
   - **First Boot Script**—Select a script that runs the first time that the virtual appliance is launched.
   - **First Login Script**—Select a script that runs the first time that an end user logs into this virtual appliance.
   - **Subsequent Boot Script**—Select a script that runs each time that the virtual appliance is launched.
   - **Subsequent Login Script**—Select a script that run each time that an end user logs into this virtual appliance.

About First-Boot Scripts

When a virtual machine in the appliance starts up, it might be necessary to perform some customization based on user input. Using a first-boot script enables you to receive this input from the end user of the appliance. Examples of the sort of input that may be needed are the root password and port numbers of applications.

InstallAnywhere uses two first-boot scripts:

- **InstallAnywhere-reserved first-boot script**—InstallAnywhere includes this script; it always includes the VM execute command. It also performs the following tasks:
  - Prints a welcome message for the end user
  - Displays the EULA and obtains the acceptance
  - Sets the root password
  - Changes the OS language
  - Changes the keyboard layout
  - Sets up the OS for unattended upgrades

- Regenerates the SSL keystores
- Installs first boot InstallAnywhere installers

- **First-boot script that is specified in the InstallAnywhere project**—This script contains customizations that might be needed for specific application files based on user interactions.

For VMware vSphere appliances, InstallAnywhere declares many of the properties for which the user input is usually needed as OVF properties. The VMware software that deploys the OVF prompts for these OVF properties. The values are made available to the scripts using OVF environment variables (just like pre-populated environment variables). So in case of VMware, InstallAnywhere uses a single, unified shell script that contains the following parts:

- A shell script constants section that aliases these OVF environment variables to the shell script using variables.
- The reserved first-boot script.
- The user-defined first-boot script.

For Amazon EC2 appliances, the OVF properties are not present, so InstallAnywhere prompt you for input and includes these customizations in the first-boot scripts.

InstallAnywhere creates a file named `appliance.info` for every VM in the virtual appliance. You can find this file in the `/etc/ia/` subdirectory of the output directory. This file contains following information, which you may want to add to your first-boot scripts:

```
ApplianceVersion=version_of_appliance
ApplianceUUID=uuid_of_appliance
ApplianceVendor=vendor_of_appliance
ApplianceEULALocation=location_of_the_EULA
VirtualmachineName=name_of_the_virtual_machine
VirtualmachineVersion=version_of_the_virtual_machine
VirtualmachineUUID=uuid_of_the_virtual_machine
```

Customizing Product Properties

InstallAnywhere lets you customize properties that are discovered from the installers. These entries will be written to the OVF XML for VMware appliances.



Note ▪ *For InstallAnywhere installers, if you do not specify a response file, but instead specify the product properties, a response file will automatically be created during the installation at first boot or build time. Specifying values in the response file overrides any values that are defined for product properties.*



Task ***To customize the properties of the virtual appliance:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to customize.
4. Click the **VM Configuration** tab.

5. For VMware vSphere appliances: In the **Select VM Tier** list, select the VM tier that you want to configure.
6. Click the **Product Properties** subtab, which lists all of the installers that have been added to this VM configuration. This tab keeps track of all installers (which are called *products* in OVF terminology) that are installed onto your appliance.
7. To view or edit the properties of an installer, click its **View/Edit** button. The Edit Installer Properties Table dialog box opens. This dialog box displays properties (or response options) that need to be supplied to the installer in order for it to silently install on the virtual appliance.
8. Edit properties as needed. For more information, see Edit Installer Properties Table Dialog Box.

When you add an installer to the virtual appliance project, InstallAnywhere detects these properties, unless you specify the properties and default values in a response file. When the appliances are being built, the default values are used to drive the installation.

Identifying Different Virtual Appliances and VMs

The virtual appliances and the VMs within the virtual appliances are uniquely identified using the UUID and the version. It is important to define them uniquely so that this information can be used for maintaining the virtual appliance properly.

When you add a new appliance configuration to your project, InstallAnywhere automatically assigns it an alphanumeric string in the Appliance UUID setting; this setting is on the Appliance Configuration tab in the Build Appliances view of the Build page. The appliance UUID uniquely identifies the appliance. To generate a new ID, click the Generate UUID button next to the Appliance UUID setting.

Similarly, when you add a new VM tier to your project, InstallAnywhere automatically assigns it an alphanumeric string in the UUID of VM setting; this setting is on the VM Configuration tab in the Build Appliances view of the Build page. The appliance UUID uniquely identifies the VM. To generate a new ID, click the Generate UUID button next to the VM UUID setting.

Managing Multiple Tiers for a Virtual Appliance

InstallAnywhere lets you multi-tier virtual appliances for VMware vCenter servers. You can create this sort of virtual appliance to enable your enterprise customers to quickly get started using your multi-tier application in their own virtualization environments, without requiring them to spend a lot of time working on complex configuration tasks. With the multi-tier support, you may want to have a different tier for a Web site, a database, and an application—or whatever fits your requirements.

Note that all VMs in a multi-tier virtual appliance must have the same operating system.



Task ***To add and configure multiple tiers for a virtual appliance:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the VMware appliance configuration that you want to configure.

4. Click the **VM Configuration** tab. The **Select VM Tier** list contains the tiers that are configured for the selected appliance configuration. Each InstallAnywhere project contains a single VM tier by default.
5. To add a new tier, click the **Add VM Tier** button. The **Add VM Tier** dialog box opens.
6. Specify a name for the VM tier, and then click **OK**. InstallAnywhere adds the tier to the **Select VM Tier** list and displays its settings.
7. Configure the tier's settings as needed.

   To view and edit the VM tier's properties, click the View/Edit button on the Product Properties subtab. The Edit Installer Properties Table dialog box opens, enabling you to manage properties (or response options) that need to be supplied to the installer in order for it to silently install on the virtual appliance. When you add an installer to the virtual appliance project, InstallAnywhere automatically discovers these properties. For more information, see Edit Installer Properties Table Dialog Box.
8. Click the **Manage VM Tiers** tab.
9. Configure tiers' settings as needed.

Building a Virtual Appliance

InstallAnywhere provides multiple ways to build a virtual appliance. You can build one or more virtual appliances for a project from within InstallAnywhere. You can also build from the command line. This section of the documentation provides more information.

Building a Specific Appliance Configuration

InstallAnywhere offers the ability to build a particular appliance configuration on demand.



Task ***To build a specific appliance configuration in your project:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to build.
4. Click the **Build Selected** button.

InstallAnywhere builds the selected appliance configuration.

Building the Appliance Configurations That Are Enabled for Project Build

InstallAnywhere offers the ability to build all of the appliance configurations whose **Add to project build** check box is selected.



Task ***To build all appliance configurations in the project build:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Build Project** button.

InstallAnywhere builds all of the appliance configurations that are enabled for the project build.

Building All of the Appliance Configurations in a Project

InstallAnywhere offers the ability to build all of the appliance configurations that are defined in a project.



Task ***To build all appliance configurations:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Build All** button.

InstallAnywhere builds all of the appliance configurations that are defined in the project.

Using build.exe to Build Virtual Appliances from the Command Line

You can use the InstallAnywhere command-line build tool (`build.exe`) to build virtual appliances for the appliance configurations in an InstallAnywhere project. To build through the command line, use the following syntax:

`build.exe` *`Project_File_Path`* `-applianceMode` *`List_of_Appliance_Configurations`*

For example, to build the appliances for the appliance configurations that are named Appliance1, Appliance2, and Appliance3, use the following command line:

```
build.exe C:\MySetups\MyProduct.iap_xml -applianceMode Appliance1 Appliance2 Appliance3
```

To build all of the appliances for the appliance configurations that have the **Add to project build** check box selected, do not list any appliance configurations after the `-applianceMode` parameter—for example:

```
build.exe C:\MySetups\MyProduct.iap_xml -applianceMode
```

To build all of the appliance configurations that are defined in a project, you can use the `-all` command-line option—for example:

```
build.exe C:\MySetups\MyProduct.iap_xml -applianceMode -all
```

Building Virtual Appliances Through Ant Task-Based Build

You can build virtual appliances using an Ant task in your `build.xml` file. A sample Ant task is provided below:

```
<project name="mytask" default="testbuild" basedir=".">
    <taskdef name="buildinstaller" classname="com.zerog.ia.integration.ant.InstallAnywhereAntTask">
        <classpath>
            <pathelement path="C:\Program Files\InstallAnywhere 2026 R1\resource\
```

```
                build\iaant.jar"/>
        </classpath>
    </taskdef>
    <target name="testbuild">
        <buildinstaller IALocation="C:\Program Files\InstallAnywhere 2026 R1"
            IAProjectFile="D:\Demo.iap_xml" buildAppliance="true">
            <applianceConfiguration name="A1" />
            <applianceConfiguration name="A2" />
        </buildinstaller>
    </target>
</project>
```

Build Output for VMware vSphere 5 Virtual Appliances

The following image is an example of the contents of a typical InstallAnywhere output directory for a VMware vSphere 5 virtual appliance:

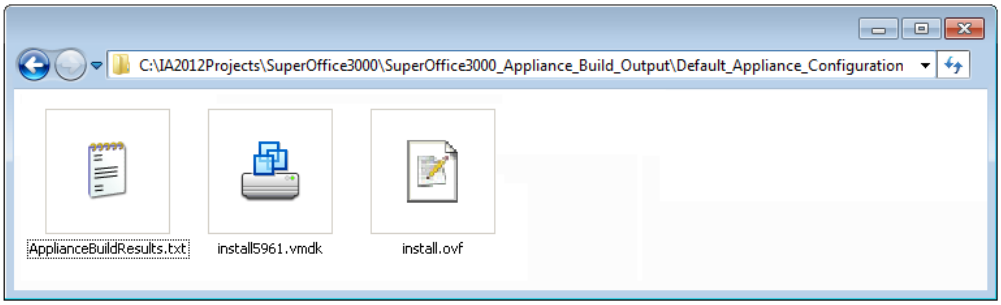


Figure 8-2: VMware Virtual Appliance Build Output Directory

Build Results File: ApplianceBuildResults.txt

In addition to the build log, each appliance build produces a file named `ApplianceBuildResults.txt` in the build output directory for each appliance configuration. The contents of this file gives a glimpse of the results of the appliance builds.

Example for a VMware Build with Deployment Enabled

```
Build Results
    Build Output Location = location_of_output_Appliance_OVF/OVA
Deployment Results
    IP of appliance = IP_of_the_appliance
    Appliance Name = name_of_appliance
    Appliance URL = URL_of_appliance
```

Build Output for Amazon EC2 Virtual Appliances

The following image is an example of the contents of a typical InstallAnywhere output directory for an Amazon EC2 virtual appliance:

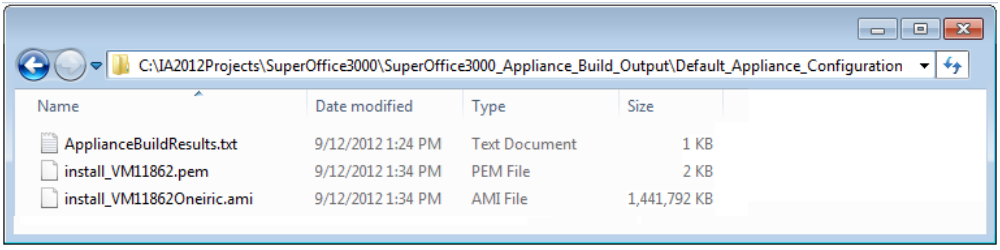


Figure 8-3: Appliance Build Output Directory

Build Results File: ApplianceBuildResults.txt

In addition to the build log, each appliance build produces a file named `ApplianceBuildResults.txt` in the build output directory for each appliance configuration. The contents of this file gives a glimpse of the results of the appliance builds.

Example for an Amazon Appliance Build with Deployment Enabled

```
Build Results
    Build Output Location = location_of_built_AMI
Deployment Results
    AMI ID = ami_id
    Key Pair = name_of_keypair
    Key Pair Location = location_of_key_pair
    Instance ID = instance_id
    Instance Public IP = public_IP_of_instance
    Instance Public DNS = public_DNS_of_instance
    Instance Private IP = private_IP_of_instance
    Instance Private DNS = private_DNS_of_instance
    Appliance URL = URL_of_appliance
```



Important ▪ *The key pair information is very important. This key pair is used to log in to this instance for any maintenance activity. Losing this key pair will render the deployed instance inaccessible to the user.*

Example for an Amazon Appliance Build with Deployment Disabled

```
Build Results
    Build Output Location = location_of_built_AMI
```

Deploying a Virtual Appliance

When you are done building your virtual appliance, you can deploy the virtual appliance to a public cloud such as Amazon EC2 or distribute it to your enterprise customers for provisioning within their virtual environments.

You can have InstallAnywhere deploy your virtual appliance automatically at build time. As an alternative, you can have InstallAnywhere deploy your virtual appliance on demand, at any time.



Task ***To specify whether your virtual appliance should be deployed automatically at build time:***

1. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
2. Click the **Appliance Configuration** tab.
3. In the **Select Appliance Configuration** list, select the appliance configuration that you want to configure.
4. To indicate that you want InstallAnywhere to deploy the virtual appliance to the target hypervisor automatically at build time, select the **Deploy this Appliance** check box.

   To indicate that you do not want InstallAnywhere to deploy the virtual appliance automatically, clear the **Deploy this Appliance** check box.

The methods for deploying the virtual appliance on demand vary, depending on which cloud infrastructure your appliance supports. For more information, see:

- Deploying a VMware vSphere 5 Virtual Appliance on Demand
- Deploying an Amazon EC2 Virtual Appliance on Demand

Deploying a VMware vSphere 5 Virtual Appliance on Demand

InstallAnywhere enables you to deploy a VMware vSphere virtual appliance to a vSphere/vCenter server on demand, at any time.



Task ***To deploy a VMware vSphere 5 virtual appliance at any time:***

1. Open the VMware vSphere 5 client and connect to your VMware vSphere/vCenter server.
2. On the **File** menu of the VMware vSphere 5 client, select **Deploy OVF Template**. The **Deploy OVF Template** wizard opens.
3. Select the `.ovf/.ova` file that was generated by the InstallAnywhere appliance build.
4. Proceed through the wizard and specify settings, including:
   - Adjusting the size of the hard drive required.
   - Choosing the automatic power on option.
5. When you complete the wizard, the VMware vSphere 5 client will deploy and power on your virtual appliance.
6. Open your virtual appliance using the console.

Upon the first login, the first boot script will run and prepare your installer.



Note ▪ *If you do not have a VMware vCenter server, before you click the Build Appliance button, select the* ***VMware vSphere compatible Appliance*** *option under OVF Appliance Type on the Appliance Configuration tab of*

the Build Appliances view. Then use the VMware vSphere Client 5 to connect to your VMware vSphere server and deploy your .ovf/.ova file.

Deploying an Amazon EC2 Virtual Appliance on Demand

InstallAnywhere enables you to deploy an Amazon EC2 virtual appliance on demand, at any time, through the Amazon Web Services Console or through command-line tools.

- Deploying an Amazon Virtual Appliance Through the Amazon Web Services (AWS) Management Console
- Deploying an Amazon EC2 Virtual Appliance Through Command-Line Tools

Deploying an Amazon Virtual Appliance Through the Amazon Web Services (AWS) Management Console

In general terms, deploying a virtual appliance on Amazon EC2 involves creating the necessary environment on the Amazon EC2 cloud for that virtual appliance and running the virtual appliance on the Amazon cloud.

The Amazon Web Services (AWS) Console is used for the deployment of virtual appliances on the Amazon EC2 cloud. When deploying virtual appliances, you use two major sections of the AWS Management Console: EC2 and S3.

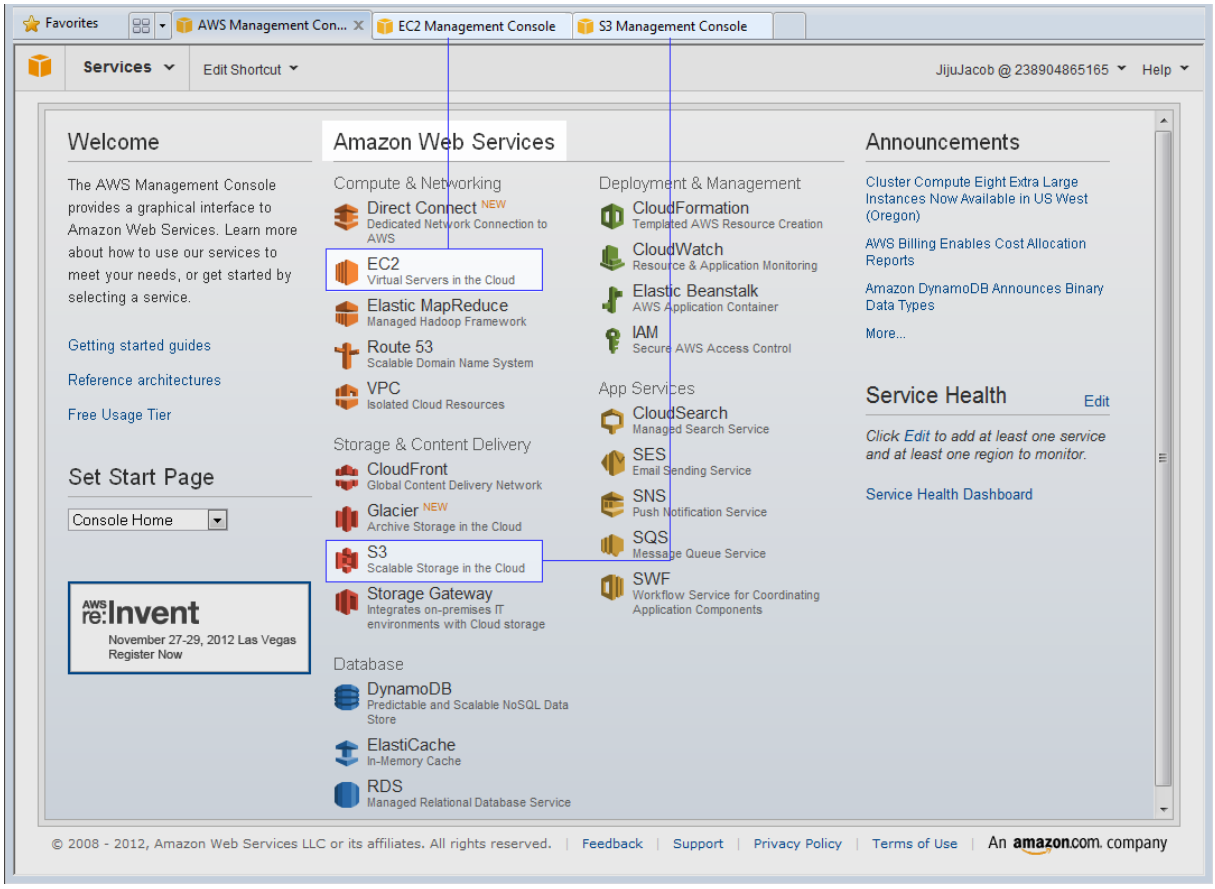


Figure 8-4: AWS Management Console Main View

To deploy a virtual appliance on the Amazon EC2 cloud, you need to perform four basic steps:

Table 8-10 ▪ Steps to Deploy a Virtual Appliance Using AWS Management Console

AWS Management Console Section	Virtual Appliance Deployment Steps
S3 (Simple Storage Service)	• **Step 1: Create S3 bucket**—First you need to create an S3 bucket. • **Step 2: Upload AMI image bundles**—Next, you upload AMI image bundles and make all bundle files public.
EC2 (Elastic Compute Cloud)	• **Step 3: Create AMI image**—Then you register the image bundles to create an AMI image. • **Step 4: Launch an EC2 instance**—Finally, you launch an EC2 instance.

S3 (Simple Storage Service)

Using the AWS Management Console, you can easily manage Amazon S3 buckets and objects. You can create buckets, upload objects, and manage all aspects of your Amazon S3 resources.

The S3 Console window gives you the flexibility to create a bucket. You can upload files into a bucket or remove files from a bucket. The files stored in the Amazon EC2 cloud can be accessed from anywhere in the world.

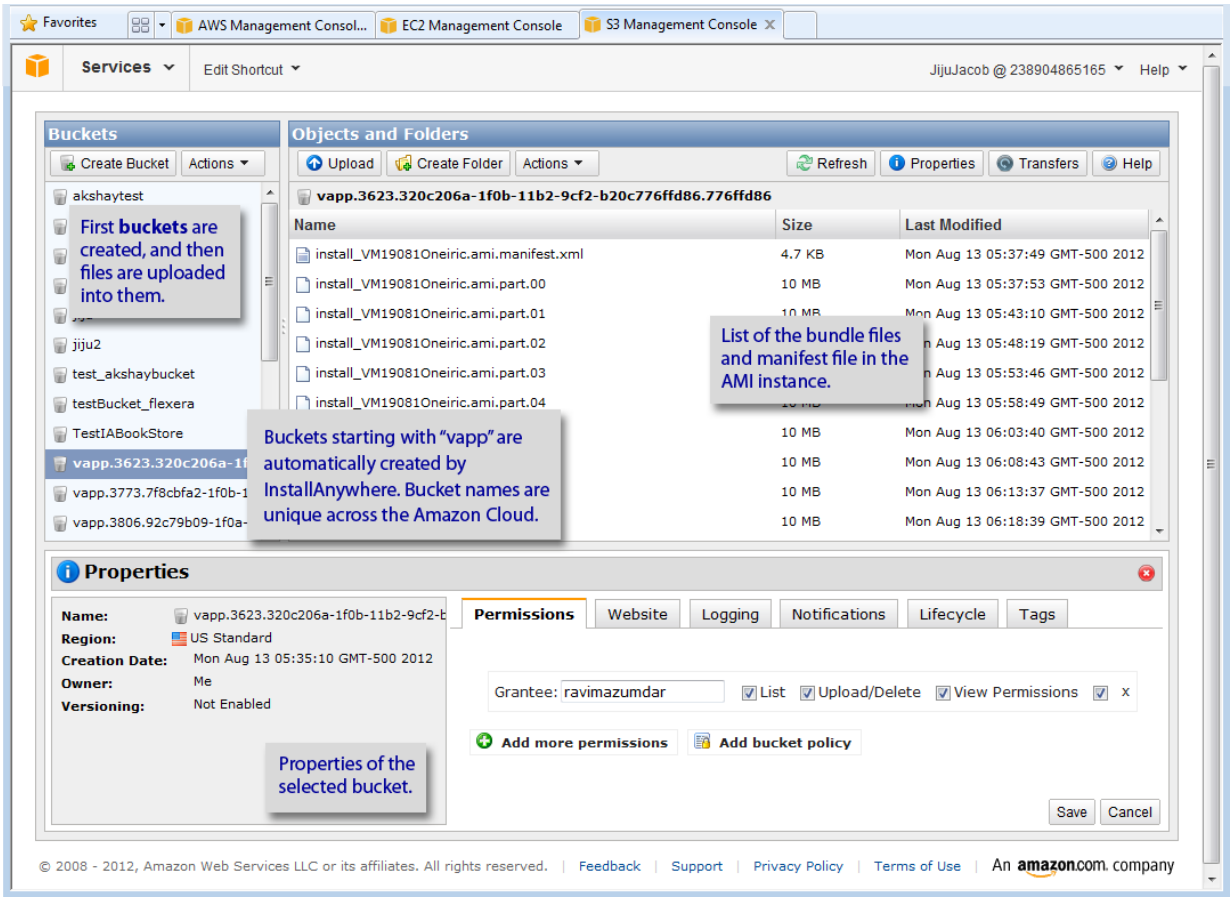


Figure 8-5: S3 Console Window

When assigning a bucket name, you need to make sure that the name is unique across the Amazon cloud; the name should not be duplicated. The InstallAnywhere appliance build creates buckets with unique names. The bucket name is a combination of random numbers and UUIDs, such as:

`vapp.3773.7f8cbfa2-1f0b-11b2-a6ab-e09dc89cb12f.c89cb12f`

InstallAnywhere can automatically upload AMI image bundle files to the AWS Management Console. The uploaded files need to be made public in order to facilitate the launching of the EC2 instance.

EC2 (Elastic Compute Cloud)

On the EC2 Console window of the AWS Management Console, you can start and stop EC2 instances, view and perform actions on running instances, and manage Elastic Block Store volumes. You can also view the configurations that you have set up (such as security groups, key pairs, and Elastic IPs), and modify them using interactive controls.

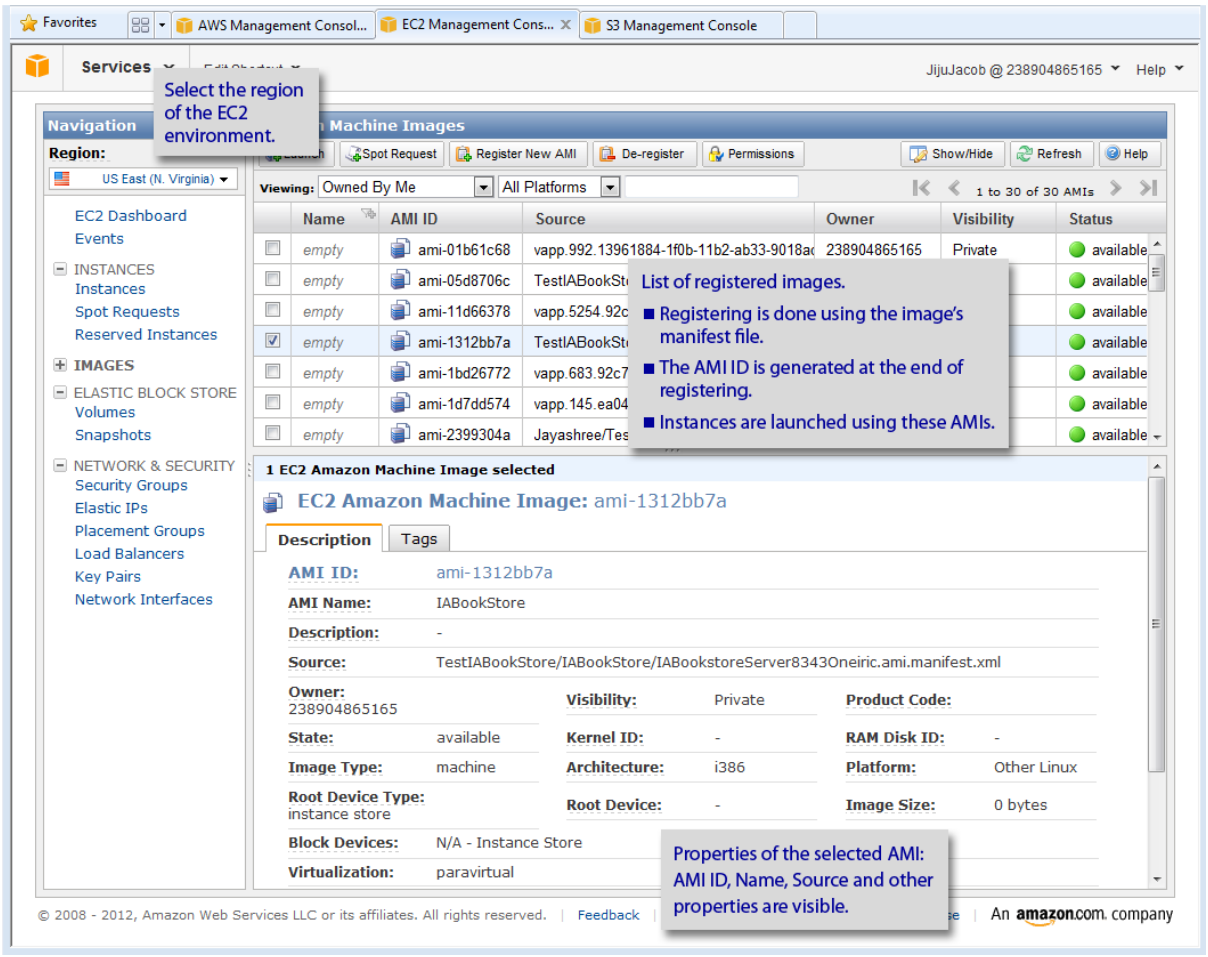


Figure 8-6: EC2 Console Window

You can use the EC2 Console window to register a prebuilt AMI image (virtual machine) and launch an instance of it. This is, in effect, running a rented virtual machine on the Amazon cloud.

The AMI image bundles, which you need to upload to the Amazon S3 bucket, will be used to register an AMI. A registered AMI image is a kind of virtual machine. This virtual machine will run on the Amazon cloud when it is launched. A running virtual machine is called a running *instance*. You can view the number of running instances you have by clicking on the Instances tab of the EC2 Console window.

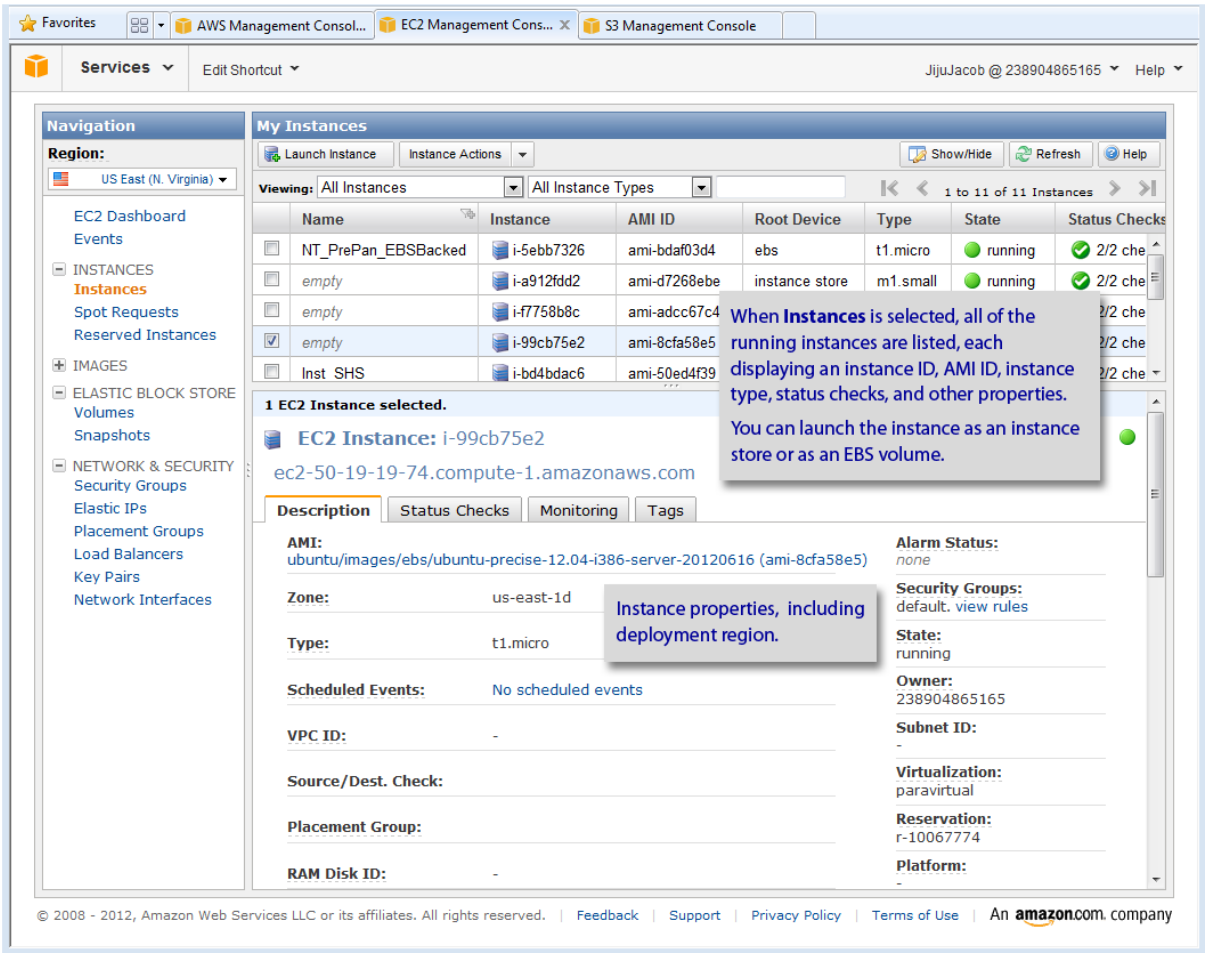


Figure 8-7: EC2 Instance Descriptions

Amazon EC2 provides the flexibility to choose from a number of different instance types to meet your computing needs: Small Instance, High CPU Medium Instance, or a Medium Instance. Each instance provides a predictable amount of dedicated compute capacity and is charged per instance-hour consumed.

Note ▪ *Specify the instance type of your virtual appliance on the Hardware subtab of the VM Configuration tab in the Build Appliances view.*

Deploying an Amazon EC2 Virtual Appliance Through Command-Line Tools

You can use command-line tools to deploy an Amazon EC2 virtual appliance.



Task ***To deploy an Amazon EC2 virtual appliance on demand through command-line tools:***

1. Make sure that the Amazon `ec2-ami-tools` and `ec2-api-tools` packages are installed on your host machine.
2. Copy the `.ami` files to the host machine.
3. Use the `ec2-bundle-image` command to split this file into bundles:

   ```
   $> ec2-bundle-image -i <Location_Of_AMI> --cert <Path_To_X509_Certficate> --privatekey
   <Path_To_Private_Key> -u <AWS_Account_number>
   ```

   Note ▪ *This requires that the* `ec2-api-tools` *are installed.*

   This process will split your AMI into a set of 10 MB segments and also provide a manifest XML file.
4. Upload these files/bundles on to the Amazon cloud using `ec2-upload-bundle` by providing relevant parameters related to your account on Amazon:

   ```
   $> ec2-upload-bundle -b <Name_Of_Amazon_S3_Bucket> -m <Location_Of_Amazon_Bundle_Manifest_XML_File>
   -a <Amazon_Access_Key> -s <Amazon_Secret_Key>
   ```
5. Once these files are uploaded, you need to register the files as an AMI (Amazon machine Image) by using the following command:

   ```
   $> ec2-register <your-s3-bucket>/<image.manifest.xml>
   ```

   This process will write an AMI-ID to your command prompt.

   Note ▪ *The* `ec2-register`*,* `ec2-create-keypair`*, and* `ec2-launch-instance` *has a GUI version using Amazon Web Services (AWS) Management Console.*
6. To access this image, once hosted on the Amazon infrastructure, you need to have a key-pair. To create a key-pair, use the `ec2-create-keypair` command:

   ```
   $> ec2-create-keypair <keyName> --aws-access-key <Amazon_Access_Key> --aws-secret-key
   <Amazon_Secret_Key>
   ```

   This process will create a key-pair, store the public key of the key-pair with the Amazon EC2, and the private key is displayed in the console.
7. Copy the generated (/displayed) key-pair key to a plain text file and store it with the extension `.pem`.
8. You can now launch the appliance using the `ec2-run-instances` command:

   ```
   $> ec2-run-instances $<Amazon_AMI_ID> -k <pathToKeyPair> -n <number of instances> [-z
   <Amazon_Availability_Zone> ] --kernel $kernel_id --aws-access-key <Amazon_Access_Key> --aws-secret-
   key <Amazon_Secret_Key>
   ```

   The kernel ID corresponding to your image can be located by looking at the AKI-ID column in the AMI locator hosted at http://cloud.ubuntu.com/ami/.



Tip ▪ *For a sample project, see Using the IABookstore Sample for Creating Virtual Appliances.*

Using the IABookstore Sample for Creating Virtual Appliances

The IABookStore is a free-to-use J2EE sample along with its corresponding InstallAnywhere project that is shipped along with InstallAnywhere to help you explore the functionality of virtual appliances.

Table 8-11 ▪ IABookstore Sample

Resource	Location
Location of the InstallAnywhere project for IABookStore application	`$IA_HOME$/IABookStore`
Location of the web archive (.war) file for the IABookStore application	`$IA_HOME$/IABookStore/data/IABookStore.war`



Note ▪ *Before you open the IABookStore project for building its virtual appliance, you need to copy the Linux JDK 6 VM pack (Sun JRE 1.6.0_26) to the `$IA_HOME$\resource\installer_vms` directory.*

You can download this VM pack from the Revenera Web site: https://www.revenera.com/install/products/installanywhere/installanywhere-files-utilities.html. First click Linux Intel (x86) and then download Sun JRE 1.6.0_26 Linux (32-bit) INTEL.

To use this sample to create virtual appliances, open the InstallAnywhere project, use the settings in the credential area on the Appliance Configuration tab (in the Build Appliances view on the Build page) to define account settings for your environment, and build the virtual appliance.



Important ▪ *After you have added the installer to your project via the Installers subtab of the VM Configuration tab in the Build Appliances view, edit the product's entry on the Product Properties subtab. To do so, click the View/Edit button on the Product Properties subtab; the Edit Installer Properties Table dialog box opens. Make the following changes:*

- *Set the `USER_INSTALL_DIR` key to `/IABookstore`.*
- *Delete the `USER_SHORTCUTS` key.*

The final bookstore application should launch at:

```
http://<url_of_virtual_machine>:8080/IABookStore
```

Using the IABankingApplication Sample for Creating Multi-Tier Virtual Appliances

`IABankingApplication.iap_xml` is a sample InstallAnywhere project that you can use to explore InstallAnywhere's support for multi-tier virtual appliances. This project file, along with application files, are installed in the following location:

IA_HOME/IABankingApplication

The IABankingApplication.iap_xml project is configured to build a three-tier virtual appliance with the following tiers:

- First tier—This tier is configured to host a Derby database server and an Apache Tomcat application server. This tier has a Start/Stop Order value of 0.
- Second tier—This tier is configured to host a second Apache Tomcat application server. This tier has a Start/Stop Order value of 1.
- Third tier—This tier is configured to host a load-balanced Apache HTTPd Web server. This tier has a Start/Stop Order value of 2.

Each of the virtual machines (one associated with each tier) is configured to use a 32-bit CentOS VM template.

The multi-tier virtual appliances uses the following technologies:

- JAAS authentication
- Java with J2EE using hibernate, servlets and JSPs
- Apache Tomcat server
- Load-balanced HTTPd server
- Apache Derby database
- SQL



Task ***To build the sample three-tier IABankingApplication virtual appliance:***

1. Obtain a JRE VM pack for 32-bit Linux systems and add it to the following location:

   IA_HOME/resource/installer_vms

2. Obtain a 32-bit CentOS VM template for virtual appliances and add it to the following location:

   IA_HOME/resource/baseline_vms

3. Open the IABankingApplication.iap_xml project in InstallAnywhere.
4. Build all configurations of the installers.
5. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
6. Click the **Appliance Configuration** tab.
7. In the **Connection Settings** area, enter the connection settings and credentials for your VMware vCenter server.
8. On the **VM Configuration** tab, configure the following installers:
   - A prebuilt installer (DB_Tomcat1 configuration) for DB_FirstAppServer. Enter the following product properties:
     - USER_INSTALL_DIR=/DBServer
     - APACHE_SERVER_ROOT=/DBServer/Downloaded/apache-tomcat-7.0.27

- A prebuilt installer (Tomcat2 configuration) for SecondAppServer. Enter the following product properties:
  - USER_INSTALL_DIR=/AppServer2
  - APACHE_SERVER_ROOT=/AppServer2/Downloaded/apache-tomcat-7.0.27
- An installer out of the current project (WebServer configuration). This installer is for the Web server. Enter the following product properties:
  - USER_INSTALL_DIR=/WebServer
  - HTTPD_SERVER_ROOT=/etc/httpd

9. Select a 32-bit CentOS VM template for each of the three tiers.
10. Build the virtual appliance.

To access the application, use the following URL:

http://*IP_Address_of_WebTier*/IAMultiTierApplication

Use the following credentials:

- Test credentials: 10001/test123
- Admin credentials: 10000/admin



Note ▪ *Modifying the VM name on the VM Configuration tab in the Build Appliances view and/or the product name on the Product Properties subtab on the VM Configuration tab requires corresponding modifications to the first boot installation scripts.*

9

Reference

Reference information for InstallAnywhere is organized into the following sections:

Table 9-1 ▪ InstallAnywhere Reference

Section	Description
Advanced Designer Reference	Describes the pages, views, and settings in the Advanced Designer.
Wizard Reference	Describes details about each of the wizards that are available in InstallAnywhere.
Menu Reference	Provides descriptions of the InstallAnywhere menus and menu commands that are available in the Advanced Designer interface.
Dialog Box Reference	Includes details about common InstallAnywhere dialog boxes: New Project, Open Project, InstallAnywhere Preferences, About InstallAnywhere, and others.
Actions	Lists the general actions, panel actions, install actions, and System i (i5/OS) actions that are available in the Advanced Designer and describes their purpose and implementation.
Rules Reference	Identifies the rules that are available in the Installer Rules view on the Project page of the Advanced Designer, as well as 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 9-1 ▪ 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 Reference

The InstallAnywhere Advanced Designer has an intuitive, graphical interface that enables you to manage all aspects of your installation projects. The Advanced Designer has two main navigational elements:

- The primary navigational element is the horizontal row of buttons near the top of the InstallAnywhere interface, directly below the menu bar. You can click these buttons to open various pages in the Advanced Designer.
- The secondary navigational element is the vertical column of buttons on the left side of the InstallAnywhere interface. The list of buttons in this area changes, depending on which page in the Advanced Designer is currently open.

The following table describes each of the pages that are available from the primary navigation—the vertical row of buttons—in the Advanced Designer.

Table 9-2 ▪ Pages in the Advanced Designer

Page	Description
Project Page	Displays project information; project settings; options to bundle or define acceptable VMs; and settings for defining the product, file installation, upgrade, and localization preferences.
Installer UI Page	Sets user interface options such as the default UI mode, splash screen images, billboard graphics, panel backgrounds, installer help, and more.
Organization Page	Provides tools to arrange install sets, product features, components, merge modules, and database server and application server hosts.
Sequence Page	Contains views that you can use to configure the order of panels and actions that occur at run time.

Table 9-2 ▪ Pages in the Advanced Designer (cont.)

Page	Description
Build Page	Contains settings for building an installer, creating and selecting build configurations, setting build targets, bundling VM packs, and defining distribution options. Also provides settings for defining and building a virtual appliance.

Project Page

The Project page contains views that let you configure project information; options to bundle or define acceptable VMs; and settings for defining the product, file installation, and localization preferences. The Project page includes the following views:

Table 9-3 ▪ Views on the Project Page

View	Description
General Settings View	Contains settings for basic information about your project, your company, and your product.
Platforms View	Sets platform-specific options for OS or OS X, Windows, UNIX, and System i.
Locales View	Provides tools to define locale settings for your project.
Manage Expressions View	Provides a rule expression manager that allows you to create complex rule expressions at the project level, associate a rule expression to a file extension, and more.
Installer Rules View	Defines rules that apply to the installers generated by the project.
JVM Settings View	Defines JVM search settings. • General Settings tab—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 that you are installing.
Variables View	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.
Advanced View	Specify settings for Maintenance Mode, Instance Management, Rollback, and build configuration tags.
Upgrades View	Configure settings to create an upgrade installer that uninstalls earlier versions of your product—if they are present—before installing the new version.

General Settings View

Use the General Settings view on the Project page to define basic information about your project, your company, and your product.

The settings in the General Settings view are organized into the following main categories:

- Project Information
- Product Information
- Vendor Information
- File Settings
- Software Tag
- Log Settings

Project Information

Use the Project Information area of the General Settings view to specify details such as the installer name and the product name. The following settings are available in this area.

Table 9-4 ▪ Project Properties Settings

Setting	Description
Installer Title	Enter the title that you want to display in the title bar of the installers that are built from the project.
Installer Name	Enter the file name—without the period or the file extension—that InstallAnywhere should use for this project's installers. For example, if you enter **myinstall**, the Windows-based installer file name will be `myinstall.exe`, the pure Java installer file name will be `myinstall.jar`, and the UNIX-based installer file name will be `myinstall.sh`.  ***Note*** ▪ *For UNIX-based installers, InstallAnywhere replaces any spaces in the installer name with underscores.*
Product Name	Enter the name of the product to be installed.  ***Note*** ▪ *This setting also determines the name of the artifacts that are built for merge module projects.*
Save Location	This read-only setting shows the location where the project file is saved.
Build Location	This setting shows the location where your project's build output is stored. To specify a custom build location, click the ellipsis button (...) in this setting. To revert back to the default path, click the Default button in this setting.

Table 9-4 ▪ Project Properties Settings (cont.)

Setting	Description
Last Saved	This read-only setting shows the date and time your project was last saved.
Last Built	This read-only setting shows the date and time at which installers from your project were last built.
Always Generate Response File	Specify whether you want to enable 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 used to provide settings for a silent installation. When the Yes 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.*
Prevent Automatic Reading From Default Response File	Specify whether you want to disable the installers' ability to automatically read the default response file. When this setting is set to **Yes**, the `installer.properties` file will be used as a response file only if it is passed with `-f` command-line switch and not used by default, when the installer is launched. By default, this setting is set to **No**.
Prevent Generation of Default Response File in Silent Mode	Specify whether you want to prevent end users from being able to generate the default response file in silent mode installation or uninstallation. When this setting is set to **Yes**, the `installer.properties` file (default response file) will be generated only if it is passed with -r command-line switch and not generated by default, when the installer and unistaller are launched in silent mode. By default, this setting is set to **No**.
Prevent Multiple Simultaneous Launches of an Installer or Uninstaller	Specify whether you want to prevent end users from being able to launch multiple simultaneous instances of the same installer or uninstaller (as identified by having the same product code). Invoking multiple simultaneous instances of the installer or uninstaller could corrupt the InstallAnywhere registry.  ***Note ▪*** *This setting has no effect if multiple end users are connected to a machine, such as in a UNIX environment, and each end user launches the same installer simultaneously. In this scenario, it is possible for multiple end users to launch simultaneous instances of the installer or uninstaller, regardless of whether Yes or No is selected in this setting.*

Table 9-4 ▪ Project Properties Settings (cont.)

Setting	Description
Disable Cancel Button When Install Is in Progress	Specify whether you want to prevent end users from being able to cancel an installation using the Cancel button (or the close button). Disabling the Cancel button during installation helps to avoid unfinished installations.  ***Note ▪*** *This setting is not applicable for uninstallers; by default, the Cancel button is disabled in uninstallers.*

Product Information

Use the Product Information area of the General Settings view to specify details such as the product code and version number. The following settings are available in this area.

Table 9-5 ▪ Product Information Settings

Setting	Description
Product Code	Enter a GUID that uniquely identifies your product in the registry. You may have several installed products that have the same name; use this ID to track a specific instance of the product. To have InstallAnywhere generate a different GUID for you, click the Generate Product Code button ({...}) in this setting. The GUID must be in the format that is displayed.
Version	Enter the version number of your product.
Copyright	Enter the copyright year for your product.
Product Info URL	Specify a URL where end users can obtain information about your product.
Support URL	Enter the URL that you would like end users to visit for technical support information for your product.
Product Description	Enter a description of your product for the product registry. To learn more, see Product Registry.
Update the Product Registry	Specify whether you want to allow this project's installers to update the InstallAnywhere product registry on target systems. The default value for this setting is Yes. To learn more, see Product Registry.

Vendor Information

Use the Vendor Information area of the General Settings view to specify details such as the vendor name and ID. The following settings are available in this area.

Table 9-6 ▪ Vendor Information Settings

Setting	Description
Vendor Name	Enter the name of the company that created the product.
Vendor ID	Enter a GUID that uniquely identifies the name of the company that created the product. To have InstallAnywhere generate a different GUID for you, click the button in this setting.
Vendor Home Page	Enter a general URL for your company or product.
Vendor Email	Enter an email address for your company or product.
Defaults for New Projects	To save the current values that are entered for the Vendor Description settings for use with any new projects that you create, click the Save As Defaults for New Projects button in this setting.

File Settings

Use the File Settings area of the General Settings view to specify file-related behavior. The following settings are available in this area.

Table 9-7 ▪ File Settings

Setting	Description
Replace In-Use Files After Restart (Windows Only)	Specify whether you want to have your installer replace files that are locked (in use) after the target system restarts. This setting applies to Windows-based target systems.

Table 9-7 ▪ File Settings (cont.)

Setting	Description
Default File Overwrite Behavior	This setting lets you specify the behavior that occurs if the installation contains a file whose name and target location matches a file that is already present on a target system. Available options are: • **Always overwrite**—The file on the target system is overwritten, regardless of file's timestamp. • **Never overwrite**—The file on the target system remains as is, regardless of file's timestamp. The installation does not overwrite it with the version of the file that is in the installation. • **Overwrite if older, do not install if newer**—If the file in the installation is newer (has a later timestamp) than the file that is present on the target system, replace the file on the target system with the one in the installation. If the file in the installation is older (has an earlier timestamp) than the file that is present on the target system, the file on the target system remains as is. • **Overwrite if older, prompt if newer**—If the file in the installation is newer (has a later timestamp) than the file that is present on the target system, replace the file on the target system with the one in the installation. If the file in the installation is older (has an earlier timestamp) than the file that is present on the target system, display a prompt that asks the end user whether the installation should replace the file on the target system. • **Prompt if older, do not install if newer**—If the file in the installation is newer (has a later timestamp) than the file that is present on the target system, display a prompt that asks the end user whether the installation should replace the file on the target system. If the file in the installation is older (has an earlier timestamp) than the file that is present on the target system, the file on the target system remains as is. • **Always prompt user**—Display a prompt that asks the end user whether the installation should replace the file on the target system.

Table 9-7 ▪ File Settings (cont.)

Setting	Description
File Modification Timestamp Behavior	Use this setting to specify how you want to timestamp the files that are installed on target systems. To change the behavior, click the ellipsis button (...) in this setting. The File Modification Timestamp Behavior dialog box opens, enabling you to select the appropriate option: ● **Preserve Timestamp**—Maintain the existing timestamp on files. That is, use the time at which the file was last modified as shown by the operating system where the file was created or saved. For example, if you modify a file and then add it to your project with the Preserve Timestamp option selected, the timestamp that the installer uses on target systems reflects the date on which you last modified the file, not the time at which it was installed. ● **Install Time Timestamp**—Use the time at which the files are installed on a target system for the files' timestamps. With this option, the creation timestamp and the modification timestamp for a file match, and they reflect the date and time at which the files are installed on a target system. ● **Specify Timestamp**—Use the date and time that you specify on the File Modification Timestamp Behavior dialog box for the files' timestamps.  ***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).*

Software Tag

Use the Software Tag area in the General Settings view to specify whether you want to include an ISO/IEC 19770-2 software identification tag in your installer. If a tag is included, this area also lets you specify the identification information for the tag. For more information, see Including a Software Identification Tag for Your Product.

The following settings are available in the Software Tag area

Table 9-8 ▪ Software Tag Settings

Control	Description
Enable Software Tagging	To specify that you want to include an ISO/IEC 19770-2 software identification tag in your installation, select Yes in this setting, and then use the other tag-related settings in this view to specify the identification information for the tag. Yes is selected by default.
Require Software Entitlement	To specify that you want to require your product to have a corresponding software entitlement in order for software reconciliation to be considered successful, select this check box. In general, if the software must be purchased, this check box should be selected; if the software is free, this check box should be cleared.

Table 9-8 ▪ Software Tag Settings (cont.)

Control	Description
Product ID	Enter a unique ID (UID) that identifies the product line for your product. This and all future releases of your product should use the same product ID for the product line to which they belong, even if the product line is rebranded. To have InstallAnywhere generate a different UID for you, click the Generate UID button in this setting.
Unique ID	Enter a unique ID (UID) that identifies the specific version of this specific product. To have InstallAnywhere generate a different UID for you, click the Generate UID button in this setting. Note that InstallAnywhere uses the value that you enter as part of the name of the tag file (`TagCreatorID_UniqueID.swidtag`). Therefore, the ID that you enter must not contain any characters that are invalid for file names.
Tag Creator Name	Enter the name of the organization that created the tag. As an alternative, if you want to use the same value that is entered in the Vendor Name setting in the Vendor Information area of this view, select the check box in the Tag Creator Name setting. Note that if the Vendor Name setting is blank, the check box in the Tag Creator Name setting is disabled.

Table 9-8 ▪ Software Tag Settings (cont.)

Control	Description
Tag Creator ID	Enter the registration ID of the organization that created the tag. This ID helps to differentiate between different legal organizations that have the same creator name but are in different countries. The convention for the registration ID is as follows: **regid.YYYY-MM.ReversedDomainName,division** The registration ID consists of the following parts: ● **regid.**—The string *regid* indicates that the XML portion is a registration ID for a software identification tag. A period (.) must be included after this string. ● **YYYY-MM.**—This part of the registration ID identifies the first full month (MM) and the year (YYYY) in which the domain name was owned by the tag creator. For example, if you are creating the tag and you purchased the domain name February 15, 1999, you would use **1999-03** in this part of the registration ID, since the first full month the domain name was owned was March (03), and the year was 1999. The year and month must be separated by a dash. ● **ReversedDomainName**—This part identifies the reversed domain name of the organization that is creating the software identification tag. For example, for the flexerasoftware.com domain name, the reversed domain name is: **com.flexerasoftware** ● **,division**—This optional part starts with a comma (,), and is followed by an additional string. You can enter a string that helps to distinguish between different divisions or areas of the organization. If you do not want to use this optional distinguishing part of the registration ID, do not include the comma or an additional string in your entry. Note that InstallAnywhere uses the value that you enter as part of the name of the tag file (*TagCreatorID_UniqueID.swidtag*). Therefore, the ID that you enter must not contain any characters that are invalid for file names. If you enter invalid text in this setting, InstallAnywhere displays the text in red. In addition, a build warning is included in the console/build log.
Software Creator Name	Enter the name of the organization that created the software. This setting is optional. If you leave this setting blank and select the check box in this setting, InstallAnywhere uses the value of the Tag Creator Name setting for the name of the software creator.

Table 9-8 ▪ Software Tag Settings (cont.)

Control	Description
Software Creator ID	Enter the registration ID of the organization that created the software. This ID helps to differentiate between different legal organizations that have the same creator name but are in different countries. This setting is optional. If you leave this setting blank and select the check box in this setting, InstallAnywhere uses the value of the Tag Creator ID setting for the software creator ID. The convention for the registration ID is as follows: **regid.YYYY-MM.ReversedDomainName,division** The registration ID consists of the following parts: ● **regid.**—The string *regid* indicates that the XML portion is a registration ID for a software identification tag. A period (.) must be included after this string. ● **YYYY-MM.**—This part of the registration ID identifies the first full month (MM) and the year (YYYY) in which the domain name was owned by the tag creator. For example, if you are creating the tag and you purchased the domain name February 15, 1999, you would use **1999-03** in this part of the registration ID, since the first full month the domain name was owned was March (03), and the year was 1999. The year and month must be separated by a dash. ● **ReversedDomainName**—This part identifies the reversed domain name of the organization that is creating the software identification tag. For example, for the flexerasoftware.com domain name, the reversed domain name is: **com.flexerasoftware** ● **,division**—This optional part starts with a comma (,), and is followed by an additional string. You can enter a string that helps to distinguish between different divisions or areas of the organization. If you do not want to use this optional distinguishing part of the registration ID, do not include the comma or an additional string in your entry. Note that InstallAnywhere uses the value that you enter as part of the name of the tag file (*TagCreatorID_UniqueID.swidtag*). Therefore, the ID that you enter must not contain any characters that are invalid for file names. If you enter invalid text in this setting, InstallAnywhere displays the text in red. In addition, a build warning is included in the console/build log.
Software Licensor Name	Enter the name of the organization that owns the copyright for the software. This setting is optional. If you leave this setting blank and select the check box in this setting, InstallAnywhere uses the value of the Tag Creator Name setting for the name of the software licensor name.

Table 9-8 ▪ Software Tag Settings (cont.)

Control	Description
Software Licensor ID	Enter the registration ID of the organization that owns the copyright for the software. This ID helps to differentiate between different legal organizations that have the same licensor name but are in different countries. This setting is optional. If you leave this setting blank and select the check box in this setting, InstallAnywhere uses the value of the Tag Creator ID setting for the software licensor ID. The convention for the registration ID is as follows: **regid.YYYY-MM.ReversedDomainName,division** The registration ID consists of the following parts: ● **regid.**—The string *regid* indicates that the XML portion is a registration ID for a software identification tag. A period (.) must be included after this string. ● **YYYY-MM.**—This part of the registration ID identifies the first full month (MM) and the year (YYYY) in which the domain name was owned by the tag creator. For example, if you are creating the tag and you purchased the domain name February 15, 1999, you would use **1999-03** in this part of the registration ID, since the first full month the domain name was owned was March (03), and the year was 1999. The year and month must be separated by a dash. ● **ReversedDomainName**—This part identifies the reversed domain name of the organization that is creating the software identification tag. For example, for the flexerasoftware.com domain name, the reversed domain name is: **com.flexerasoftware** ● **,division**—This optional part starts with a comma (,), and is followed by an additional string. You can enter a string that helps to distinguish between different divisions or areas of the organization. If you do not want to use this optional distinguishing part of the registration ID, do not include the comma or an additional string in your entry. Note that InstallAnywhere uses the value that you enter as part of the name of the tag file (*TagCreatorID_UniqueID.swidtag*). Therefore, the ID that you enter must not contain any characters that are invalid for file names. If you enter invalid text in this setting, InstallAnywhere displays the text in red. In addition, a build warning is included in the console/build log.

Log Settings

Use the Log Settings area of the General Settings view to specify preferences for logging successful and failed run-time behavior. The following settings are available in this area.

Table 9-9 ▪ Logs Settings

Setting	Description
Log Settings	To enable run-time logging, select the Enable Logging check box in this setting. Then configure the subsettings in this area as needed.
Install	To have the installer generate a log file during installation, select the check box in this setting, and then optionally change the path for the log file in this setting. The default path for the log file on target systems is: `$USER_INSTALL_DIR$$\$_$PRODUCT_NAME$_installation$\$Logs$/$` For more information, see Configuring Run-Time Logging Preferences.
Add	 ***Note ▪*** *This setting is available if the following conditions are true:* • *The Enable Maintenance Mode support check box in the Advanced view on the Project page is selected.* • *The Add Features check box is selected. This check box is under the Enable Maintenance Mode support check box.* To have the installer generate a log file during the add features maintenance mode, select the check box in this setting, and then optionally change the path for the log file in this setting. The default path for the log file on target systems is: `$USER_INSTALL_DIR$$\$_$PRODUCT_NAME$_installation$\$Logs$/$` For more information, see Configuring Run-Time Logging Preferences.
Repair	 ***Note ▪*** *This setting is available if the following conditions are true:* • *The Enable Maintenance Mode support check box in the Advanced view on the Project page is selected.* • *The Repair Installation check box is selected. This check box is under the Enable Maintenance Mode support check box.* To have the installer generate a log file during the repair maintenance mode, select the check box in this setting, and then optionally change the path for the log file in this setting. The default path for the log file on target systems is: `$USER_INSTALL_DIR$$\$_$PRODUCT_NAME$_installation$\$Logs$/$` For more information, see Configuring Run-Time Logging Preferences.

Table 9-9 ▪ Logs Settings (cont.)

Setting	Description
Remove	 ***Note ▪*** *This setting is available if the following conditions are true:* ● *The Enable Maintenance Mode support check box in the Advanced view on the Project page is selected.* ● *The Remove Features check box is selected. This check box is under the Enable Maintenance Mode support check box.* To have the installer generate a log file during the remove features maintenance mode, select the check box in this setting, and then optionally change the path for the log file in this setting. The default path for the log file on target systems is: `$USER_INSTALL_DIR$$\$_$PRODUCT_NAME$_installation$\$Logs$/$` For more information, see Configuring Run-Time Logging Preferences.
Uninstall	 ***Note ▪*** *This setting is available if the following conditions are true:* ● *The Enable Maintenance Mode support check box in the Advanced view on the Project page is selected.* ● *The Uninstall Product check box is selected. This check box is under the Enable Maintenance Mode support check box.* To have the installer generate a log file during the uninstallation of the product, select the check box in this setting, and then optionally change the path for the log file in this setting. The default path for the log file on target systems is: `$USER_INSTALL_DIR$$\$_$PRODUCT_NAME$_installation$\$Logs$/$` For more information, see Configuring Run-Time Logging Preferences.
Log Format	Specify the format that you want to use for the log files. Available options are: ● Plain text format ● XML format
Uninstall All Logs	Specify whether you want the log files that the installer or uninstaller generated to be removed. Note that the most recent uninstall log is not removed.

Table 9-9 ▪ Logs Settings (cont.)

Setting	Description
Skip Logging in Pre-Install	Select one of the following options to specify whether you want enable or disable the creation of the standard installation log if the user cancels installation during the Pre-install Sequence: ● **Yes**—Do not create the standard installation log. ● **No**—Create the standard installation log.
Include Debug Output (stderr and stdout)	Specify whether you want the stderr and stdout entries to be included in the log files. Available options are: ● **Yes**—Append the debug output to the log files. The stderr and stdout entries are appended to the end of the log files (for text log files) or inserted as <![CDATA[stderr/stdout entries]]> (for XML log files). ● **No**—Prevent the debug output from being included in the log files. If this option is selected, you can optionally specify values in the **Send stderr to** setting and in the **Send stdout to** setting.

Table 9-9 ▪ Logs Settings (cont.)

Setting	Description
Send stderr to	 ***Note ▪*** *This setting is enabled if the following conditions are true:* ● *The Enable Logging check box in the Log Settings setting is selected.* ● *No is selected in the Include Debug Output (stderr and stdout) setting.* To specify the location for stderr output, enter one of the following: ● **Console**—To send stderr output to a console window (or to Console.app on OS or OS X–based target systems), enter **`console`**. ● **File**—To send stderr output to a file, enter the name of 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**—To specify platform-neutral paths for the debug output, use $/$ or $\$, Java properties ($prop.*$), or LAX environment variables ($lax.nl.env.*$ or $lax.nl.env.exact_case.*$). (LAX environment variables are not supported for pure Java-based installers or OS or OS X–based installers.) ● **No output**—To discard stderr output, leave this setting blank. For information on interpreting the output that the installer sends to stderr, see Reviewing Debug Information.  ***Tip ▪*** *The value of the* ***Send stderr to*** *setting maps to the LAX property* `lax.stderr.redirect`. *If your project includes one or more merge modules, the stderr setting of the parent project overrides the stderr setting in the merge modules.*  ***Note ▪*** *The output that is generated for this setting produces different results than the output of the Output Debug Information panel action.*

Table 9-9 ▪ Logs Settings (cont.)

Setting	Description
Send stdout to	 ***Note ▪*** *This setting is enabled if the following conditions are true:* • *The Enable Logging check box in the Log Settings setting is selected.* • *No is selected in the Include Debug Output (stderr and stdout) setting.* To specify the location for stdout output, enter one of the following: • **Console**—To send stdout output to a console window (or to Console.app on OS or OS X–based target systems), enter **`console`**. • **File**—To send stdout output to a file, enter the name of a file. For example, **`installer_stdout.txt`** 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**—To specify platform-neutral paths for the debug output, use $/$ or $\$, Java properties ($prop.*$), or LAX environment variables ($lax.nl.env.*$ or $lax.nl.env.exact_case.*$). (LAX environment variables are not supported for pure Java-based installers or OS or OS X–based installers.) • **No output**—To discard stdout output, leave this setting blank. For information on interpreting the output that the installer sends to stdout, see Reviewing Debug Information.  ***Tip ▪*** *The value of the* ***Send stdout to*** *setting maps to the LAX property lax.stdout.redirect.* *If your project includes one or more merge modules, the stdout setting of the parent project overrides the stdout setting in the merge modules.*  ***Note ▪*** *The output that is generated for this setting produces different results than the output of the Output Debug Information panel action.*
Show 24 hour date format in logs	Specify whether to display logs in 12-hour time format or 24-hour time format. Available options are: **No** (default) - display the logs in 12-hour time format and Month date, YYYY for standard log entries. **Yes** - display the logs in 24-hour time format and YYYY-MM-DD for standard log entries.

Platforms View

Use the Platforms view on the Project page to define default settings that are unique to each target operating system. While InstallAnywhere-generated installers runs on any Java-enabled platform, there are features that should be defined separately for each target operating system.

The settings in the Platforms view are organized into the following main categories:

- Mac OS X
- Windows
- UNIX
- System i (i5/OS)
- Pure Java

Mac OS X

Use the OS X area to specify default locations for install and alias folders, the default Java VM for LaunchAnywhere, whether authentication is required for installation, and permissions for files and folders that are created on the target system. The following settings are available in this area:

Table 9-10 ▪ OS X Settings

Setting	Description
Default Install Folder	Specify the default value for your install location. Select a magic folder in the list and specify a subdirectory as necessary. For more information, see Magic Folders and Variables.
Default Alias Folder	Specify the default value for the alias location. Select a magic folder in the list and specify a subdirectory as necessary. End users can override this location on the Choose Alias Folder panel. For more information, see Magic Folders and Variables.
Select Java VM for LaunchAnywhere	By default, OS X–based installers use the latest valid JVM that is available on the target system. If you want LaunchAnywhere to use an earlier JVM, select the appropriate version. Available options are: ● **11.0***—Require LaunchAnywhere to use any 11.0 JVM. The JVM version must be greater than or equal to 11.0.0_0 but less than 12.0.0_0. ● **11.0+**—Require LaunchAnywhere to use any 11.0 or later JVM. The JVM version must be greater than or equal to 11.0.0_0.  ***Note ▪*** *OS X–based installers do not show Choose Java VM panels; in addition, they do not respond to the settings in the Valid VM list field on the Installer Settings tab (Project > JVM Settings).*

Table 9-10 ▪ OS X Settings (cont.)

Setting	Description
Authentication	Use the following subsettings to identify authorization requirements for your installer: • **Requires an Administrator Name and Password to Install**—Specify whether your installer and its uninstaller require administrative privileges to install files to and remove files from locations where write permissions are restricted for standard users. • **Always Show GUI**—Specify whether you want to ensure that the Authenticate dialog box is shown. It is unnecessary to show the Authenticate dialog box when the end user who runs the installer is logged in as the root user. When an end user successfully authenticates, the installer can write to protected directories and files, such as /usr/bin and /usr/lib. The default value is Yes. Note that if you want to require authentication, you must code sign the authentication wrapper, the helper tool, your installer, and your uninstaller. To learn more, see About Authentication and Code-Signing Support for OS or OS X–Based Installers.
Default Permissions	To configure default read, write, and execute permissions for the files that your installer deploys, click the ellipsis button (...) in this setting. The Permissions dialog box opens, enabling you to specify the permissions for owner, group, and others. This setting shows the value of the specified permissions in octal notation, where the first digit represents the permissions for the owner, the second digit represents the permissions for the group, and the third digit represents the permissions for others.

Table 9-10 ▪ OS X Settings (cont.)

Setting	Description
Code Signing	Specify whether and how you want InstallAnywhere to code sign your OS X–based installer at build time. If you sign the installer, end users can download your installer from outside the App Store and install the product without being blocked by Gatekeeper. The settings in this area are: • **Code Sign the Generated Installer**—To code sign your installer, select this check box and complete the other settings in this area. • **PKCS #12 File**—Specify the fully qualified path for your PKCS #12 file (.p12). The use of a build-time variable is highly recommended for security purposes. If the Code Sign the Generated Installer check box in the Code Signing setting is cleared, this setting is disabled. • **Keystore Password**—Specify the password for the certificate. The use of a build-time variable is highly recommended for security purposes. If the Code Sign the Generated Installer check box in the Code Signing setting is cleared, this setting is disabled. Note that if you want to require authentication, you must code sign the authentication wrapper, the helper tool, your installer, and your uninstaller. To learn more about code signing or authentication, see About Authentication and Code-Signing Support for OS or OS X–Based Installers.

Table 9-10 ▪ OS X Settings (cont.)

Setting	Description
App Notarization	Specify whether and how you want InstallAnywhere to notarize your OS X–based installer at build time. If you notarize the installer, end users can download your installer from outside the App Store and install the product without being blocked by Gatekeeper. The settings in this area are: • **Notarize the Generated Installer** - To notarize your installer, select this check box and complete the other settings in this area. • **Developer Username** - Specify the apple's username for your notarization. If the Notarize the Generated Installer check box in the App Notarization setting is cleared, this setting is disabled. • **Developer App Specific Password** - Specify the App specific password for your application. If the Notarize the Generated Installer check box in the App Notarization setting is cleared, this setting is disabled. • **Team Identifier** - Specify the Team identifier information from the Apple Developer certificate. If the **Notarize the Generated Installer** check box in the **App Notarization** setting is cleared, this setting is disabled. • **Notarization Response Timeout (min)** - Specify the notarization response timeout for your application. The default value is 5 minutes. You can also modify according to the payload. Based on the specified Notarization Response Timeout (min), InstallAnywhere will await for a response of the Notarization status from the Apple server. If the response is not received by the specified time, the installer build fails. If the Notarize the Generated Installer check box in the App Notarization setting is cleared, this setting is disabled. • **Notarization Process Delay Interval (sec)** - Specify the notarization process delay interval for your application. The default value is 2 seconds. You can also modify according to the configuration of system processor. Based on the specified notarization process delay interval (sec), process will wait for the specified time to complete the command execution. If the Notarize the Generated Installer check box in the App Notarization setting is cleared, this setting is disabled. Note that if you want to notarize the application, you must code sign your installer, select the Code Sign the Generated Installer check box and complete the other settings in that area. To learn more about App Notarization, see OS X Notarization.

Windows

Use the Windows area to specify default locations for install and shortcut folders. The following settings are available in this area:

Table 9-11 ▪ Windows Settings

Setting	Description
Default Install Folder	Enter the default value for your install location. Select a magic folder in the list and specify a subdirectory as necessary. For more information, see Magic Folders and Variables.
Default Shortcut Folder	Enter the default value for the shortcut location. Select a magic folder in the list and specify a subdirectory as necessary. End users can override this location on the Choose Shortcut Folder panel. For more information, see Magic Folders and Variables.
Disable All Users Shortcut Icon Creation for Non-Admin Users	Specify whether you want to enable or disable the **Create Icons for All Users** checkbox on the Choose Shortcut Folder panel when the installer is executed with non-administrative privileges. The possible values for this setting are as follows: ● **Yes**—Disables the **Create Icons for All Users** checkbox on the Choose Shortcut Folder panel when the installer is run by a non-administrative user. ● **No**—Enables the **Create Icons for All Users** checkbox on the Choose Shortcut Folder panel when the installer is run by a non-administrative user. By default, this setting to **No**. The **Create Icons for All Users** checkbox typically allows users only with administrative credentials to create shortcut icons in system-wide (All Users) locations such as the shared Desktop or Start Menu. However, if this checkbox is enabled, non-administrative users remain restricted by Windows permission requirements.
Specify the Launcher JVM Settings on a 64-Bit Machine	Indicate which JVM—32 bit or 64 bit—the installer launcher should use on 64-bit Windows-based target systems.

Table 9-11 ▪ Windows Settings (cont.)

Setting	Description
Windows Execution Level	Select the execution level that your installer launcher requires for Windows-based systems. Available options are: • **As Invoker**—The installer launcher is run with the same execution level as its parent process. This execution level is typically standard user privileges, since Windows Explorer runs as a standard user. If the installer launcher does not require administrative privileges and all users can run it without administrative privileges, select this option. • **Highest Available**—The installer launcher is run with the highest Windows privileges and user rights that are available to the current user. Administrators must authorize it; non-administrators run it without administrative privileges. • **Administrator**—The installer launcher requires local administrative 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.
Estimated Size (in MB)	Indicates size specified for installer built, this value appears in ARP entries. Enter an **Estimated Size** for the installer in Mega Bytes. Any whole number between 1 and 4194303 (4 TB) is valid value. By default, the value is blank and you cannot enter the floating/decimal value for an **Estimated Size**.
Apply Installer File Attributes	Specify whether the **File Attributes** should be applied to the installer or not. The possible values are as follows: • Yes - applies **File Attributes** to the Installer. By default, this option is selected. • No - does not apply **File Attributes** to the Installer.
Apply Uninstaller/ Launcher File Attributes	Specify whether the **File Attributes** should be applied to the Uninstaller and Launchers. The possible values are as follows: • **Yes** —Applies **File Attributes** to the Uninstaller and Launchers. By default, this option is selected. • **No** —Does not apply **File Attributes** to the Uninstaller and Launchers.  ***Note ▪*** *This option should be set to* ***No*** *when you perform manual signing of launchers.*

Table 9-11 ▪ Windows Settings (cont.)

Setting	Description
Secure JVM Folder Check	Specify whether you want to enable or disable the check in the current project that verifies whether the JVM is located within the secure or admin folder. This setting allows you to control how InstallAnywhere handles JVM location in a project during the installer or uninstaller launch. The following are the available options: • **Enable**—Enables the check in the current project that verifies whether the JVM is located within the secure or admin folder, and selecting this option overrides the configuration of the `designer.jvm.securefolder.check` property in the `com.zerog.ia.Designer.properties` file. • **Disable**—Disables the check in the current project that verifies whether the JVM is located within the secure or admin folder, and selecting this option overrides the configuration of the `designer.jvm.securefolder.check` property in the `com.zerog.ia.Designer.properties` file. • **Same as Designer Property**—InstallAnywhere enables or disables the JVM secure-folder check in the current project based on the existing configuration of the `designer.jvm.securefolder.check` property in the `com.zerog.ia.Designer.properties` file. If the `designer.jvm.securefolder.check` property is not configured, the JVM secure-folder check is enabled by default for the current project.  ***Note ▪*** *When the check that verifies whether the JVM is located within the secure or admin folder is enabled for the current project, only a JVM located within the secure or admin folder is allowed to launch the installer or uninstaller.* *If this check is disabled, a JVM within the non-admin folder is also permitted to launch the installer or uninstaller, in addition to a JVM within the secure or admin folder.*

Table 9-11 ▪ Windows Settings (cont.)

Setting	Description
Default Windows UI Mode	Use the following subsettings to select the appropriate default UI mode for the installer and the uninstaller: ● **Installer UI Mode**—Select the UI mode for the Windows-based installer. ● **Uninstaller UI Mode**—Select the UI mode for the Windows-based uninstaller. The choices that are available in these settings depend, in part, on the options that you select for the Allowable UI Modes setting (Installer UI page > Look & Feel Settings view > General UI Settings area). If, for example, Silent is not selected as an allowable mode in the Look & Feel Settings view, Silent is not available in the Installer UI Mode or Uninstaller UI Mode settings.  ***Note ▪*** *InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers will not run under those locales* For more information about silent and console installers, see Silent Installers and Console Installers.

Table 9-11 ▪ Windows Settings (cont.)

Setting	Description

Digital Signing	During build time, InstallAnywhere provides support for digitally signing your Windows-based installers (installer .exe file, installer launcher, and uninstaller launcher). Digitally signing your Windows-based installers assures end users that your installers have not been tampered with or altered since release. End users are presented with a digital certificate when they run your installers. If you have not digitally signed an installer, launching your installer on Windows XP SP2 and later results to display an unknown publisher warning to end users.
	
	
	
	Note ▪ *The ability to digitally sign Windows-based installers at build time requires a personal information exchange file (.pfx) type of digital certificate or an external tool that is configured to access the required certificate, depending on the signing method opted. In addition, it requires that you are using InstallAnywhere on a Windows-based system. If you try to build a Windows-based installer on a non-Windows system, InstallAnywhere is unable to sign the resulting installer.*
	
	Use the following subsettings to configure digital signature information—including the digital signature files granted to you by a certification authority—that InstallAnywhere should use to sign your Windows-based installers (installer .exe file, installer launcher, and uninstaller launcher) during build time:
	
	● **Signing Method**—This setting enables you to specify the method to digitally sign your Windows-based installers (installer .exe file, installer launcher, and uninstaller launcher) during build time. The available options for this setting are:
	  ● **Standard**—Select this option to utilize the default InstallAnywhere sign tool to digitally sign your Windows-based installers during build time. Selecting this option enables the **Certificate Information**, **Timestamp Server**, and **Password** settings.
	  ● **Custom**—Select this option to utilize an external sign tool to digitally sign your Windows-based installers during build time. Selecting this option enables the **Custom Sign Tool Location** and **Sign Tool Arguments** settings.
	
	By default, the **Signing Method** setting is set to the **Standard** option.
	
	● **Certificate Information**—Specify the path and file name for the .pfx file that is provided to you by a certification authority. You can use the ellipsis button (...) in this setting to navigate to the file location in the **Certificate Selection** dialog box.
	
	InstallAnywhere uses this file to sign your Windows-based installer during build time. The **Certificate Selection** dialog box enables you to specify either the location of the .pfx file, or information about the certificate store that contains the certificate.
	
	For more information, see Support for Digital Signature in Standard Signing.
	
	● **Timestamp Server**—Specify the URL of the timestamp server—you want InstallAnywhere to use while signing your Windows-based installers during build time. For instance, the following URL can be used:

Table 9-11 ▪ Windows Settings (cont.)

Setting	Description
(Continued)	**http://timestamp.verisign.com/scripts/timstamp.dll** This is the URL for VeriSign's server. If you do not specify any required URL in this setting and specify a file in the **Certificate Information** setting, InstallAnywhere generates a build-time warning to alert that the digital signature for your installer cannot be timestamped. Disabling timestamping may affect how long your digital signature is considered to be valid. For more information, see Digitally Signing Windows-Based Installers. • **Password**—Specify the password for your certificate file. For more information, see Support for Digital Signature in Standard Signing. • **Custom Sign Tool Location**—Specify the location of the external sign tool that you want to use to digitally sign your Windows-based installers during build time. To specify sign tool's location, click the ellipsis button (...). For more information, see Support for Digital Signature in Custom Signing. • **Sign Tool Arguments**—Specify the valid argument for an external sign tool's configuration. For instance, the following argument can be used for an external sign tool's configuration: **sign /a /fd SHA256 /t http://timestamp.digicert.com** For more information, see Support for Digital Signature in Custom Signing. If a Windows-based installer fails to digitally sign in the custom signing approach then an error message/warning with a corresponding error code appears during build. This error message/warning with a corresponding error code displays via an external tool in the build log. For instance, when the Azure SignTool fails to digitally sign a Windows-based installer due to an invalid argument, the following warning appears in the build log: `<Warning>Code signing failed with the following error: Unable to sign, Error Code : 80070057</Warning>`  ***Note ▪*** *As an alternative to specify the actual certificate file, timestamp server, password, external sign tool's location, and argument—you can use build-time variables in these settings (that is, enclose the name of each variable within at symbols:* ***@VariableName@****). You can set build-time variables in the* ***Variables*** *view on the* ***Project*** *page using a .properties file, or as environment variables.* *For more information, see Resolving Variables at Build Time.*

UNIX

The default settings for UNIX include default locations for install and link folders, default user interface mode, permissions for files and folders that are created on the target system, and Red Hat Package Management (RPM) settings for Linux installations. The RPM feature enables the installer to interact with and make entries into the RPM database. The following settings are available in this area:

Table 9-12 ▪ UNIX Settings

Setting	Description
Default Install Folder	Enter the default value for your install location. Select a magic folder in the list and specify a subdirectory as necessary. For more information, see Magic Folders and Variables.
Default Link Folder	Enter the default value for the link location. Select a magic folder in the list and specify a subdirectory as necessary. The default location is the end user's home directory. End users can override this location on the Choose Link Folder panel. For more information, see Magic Folders and Variables.
Default Unix Installer UI Mode	Use the following subsettings to select the appropriate default UI mode for the installer and the uninstaller: ● **Installer UI Mode**—Select the UI mode for the UNIX-based installer. ● **Uninstaller UI Mode**—Select the UI mode for the UNIX-based uninstaller. The choices that are available in these settings depend, in part, on the options that you select for the Allowable UI Modes setting (Installer UI page > Look & Feel Settings view > General UI Settings area). If, for example, Silent is not selected as an allowable mode in the Look & Feel Settings view, Silent is not available in the Installer UI Mode or Uninstaller UI Mode settings.  ***Note*** ▪ *InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers will not run under those locales* For more information about silent and console installers, see Silent Installers and Console Installers.
Default Permissions	To configure default read, write, and execute permissions for the files that your installer deploys, click the ellipsis button (...) in this setting. The Permissions dialog box opens, enabling you to specify the permissions for owner, group, and others. This setting shows the value of the specified permissions in octal notation, where the first digit represents the permissions for the owner, the second digit represents the permissions for the group, and the third digit represents the permissions for others.

Table 9-12 ▪ UNIX Settings (cont.)

Setting	Description
Enable RPM Registration (Linux Only)	Specify whether you want to enable Red Hat Package Management (RPM) registration. On supported Linux systems, RPM registration creates a virtual package and uses it to make entries to the RPM database. If you select Yes in this setting, the ellipsis button (...) in this setting is enabled. Click this ellipsis button and configure additional settings. For detailed information, see RPM Specification Settings Dialog Box.
Enable SWVPD Registry Integration (AIX Only)	Specify whether you want 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. If you select Yes in this setting, the ellipsis button (...) in this setting is enabled. Click this ellipsis button and configure additional settings. For detailed information, see SWVPD Registry Settings Dialog Box.  ***Note ▪*** *If you enable SWVPD registry integration and leave one or more settings on the SWVPD Registry Settings dialog box blank, the installer uses corresponding values from the General Settings view on the Project page.*
Restrict Installation to Build Target Platform	Specify whether you want to restrict the installer from running on a system platform different from the one it was originally built for. When this setting is set to **Yes**, the generated installer is allowed to run only on the specific target platform for which it was built. If attempted to execute or run on a different platform, even if compatible, the installer will display an appropriate error message and exit the installation. By default, this setting is set to **No**.
Require Root	Specify whether the installer requires root account permissions. A root account is the user name or account that by default has access to all commands and files on a Linux or other Unix-like operating system. It is also referred to as the root account, root user, or the superuser. If you select **Yes** in this setting, the **Error Message** setting under **Require Root** is enabled.
Error Message	Specify an error message to appear if the installer is run with a non-root account. This field is enabled when **Yes** is selected from the **Require Root** drop-down list.

System i (i5/OS)

The System i (i5/OS) area contains a setting for installers that target the i5/OS operating system running on System i. To build installers for i5/OS, you must have access to an i5/OS system. The following settings are available in this area:

Table 9-13 ▪ System i (i5/OS) Settings

Setting	Description
Enable (RAIR) System i (i5/OS) Registration	Specify whether you want the installer to enter product information in the i5/OS Registered Application Information Repository (RAIR). If you select Yes, the installer adds information about the product's features to the RAIR when the product is installed.  ***Note*** ▪ *Some systems management products, such as Management Central, use RAIR to determine what software is installed on an i5/OS system.*

Pure Java

The Pure Java area contains settings that determine the UI mode that pure Java installers and uninstallers use. The following settings are available in this area:

Table 9-14 ▪ Pure Java Settings

Setting	Description
Follow Platform-Specific Default UI Mode	Specify whether pure Java installers use the project's default UI mode. The default value is Yes. If you select No, use the Installer UI Mode setting and the Uninstaller UI Mode setting to specify the default UI modes for pure Java installers and uninstallers.

Table 9-14 ▪ Pure Java Settings (cont.)

Setting	Description
Installer UI Mode	Select the UI mode for pure Java–based installers. Available options are: • **GUI**—GUI mode renders an installer with wizard panels and dialog boxes. • **Console**—Also known as command-line interface, console-mode installers can perform remote installations over telnet or on systems without a graphical windowing environment. • **Silent**—Silent installers do not interact with end users. They are suitable for distribution when all of the settings are already known, or they are provided in a response file. The choices that are available in this setting depend, in part, on the options that you select for the Allowable UI Modes setting (Installer UI page > Look & Feel Settings view > General UI Settings area). If, for example, Silent is not selected as an allowable mode in the Look & Feel Settings view, Silent is not listed in the Installer UI Mode setting. Note that this setting is disabled if Yes is selected for the Follow Platform-Specific Default UI Mode setting.  ***Note ▪*** *InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console-mode installers will not run under those locales.* For more information about silent and console installers, see Silent Installers and Console Installers.

Table 9-14 ▪ Pure Java Settings (cont.)

Setting	Description
Uninstaller UI Mode	Select the UI mode for pure Java–based uninstallers. Available options are: • **GUI**—GUI mode renders an uninstaller with wizard panels and dialog boxes. • **Console**—Also known as command-line interface, console-mode uninstallers can perform remote uninstallations over telnet or on systems without a graphical windowing environment. • **Silent**—Silent uninstallers do not interact with end users. They are suitable for distribution when all of the settings are already known, or they are provided in a response file. • **Same as Installer**—Use the same UI mode that is configured for the pure Java–based installer. The choices that are available in this setting depend, in part, on the options that you select for the Allowable UI Modes setting (Installer UI page > Look & Feel Settings view > General UI Settings area). If, for example, Silent is not selected as an allowable mode in the Look & Feel Settings view, Silent is not listed in the Installer UI Mode setting. Note that this setting is disabled if Yes is selected for the Follow Platform-Specific Default UI Mode setting. ***Note ▪*** *InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers will not run under those locales.* For more information about silent and console installers, see Silent Installers and Console Installers.

Locales View

Use the Locales view on the Project page to configure project-level language settings.

Tip ▪ *To specify the locales that your project supports, use the Locales subtab. To access the Locales subtab: On the Build page, click Build Installers, click the Build Configurations tab, and then click the Locales subtab. For more information, see Locales Subtab.*

The following settings are available in the Locales view:

Table 9-15 ▪ Locales Settings

Setting	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	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). If the **Preserve layout of the locale files** check box is selected, InstallAnywhere retains all existing comments in the project's locale files.  ***Note ▪*** *Even if this check box is selected, the comments that are immediately preceding each key (containing the English version of the translated text) are still overwritten.*
Remove unused keys from the locale files	Deletes the keys that are 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. Selecting this check box removes the unused keys that InstallAnywhere automatically generates for a panel or console each time that you save the project. This option also removes the comments that are immediately preceding each key as well as the line break following each key-value pair. Keys that are associated with custom code are always preserved.
External Resource Bundle Settings	Shows any external resource bundles that are included in the project: ● **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.  ***Note ▪*** *For more information about adding external resource bundles to the Locales view on the Project page, see Adding an External Resource Bundle.*

Manage Expressions View

Use the Manage Expressions view on the Project page to:

- Create complex rule expressions at the project level

- Combine multiple rules and save them to a single rule expression
- Reuse a rule or a set of rules across the project
- Associate a rule expression to a file extension, including the option to automatically associate the rule expression to a file whenever a file containing that file extension is added to the project
- Remove associated rules from custom file extensions

The following settings are available in this view:

Table 9-16 ▪ Manage Expressions View Settings

Setting	Description
Expression Name	When you click New Expression to add a new rule expression to your installation project, the unique rule ID of each rule is listed in this field. You can enter a new name that is more descriptive of what rule expression you are saving.  ***Note*** ▪ *For more information, see Using the Rules Manager to Create Complex Rule Expressions at the Project Level.*

Table 9-16 ▪ Manage Expressions View Settings (cont.) (cont.)

Setting	Description
Rule Expression	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. If you want to write complex rule expressions, you can click Add Rule to add additional rules to save to the expression. You can also edit the expression in this field to use multiple operators to express the relationship between two or more rules, such as: ● **&&** (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. You can also use precedence operators, such as parentheses—**( )**, to compose rule expressions.  ***Note ▪*** *For more information, see Using the Rules Manager to Create Complex Rule Expressions at the Project Level as well as Building Complex Rule Expressions.*  ***Note ▪*** *You can also write complex rule expressions in the Rules customizers on individual actions, panels or consoles.*
Rules List	Shows all rules currently defined for your project.  ***Note ▪*** *New projects have no rules set; thus, the Rules List is initially empty.*
New Expression	Adds a new rule expression at the project level.
Delete Expression	Deletes the currently selected expression from the project.

Table 9-16 ▪ Manage Expressions View Settings (cont.) (cont.)

Setting	Description
Add Rule	Opens the **Choose a Rule/Expression** dialog box. Available options are: • Check File/Folder Attributes • Check Platform • Check System Architecture • Check User-Chosen Language • Compare InstallAnywhere Variables • Compare InstallAnywhere Variables Numerically • Compare Versions • 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 When you choose a rule, InstallAnywhere shows the controls for providing settings specific to that rule type in the area below the Rules table.
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.
Associate Expression to File Expression	This area lets you associate a rule expression to a file extension.  ***Note ▪*** *The section assumes that the rule expression you will be associating to a file extension has already been saved. If you do not have a rule expression saved, refer to Configuring and Saving a New Rule Expression for information about how to create a rule expression.*
Add	Click to associate a rule expression to a file extension. When you click Add a Choose an Expression dialog appears. Choose the expression that you want to associate to a file and then click Load.

Table 9-16 ▪ Manage Expressions View Settings (cont.) (cont.)

Setting	Description
Remove	Click to remove the currently selected expression from the Associate Expression to File Expression area. Any files with the specified file extension will remove the associated rule expression from them that yo have removed.
Apply	By default, the **Apply** check box is checked. This means that any files in the project that contain the file extension you have specified will be updated to have the rule expression applied to them. Conversely, if you uncheck the **Apply** check box and then click the **Apply** button, any files with the specified file extension will remove the rule expression from them.
Apply automatically when new files are added	Click to automatically associate the rule expression to a file whenever a file containing that file extension is added to the project on the Sequence page.

Installer Rules View

Use the Installer Rules view on the Project page to add logic that executes prior to items in the pre-install sequence. Use this option to check if the target system is the right platform for this installation, if the end user is logged in to the root, or if the end user has the necessary permissions to perform the installation.

The following settings are available in this view:

Table 9-17 ▪ Installer Rules View Settings

Setting	Description
Rule Expression	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. 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: ● `&&` (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. You can also use precedence operators, such as parentheses—`( )`, to compose rule expressions.  ***Note*** ▪ *For more information, see Building Complex Rule Expressions.*  ***Note*** ▪ *You can also write complex rule expressions in the Rules customizers on individual actions, panels or consoles.*
Rules List	Shows all rules currently defined for your project.  ***Note*** ▪ *New projects have no rules set; thus, the Rules List is initially empty.*

Table 9-17 ▪ Installer Rules View Settings (cont.) (cont.)

Setting	Description
Add Rule	Opens the Choose a rule dialog box. Available options are: ● Check File/Folder Attributes ● Check Platform ● Check System Architecture ● Check User-Chosen Language ● Compare InstallAnywhere Variables ● Compare InstallAnywhere Variables Numerically ● Compare Versions ● 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 When you choose a rule, InstallAnywhere shows the controls for providing settings specific to that rule type in the area below the Rules table.
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 View

Use the JVM Settings view on the Project page to define JVM search settings. The JVM Settings view contains the following tabs:

- General Settings Tab in the JVM Settings View
- Installer Settings Tab in the JVM Settings View

- Search Panel Settings Tab in the JVM Settings View

General Settings Tab in the JVM Settings View

On the General Settings tab of the JVM Settings view, you can modify the classpath settings for your project's LaunchAnywhere applications.

Use the following controls to modify the classpath settings for your project's LaunchAnywhere applications:

Table 9-18 ▪ General Settings Tab

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.
Restrict Non-platform files	Allows only the platform-specific service support files to be built into the installer and uninstaller archives for each target platform when support for the custom code services layer is included.  ***Note ▪*** *Selecting the* ***Restrict Non-platform files*** *check box additionally restricts the contents of a HP-UX file named* `ismpHpUxNativeLib.jar`*, which is located in the* `<IA_HOME>/resource/nativetools/hpux` *directory, from being built into the installer and uninstaller archives for non HP-UX platforms.*

Installer Settings Tab in the JVM Settings View

Use the Installer Settings tab of the JVM Settings view to define a list of valid Java VMs that 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 that the installer already sets), indicate whether to install the bundled/downloaded Java VM, and define classpath settings.

The settings on the Installer Settings tab are organized into the following areas:

- Installer Valid VM List
- Installer VM Heap Size
- Optional Installer Arguments
- Bundled/Downloaded Virtual Machine
- Additional Classpath Settings

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 and version. For example, `JDK_11.0+` selects any JDK of version 11.0.0_0 or later.

For details about valid entries for this setting, see About Java VM Selection Criteria.



Tip ▪ *These settings have no effect on OS or OS X–based installers. By default, OS or OS X–based installers use the most recent, valid VM that is available on the target system. To force the LaunchAnywhere to use an earlier VM for OS or OS X–based installers, use the Select Java VM for LaunchAnywhere setting in the OS X area in the Platform Properties view of the Project page.*

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 9-19 ▪ 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 9-20 ▪ Bundled/Downloaded Virtual Machine Controls

Control	Description
Install Bundled/ Downloaded Virtual Machine	Determines whether the installer deploys a bundled VM for use by the application you are installing. 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. Set the conditions under which your installer deploys the bundled/downloaded VM. Available options are: ● **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	Choose the install location (or parent directory) for the bundled VM from a list of Magic Folders. By default, the installer stores the bundled VM in a JRE subdirectory of the Magic Folder you chose.

Table 9-20 ▪ Bundled/Downloaded Virtual Machine Controls (cont.)

Control	Description
Advanced JRE handling for Version Upgrades	If you are creating an upgrade installer that contains a JRE with a different processor type or version than the previous version of the installer, problems could occur during installation of that JRE. To prevent those problems, select this option. The following are scenarios when you should select this option: ● **New installer has a JRE with a different processor type than previous**—For example, you are currently building App A V2 with a 64-bit JRE version X, but App A V1 had a 32-bit JRE version X. ● **New installer has a JRE with a different version than previous**—For example, you are currently building App A V2 with a 64/32-bit JRE version X, but App A V1 had a 64/32-bit JRE version Y. Selecting this option will enable Maintenance Mode and Instance Management Panels to display the existing instances of the same product across both 32-bits and 64-bits. If this option is not selected, the panels will display only instances which correspond to the processor type of the upgrade installer.  ***Note ▪*** *For more information, see Configuring Upgrade Installers that Use a Different JRE from the Previous Installer.*

The following are scenarios when you should select this option:

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 9-21 ▪ 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.

Table 9-21 ▪ Additional Classpath Settings (cont.)

Control	Description
Remove	Click to remove the selected classpath from the list.

Search Panel Settings Tab in the JVM Settings View

Use the Search Panel Settings tab in the JVM Settings view to specify criteria to use to find valid VMs for the application that you are installing. The Search Panel Settings tab contains the following subtabs:

- General Subtab
- Windows Subtab
- UNIX Subtab

General Subtab

The VM Search Settings area of the General subtab is where you indicate the criteria that the installer uses to find valid VMs for the software that your project installs.

The General subtab includes the following controls in the VM Search Settings area:

Table 9-22 ▪ General Subtab Controls

Control	Description
Use installer's valid VM list	Uses the criteria specified in the Installer Valid VM List (Project > JVM Settings > Installer Settings).
Use the valid VM list of all LaunchAnywhere actions	Uses the `lax.nl.valid.vm.list` value for your application's launcher as the criteria for the VM search.  ***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.*
Use a specific valid VM list	Specify the VM criteria you want the installer to use.  ***Note ▪*** *See About Java VM Selection Criteria for valid VM criteria.*

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.

The Search Paths tab includes the following controls:

Table 9-23 ▪ Search Paths Tab Controls

Control	Description
Add	Click to insert a new entry in the Search Paths list. In the first column of the new entry, enter a path to search for Java virtual machines. In the second column, select one of the following to specify how deep the installer searches subdirectories of the given path: ● **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.
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.

The Java Executable Patterns tab includes the following controls:

Table 9-24 ▪ 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 that the installer uses 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-based 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.

The Search Paths tab includes the following controls:

Table 9-25 ▪ Search Path Tab Controls

Control	Description
Add	Inserts a new entry in the Search Paths list. Type a path to search for Java virtual machines. The following options are available for specifying how deep the installer searches subdirectories of the given path: • **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.
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.

The Java Executable Patterns tab includes the following controls:

Table 9-26 ▪ 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 View

Use the Variables view on the Project page to advertise variables for merge modules, configure build time variables, list variables to encrypt in or exclude from the response files, set security options, and view a list of all defined variables.

The settings in the Variables view are organized into the following areas:

- Advertise Variables
- Build Time Variables
- Configure Variables
- Security
- Variables

Advertise Variables

In the Advertise Variables area, you advertise variables for merge modules.

Table 9-27 ▪ Advertise Variables Controls

Control	Description
Advertise Variables	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. 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.  ***Note*** ▪ *For more information, see Advertise Variables Dialog Box.*

Build Time Variables

In the Build Time Variables area, you can configure your installer to use *build time variables*, which are variables that have their values set at build time.

Table 9-28 ▪ Build Time Variables Controls

Control	Description
Build Time Variables	Click to open the Build Time Variables dialog box, which lists defined build time variables. Click the Edit Variables button on the Build Time Variables dialog box to edit the list of variables. For information on configuring build time variables, see Build Time Variables Dialog Box and Edit Build Time Variables Dialog Box.  ***Note ▪*** *When defining a build time variable, instead of using the $ symbol as the variable delimiter (which is used for standard variables, such as* `$var$`*), you instead use the @ symbol as the delimiter for build time variables, such as* `@var@` *or* `@Apple@`*.*  ***Note ▪*** *In addition to defining build time variables using the Advanced Designer interface, you can also configure build time variables using a property file or as environmental variables. For more information, see Resolving Variables at Build Time.*

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 9-29 ▪ Configure Variables Controls

Control	Description
Configure	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: • Exclude Variable Entirely • Exclude Value Only • Encrypt Variable Value  ***Note ▪*** *For more information on excluding variables, see Generating Response Files.*  ***Note ▪*** *For more information on encrypting variable values, see Encrypting Variable Values.*

Security

Use the following controls to customize security settings for encryption.

Table 9-30 ▪ 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 JRE 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 Variables view provides a scrollable list of all variables defined in the installation project.

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.

Advanced View

Use the Advanced view on the Project page to indicate whether you want to enable maintenance mode support, specify the number of instances of the product per machine, manage build configuration tags, and indicate whether you want to enable product rollback.

The settings on the Advanced tab are organized into the following main categories:

- Maintenance Mode Settings
- Windows WOW64 Emulator Settings
- Instance Management Settings
- Manage Tags Settings
- Rollback Settings

Maintenance Mode Settings

You can choose to support 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 in the Advanced view on the Project page.

When maintenance mode is enabled, end users can launch maintenance mode through the following methods:

- Execute the Change Installation launcher.
- On Windows-based target systems: In the Windows Control Panel, use Programs or Features (formerly called Add or Remove Programs).

When end users launch maintenance mode, they are prompted to select from the listed options, which may include Add Features, Remove Features, Repair Product, and Uninstall Product.

The following settings are available in the Maintenance Mode area of the Advanced tab:

Table 9-31 ▪ Maintenance Mode Settings

Setting	Description
Enable Maintenance Mode Support	To enable maintenance mode support, select this check box. When you select this maintenance mode, the four maintenance check boxes become available to be selected or cleared. Select the check boxes for the support that you would like to include in this installation project. By default, this check box is cleared.
Add Features	To enable the end user to use maintenance mode to install previously uninstalled product features, select this check box. When the end user launches maintenance mode and chooses Add Features, the installer advances to the Pre-Install sequence of the installation and then proceeds as a regular installation. For more information, see Run-Time Behavior for Maintenance Mode.  ***Note ▪*** *The actions that are 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. To learn more, see About Check Running Mode Rules.*

Table 9-31 ▪ Maintenance Mode Settings (cont.)

Setting	Description
Remove Features	To enable the end user to use maintenance mode to remove previously installed product features, select this check box. ***Tip ▪*** *The InstallAnywhere Uninstaller also provides you with the ability to selectively remove individual features.* When the end user launches maintenance mode and chooses Remove Features, the installer advances to the Pre-Uninstall phase of the uninstallation and then proceeds with the uninstallation of the selected features. For more information, see Run-Time Behavior for Maintenance Mode. ***Note ▪*** *The actions that are 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. To learn more, see About Check Running Mode Rules.*
Repair Installation	To enable the end user to use maintenance mode to repair a previously installed product, select this check box. For more information, see Run-Time Behavior for Maintenance Mode. ***Note ▪*** *The actions that are executed when the end user selects Repair Product do not include the actions in the Pre-Install sequence with a Check Running Mode rule that is set to Repair Installation. To learn more, see About Check Running Mode Rules.*
Uninstall Product	To enable the end user to use maintenance mode to uninstall a previously installed product, select this check box. ***Tip ▪*** *The InstallAnywhere Uninstaller also provides you with the ability to uninstall a product.* When the end user launches maintenance mode and chooses Uninstall Product, the installer advances to the Pre-Uninstall phase of the uninstallation and proceeds with the uninstallation of the product. For more information, see Run-Time Behavior for Maintenance Mode.

Windows WOW64 Emulator Settings

The Windows WOW64 Emulator Settings area has the following settings:

Setting	Description
Enable WOW64 Redirection for	To enable Windows WOW64 emulation for one or more of the following elements, select the appropriate check boxes. • **Install**—To enable WOW64 redirection for the Install sequence, select this check box. • **Uninstall**—To enable WOW64 redirection for the Uninstall sequence, select this check box. • **InstallAnywhere Win32 Services**—To enable WOW64 redirection for your custom code, regardless of whether your custom code is placed in the Install sequence or the Uninstall sequence, select this check box.

Instance Management Settings

The Instance Management area in the Advanced View contains the following settings:

Table 9-32 ▪ Instance Management Settings

Setting	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.  ***Note ▪*** *This check box is enabled if the* ***Enable Maintenance Mode support*** *check box is selected.*
Do not Limit Instances	Select this option to enable end users to install an unlimited number of instances of this product on a machine. 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). If an installer was created with the Do not limit instances option selected, the following occurs: • 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. For more information, see Run-Time Behavior for Maintenance Mode.

Table 9-32 ▪ Instance Management Settings (cont.)

Setting	Description
Restrict to a Single Instance	Select this option to restrict end users to be able to install only one instance of the product on a machine. By selecting the Restrict to a Single Instance option, you are preventing the end user from being able to do either of the following: ● Installing additional instances of the product on the same machine. ● Overwriting an existing instance of the product. 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.
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.
Identify Instance Using	Select one of the following options to specify how to identify an instance of an application: ● **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.

Table 9-32 ▪ Instance Management Settings (cont.)

Setting	Description
Version Field Level That Identifies New Instance	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: • Major • Minor • Revision • Subrevision  ***Note ▪*** *Versions are conventionally represented in the following format: [Major].[Minor].[Revision].[Subrevision], such as:* `1.0.2.1047`*.* For more information, see Specifying Instance Management Behavior.  ***Note ▪*** *Reinstalling an application to the same location as an existing application would result in only one instance, irrespective of the version.*  ***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.*
Enable Instance Overwrite Warning	To warn end users when they are about to overwrite an existing instance of an application and specify whether they will be permitted to overwrite the instance, select this check box. After you select this check box, select one of the following options: • **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.  ***Note ▪*** *If this check box is selected, you are required to include a Panel: Choose Install Folder action in the Pre-Install sequence.*

Manage Tags Settings

Use the Manage Tags area in the Advanced view of the Project page to manage project tags.

The following settings are available in the Manage Tags area:

Table 9-33 ▪ Manage Tags Setting

Setting	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 rollback support through the Rollback Settings area in the Advanced view of the Project page.

The following settings are available in the Rollback Settings area:

Table 9-34 ▪ Rollback Settings

Setting	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.

Table 9-34 ▪ Rollback Settings (cont.)

Setting	Description
Restart Windows in case of Rollback	Select this check box to instruct a target Windows machine to restart after a rollback occurs during installation. If this check box is selected, the machine will restart once the user clicks the Done button in the Install Complete panel. ***Note ▪*** *If this* ***Restart Windows in case of Rollback*** *check box is selected, the target Windows machine restarts regardless of whether a Restart Windows action is present in the Post-Install sequence, with one exception. If a Restart Windows action is present and you assign to it a rule that would prompt the end user to choose between* ***Yes, restart*** *or* ***No, I will restart later****, if the end user selects* ***No****, the machine will not restart even if the* ***Restart Windows in case of Rollback*** *check box is selected.* *The* ***Restart Windows in case of Rollback*** *check box is enabled if the Enable Rollback check box is selected.*

Upgrades View

Use the Upgrades view on the Project page to configure settings for creation of an upgrade installer that uninstalls earlier versions of your product—if they are present—before installing the new version.

The settings in the Upgrades view are organized into the following areas:

- Upgrades
- Upgrade Configuration
- Upgrade Behavior

Upgrades

Use the Upgrades area in the Upgrades view to specify whether you want to include upgrade support in your installers. This area also shows the upgrade code of the project.

Table 9-35 ▪ Upgrades Settings

Control	Description
Upgrade Code	Enter a GUID that uniquely identifies a product family. In order to include upgrade support, the upgrade code should be consistent across different versions and languages of a family of related products. To have InstallAnywhere generate a different GUID for you, click the Generate Upgrade Code button ({...}) in this setting. The GUID must be in the format that is displayed. At run time, the installer writes the upgrade code in the InstallAnywhere registry, regardless of whether upgrade support is enabled.

Table 9-35 ▪ Upgrades Settings (cont.)

Control	Description
Enable Upgrade Support	Specify whether you want to create installers that have support for upgrading earlier versions of your product that are present on target systems before installing the new version. If upgrade support is not enabled, the installer does not check target systems at run time to verify whether an earlier version of the product needs to be uninstalled before installing the new version. The default value is No. For background information on requirements for including upgrade support, see Requirements for Upgrade Support.

Upgrade Configuration

Use the Upgrade Configuration area in the Upgrades view to define information about the earlier versions of your product that you want to be able to upgrade with installers that are built out of the current project. When you click the Manage Upgrade Configurations button in this area, the Manage Upgrade Configurations dialog box opens. To learn more, see Manage Upgrade Configurations Dialog Box.

Upgrade Behavior

Use the Upgrade Behaviors area in the Upgrades view to configure details about the upgrade that you are creating in the current project.

Table 9-36 ▪ Upgrade Behavior Settings

Control	Description
Abort Installation of New Version If Removal of Earlier Version Fails	Indicate whether you want to avoid installing the new version of the product if an earlier version of the product was detected on the target system but it could not be successfully removed. Available options are: ● **Yes**—Skip the installation of the new version on the target system. ● **No**—Install the new version on the target system.

Table 9-36 ▪ Upgrade Behavior Settings (cont.)

Control	Description
If Multiple Earlier Versions Meet the Upgrade Configuration Conditions	Select the behavior that you want to occur if multiple earlier versions of the product are detected to be present on the target system—that is, if multiple earlier versions of the product meet the upgrade configuration conditions that are configured in the Upgrade Configuration area in the Upgrades view. Available options are: ● **Automatically Remove All of Them and Install the New Version**—All of the detected earlier versions are silently removed from the target system, and then a single instance of the new version is installed. ● **Prompt the End User to Select Which One to Remove**—At run time, enable end users to either select one of the existing instances that they want to upgrade or optionally cancel the upgrade. If the installer is running in GUI mode, the installer displays a dialog box to enable end users to select the appropriate option. If the installer is running in console mode, the Command Prompt window displays equivalent prompts to enable end users to select the appropriate option. If the end user selects an existing instance, that instance is silently removed from the target system, and then a single instance of the new version is installed. If the installer is running silently, the first instance that the installer finds in the InstallAnywhere registry is removed.
Allow User Install Directory to Be Customized at Upgrade	Indicate whether you want end users to be able to specify a new location for the new version that the upgrade is installing. Available options are: ● **Yes**—The upgrade installer displays the Chose Install Folder panel, or the equivalent console, to enable the end user to choose where to install the new version that the upgrade is installing. ● **No**—The upgrade installer installs the new version in the same location where the earlier version was detected on the target system.

Table 9-36 ▪ Upgrade Behavior Settings (cont.)

Control	Description

Retain Feature Preferences During Upgrade	Indicate whether you want end users' feature preferences for the earlier version of the product to be retained for the new version that the upgrade is installing. Available options are:
	• **Yes**—The resulting behavior depends on the display option that you select for the Choose Install Sets panel or console action:
	  • **Choose Install Sets [only]**—The install sets that were selected for the earlier version of the product are selected by default in the upgrade.
	  • **Choose Product Features [only]**—The features that were selected for the earlier version of the product are selected by default in the upgrade.
	  • **Choose Install Sets [followed by] Choose Product Features**—The install sets and features that were selected for the earlier version are selected by default in the upgrade.
	  The Yes option may be useful if the feature tree does not change significantly between the earlier version and the new version.
	• **No**—The upgrade installs the default install set and its features.

Installer Updates

In InstallAnywhere 2018 SP1, you can now enable and configure a software downloadable update. The end user can either download a newer version or skip and proceed with the current installation process. A new Installer Updates option has been added to the Upgrades menu where you can specify these downloadable update settings.

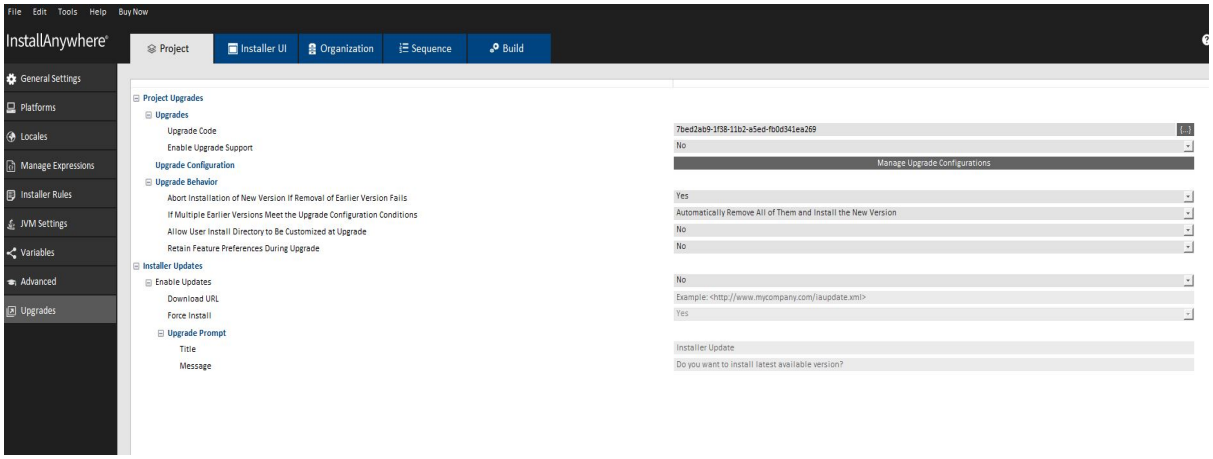
Requirements and Recommendations

The requirements and recommendations for supporting the Downloadable Updates are given below:

1. The URL for hosting a metadata file should be called iaupdate.xml file.
2. The URL for hosting the update installer setup, should be same as iaupdate.xml location.
3. During the build time, InstallAnywhere creates iaupdate.xml file and places it under the project's build folder inside UpdateMetadata subfolder. The iaupdate.xml file identifies the different versions of installer and their downloaded locations. You can update this file manually as required before uploading to the website.
4. While releasing an update, upload the iaupdate.xml file and update the installer setup to your website.
5. The iaupdate.xml file contains the URL for updated installer setup, which by default is same as iaupdate.xml host URL and can be changed as required.

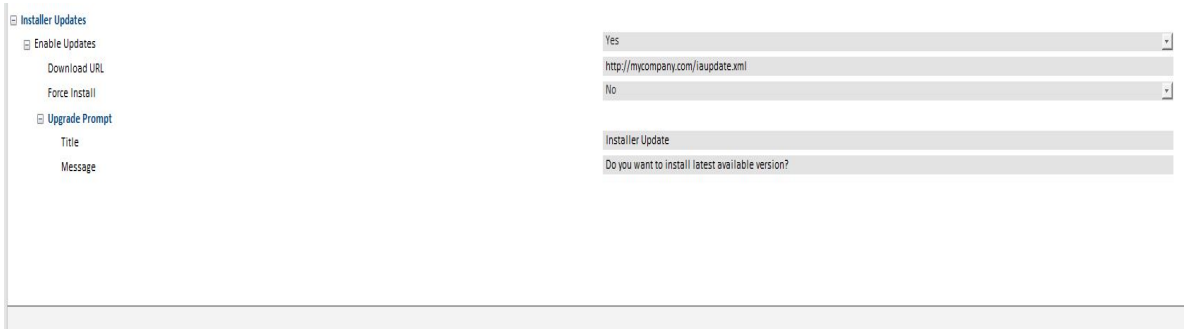


Note ▪ *GUI and Console mode of installation are support Installer Update. Silent mode will proceed with the current installation.*



Control	Description
Enable Updates	You can now enable updates for the end users to see the downloadable update support. To enable updates, select the Enable Updates option on the Installer Updates tab of the Upgrades menu. ***To Enable Downloadable Update Support for a Base Project:*** 1. Click and open InstallAnywhere project. 2. In the **Project** tab, click **Upgrades** view. 3. In the **Installer Updates** option > **Enable Updates**, select the appropriate value: ● **Yes**—Enable the downloadable installer updates. ● **No**—Disable the downloadable installer updates.
Download Url	You can now enter the download URL (iaupdate.xml) in the Download URL field on the Enable Updates option on the Installer Updates tab of the Upgrades menu. Yo can enter the absolute path URL (starting with either http:// or https:/ or ftp://) on the Installer Updates option in the Download URL. The iaupdate.xml file needs to be uploaded to the same path that you specify in the Download URL setting.

Control	Description
Force Install	***To Force Install:*** 1. Click and open InstallAnywhere project. 2. In the **Project** tab, click **Upgrades** view. 3. In the **Installer Updates** option > **Force Install**, select the appropriate value: ● **Yes**—The end users can forcibly download and install the new version of the setup when available. ● **No**—The end users receive an option to either download a newer version or skip and proceed with the current installation process. ***Note ▪*** *You can override the location of the setup launcher that you specify in the Download URL setting by specifying the appropriate path in the iaaupdate.xml file.*
Prompt Message	You can now customize the message and provide a title for the message that is displayed during the installation to prompt the end users. ***Note ▪*** *This setting is enabled only when the Force Install is set as 'No'.* The default value for the Title is 'Installer Update' and the Message is displayed as 'Do you want to install latest available version?' During the installation, if the update is available; The end users will see a prompt dialog to download a newer version. The user can select 'Yes' and proceed with updated installation or select 'No' to skip and proceed with the current installation.



Control	Description
Title	You can now specify the title of the Upgrade Prompt message for the end users.

Control	Description
Message	You can enter the customized message in the Message field on the Upgrade Prompt option on the Installer Updates tab of the Upgrades menu. You can either create a new message or select a message from the localized available list of strings. ***Note ▪*** *If the Force Install is set as 'Yes', the updates are automatically downloaded and no prompt will be available.*

Task ***To Prepare Setup Launcher:***

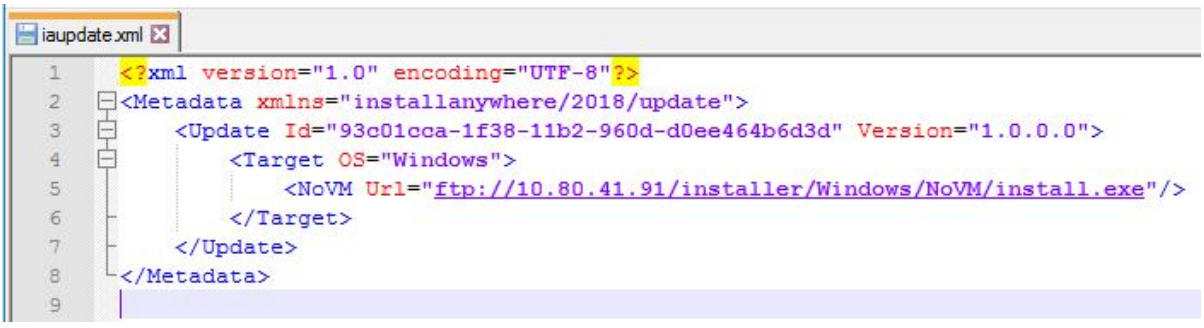
To prepare an update setup launcher for a project:

1. Click and open InstallAnywhere project.
2. Configure Installer Updates setting.
3. Download URL field is mandatory and should point to iaupdate.xml - http://MyWebSite.com/path/iaupdate.xml.
4. Configure Force Install and Upgrade Prompt option as necessary or default values will be set
5. Build the project.

InstallAnywhere creates the iaupdate.xml file and places it under project's build folder inside UpdateMetadata subfolder

The XML file contains elements as following:

- Update Id: Id is same as Product ID which helps base installer to identify corresponding latest version
- Version: Identifies latest version
- Target OS: Identifies latest update is available for specific Operating System
- URL: Installer update latest version location



Example for iaupdate.xml content.

You can edit the file as required before uploading to your site.

For example, if you want to change the path of the updated installer, you can change the value of the URL attribute and then re-upload the iaupdate.xml file to the designated location.

If you are interested in using the same iaupdate.xml file to define the upgrades for multiple products, you can add multiple

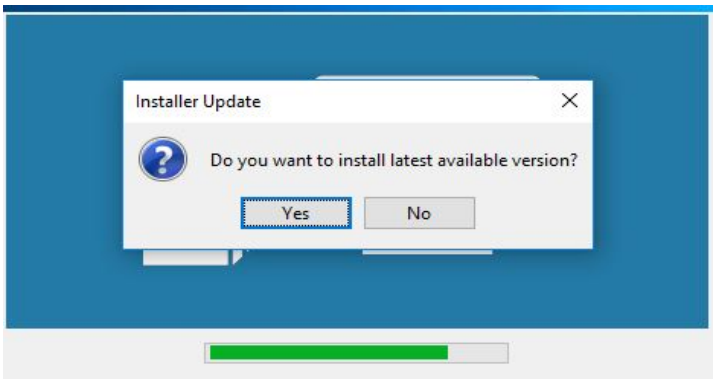
Update element for each of your products to the file.

6. Upload the iaupdate.xml file and the update installer setup to site.

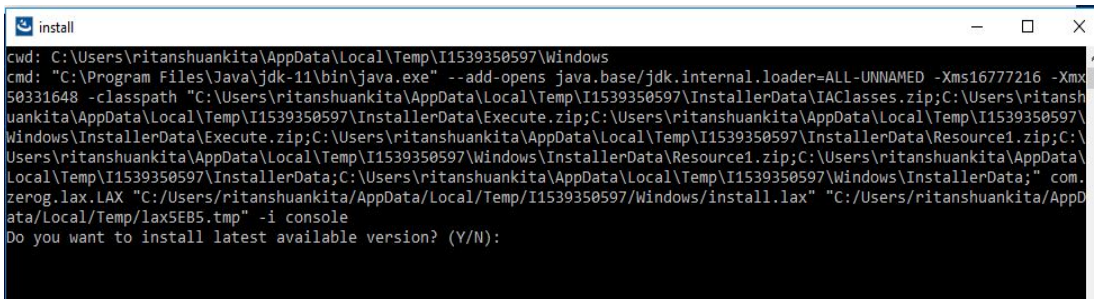


Task ***To Run-Time Behavior for Downloadable Updates:***

1. When the Force Install setting is set as **Yes**:
   - The iaupdate.xml file is present and it references to a newer version of the installation that is available in the specified path; And the base setup downloads and launches automatically.
2. When the Force Install setting is set as **No**:
   - The iaupdate.xml file is present and it references to a newer version of the installation that is available in the specified path; And the base setup displays Prompt as Follows.
     - Yes: Downloads and launches update installer setup
     - No: Proceed with base setup installation



3. If the iaupdate.xml file does not include a reference to a newer version of the installation, or if the iaupdate.xml file is not present, the installation of base setup proceeds.
4. If the iaupdate.xml file is present and it references a newer version of the installation that is not available in the specified path, the installation of base setup proceeds.



Installer UI Page

The Installer UI page includes the following views.

Table 9-37 ▪ Views on the Installer UI Page

View	Description
Look & Feel Settings View	Contains settings that enable you to configure the appearance and behavior of your installer.
Billboards View	Provides the ability to add billboards to the installer panels.
Help View	Defines general installer help—as plain text or HTML. It is also possible to define more specific help for each installer panel.

Look & Feel Settings View

The Look & Feel Settings view on the Installer UI page contains settings that enable you to configure the appearance and behavior of your installer. You can use these settings to customize many graphic elements and progress panels within the installer.

The settings in the Look & Feel Settings view are organized into the following main categories:

- UI Panel Settings Area
- General UI Settings Area
- Maintenance Mode Runtime Panel Label Settings Area
- Installer Icon Area

UI Panel Settings Area

Use the UI Panel Settings area in the Look & Feel Settings view to specify customizable advanced runtime UI themes in order to improve the runtime user experience of the installer.

A theme is a set of runtime UI settings that are used to customize the installer panels and frames. For example, you can specify fonts, font attributes, and even import your own fonts. You can customize the colors of windows and buttons, choose to use frameless windows, display your company's own logo in the installer steps panels, and many other settings.



Task ***To customize advanced runtime UI themes***

1. Click the **Installer UI** page and then click the **Look & Feel Settings** view.
2. In the **UI Panel Settings** area, click the **Custom UI Designer** button. When you click the **Custom UI Designer** button, a new window appears that contains all of your saved themes containing runtime UI design settings



Note ▪ *The **Custom UI Designer** button is disabled if the **GUI** check box in the Allowable UI Modes setting under the General UI Settings is not checked.*

3. When you click the **Custom UI Designer** button, a new window appears that contains all of your saved themes containing runtime UI design settings. The individual UI settings are grouped into the following three areas that are selectable in the **Installer Area** drop-down list:
   - Inner Install Frame
   - Installer Steps
   - Outer Installer Frame
4. When you make changes to any of the settings on the Inner Install Frame, Installer Steps, or Outer Installer Frame, your changes are stored to a theme named **unsavedTheme**. To save your changes to a new theme name, click the **Save Theme** button and a Save Theme dialog box appears that lets you save the current customization under a new or existing theme name.
5. The default for most of the theme settings is to Use default. To customize a setting that includes a Use default setting, deselect the **Use default** option (or else click the **Customize** radio button, where applicable) and then click the corresponding ellipsis button (...) to specify a new setting such as a new color or font.

   For any customizations that you make, a preview section shows the effect of the change and a **Panel to Preview** drop-down list is available so that you can view the effects of your customizations on any of the installation panels.

   Each of the runtime UI design customization settings are described in the following subsections.

Inner Install Frame

Choosing **Inner Install Frame** from the **Installer Area** drop-down list lets you customize the look and feel of the inner install frame section of the installer UI panels.

Table 9-38 ▪ Inner Install Frame Control

Inner Install Frame Control	Description
Background Color	Specify the background color for the main body part of each panel in your UI. To make the main body part of each panel transparent, select the **No Color** option.
Font Settings	Specify the font settings (font family, style, and color) of the inner install frame text. When you click the ellipsis button (...), a **Choose Font** dialog box appears. Here, you can also click **Import Font** to import a true type font to use. The imported font will also be shipped with your installer.
Show Status Message	Specify whether actions should display status messages in **Install** and **Uninstall** progress panels. By default, this option is set to **Yes**. If this option is set to **No**, status messages will not be displayed above the progress bar on the Progress Panel during **Install** and **Uninstall**.

Table 9-38 ▪ Inner Install Frame Control

Inner Install Frame Control	Description
Show Progress Bar	Specify whether actions should display a progress bar in **Install** and **Uninstall** progress panels. When this option is set to **Yes**, a standard progress bar that shows the progress of the entire installer by percentage complete, from 0% to 100%, is displayed. By default this option is set to **Yes**. If this option is set to **No**, the **Show Indeterminate Progress Bar** will be enabled. If this option is set to **Yes**, the **Show Indeterminate Progress Bar** will be disabled.
Show Indeterminate Progress Bar	Specify whether actions should display an indeterminate progress bar in **Install** and **Uninstall** progress panels. When this option is set to **Yes,** a bar is displayed that indicates that the installer is running, but it does not show the percentage of progress. By default, this option is set to **Yes** when **Show Progress Bar** is set to **No**. If this option is set to **No**, progress bar in Install and Uninstall progress panels will not be displayed at all.
Button Color	Use this setting to customize the colors used for the UI panel buttons. To override the default colors, choose the **Customize** radio button (which will clear the **Use Default** radio button setting), and then click the ellipsis button (...) and choose the appropriate color. To restore the default colors, select the **Use Default** radio button in this setting. You can specify colors for each of the following button characteristics: ● **Display Color**—Specify the color of the button. ● **Hover Color**—Specify the color to be used for the button to indicate that the mouse is hovering over the button. ● **Select Color**—Specify the color to be used for the button to indicate that the button has been clicked. ● **Disable Color**—Specify the button color to be used to indicate that the button is disabled.
Progress Bar Color	Use this setting to customize the color used for the installer progress bar. To override the default colors, choose the **Customize** radio button (which will clear the **Use Default** radio button setting), and then click the ellipsis button (...) and choose the appropriate color. To restore the default colors, select the **Use Default** radio button in this setting.

Table 9-38 ▪ Inner Install Frame Control

Inner Install Frame Control	Description
Border Settings	Use to specify panel and component border settings: • **Panel Border**—Specify whether you want to include a border around the main body area of your UI panels. • **Component Border**—Specify whether you want to include a border around components like description area, list boxes, text area, etc. in inner installer frame. This setting is disabled if the **GUI** check box in the **Allowable UI Modes** setting is cleared.

Installer Steps

Choosing **Installer Steps** from the **Installer Area** drop-down list lets you customize the look and feel of the inner installer steps section of the installer UI panels.

The Installer Steps controls are broken up into two sections:

- Installer Steps Controls
- Install Progress Panel

Installer Steps Controls

The following settings are available under Installer Steps.

Table 9-39 ▪ Installer Steps Controls

Installer Steps Control	Description
Show Installer Steps	Specify whether you want the left side of the installer panels to show the progress steps of the installation. If you select **Yes** in this setting, InstallAnywhere enables settings in the Installer Steps area, as well settings in the Install Progress area. If you select **No**, InstallAnywhere disables the majority of these settings.
Installer Steps Type	Select the type of UI control that you want to be displayed on the left side of the installer panels to show the progress of the installation. The **Show Installer Steps** setting must be set to **Yes** for the **Installer Steps Type** settings to be enabled. Available options of Installer Steps Type are: • **Images**—The left side of the progress installer panels use images to show progress. • **List of installer steps**—The left side of the progress installer panels use text-based steps to show progress.

Table 9-39 ▪ Installer Steps Controls

Installer Steps Control	Description
Enable Uninstall Label Settings	Specify whether you want to configure the uninstaller step labels. By default, this setting is set to **No**. This setting is enabled, if the **List of installer steps** is selected in the **Installer Steps Type** setting and disabled, if the **images** is selected in the **Installer Steps Type** setting.

Installer Steps Panel	This setting contains the following subsettings that let configure the appearance of the panel area that shows the installer steps.
	
	● **Beveled Border**—Specify whether you want to add a beveled border around the images or installer steps that are displayed on the left side of the installer panels.
	
	● **Scale Width**—Specify whether you want the size of the pane that contains the list of installer steps or image to dynamically resize to fit the size of the installer panel.
	
	● **Background Color**—Specify whether you want to use the background color behind the image or list of installer steps. If you select **Yes**, InstallAnywhere enables you to choose following drop-down options in this setting:
	
	  ● **Use window default**—Use the default window color of the machine running the installer as the background color for the are of the panels that shows the installer progress.
	
	  ● **Use dialog default**—Use the default dialog box color of the machine running the installer as the background color for the area of the panels that shows the installer progress.
	
	  ● **Customize**—Customize the color that shows the installer progress. To specify the color, click the ellipsis button (...) in this setting.

Table 9-39 ▪ Installer Steps Controls

Installer Steps Control	Description

Panel Image	This setting changes, depending on which option you select in the Installer Steps Type setting. Possible check boxes in this setting are:
	• **Use this image when no alternate image is specified for an installer step**—This check box is available if **Images** is selected in the **Installer Steps Type** setting. This check box is always enabled. Use the **File** subsetting to specify the image to use for the installer steps.
	• **Use this image as the background behind the list of installer steps**—This check box is available if **List of installer steps** is selected in the **Installer Steps Type** setting. To specify an image that you want to use behind the list of installer steps, use the **File** setting to identify the image. The recommended image dimensions are 174x308 pixels.
	The following Panel Image subheadings are also available:
	• **File**—Select the appropriate file, depending on what is selected in other settings of this view:
	  ◦ If **Images** is selected in the **Installer Steps Type** setting, click the ellipsis button (...) in this setting to select the image file that you want to use for installer steps.
	  ◦ If **List of Installer steps** is selected in the **Installer Steps Type** setting, click the ellipsis button (...) in this setting to select the background image for the area behind the list of installer steps. This setting is disabled if the **Use this image as the background behind the list of installer steps** check box in the **Panel Image** setting is cleared.
	• **Configure Install Labels**—To configure the text and icons for the installer steps that the installer UI shows for the installer progress, click the **Installer Steps Labels settings** button in this setting. The Installer Steps dialog box opens, enabling to you to configure settings as needed. To learn more, see Installer Steps Dialog Box.
	This setting is enabled, if the **List of installer steps** is selected in the **Installer Steps Type** setting and disabled, if the **Images** is selected in the **Installer Steps Type** setting.
	• **Configure Uninstall Labels**—To configure the text for the uninstaller steps that the uninstaller UI shows for the uninstaller progress, click the **Uninstaller Steps Labels settings** button in this setting. The Uninstaller Steps dialog box opens, enabling to you to configure settings as needed. To learn more, see Uninstaller Steps Dialog Box .

Table 9-39 ▪ Installer Steps Controls

Installer Steps Control	Description
Panel Image	Refer the following to enable/disable this setting: • Selecting the **List of installer steps** in the **Installer Steps Type** setting and selecting **Yes** in the **Enable Uninstall Label Settings** setting, enable this setting**.** • Either selecting the **images** in the **Installer Steps Type** setting or selecting **No** in the **Enable Uninstall Label Settings** setting, disable this setting**.**

Install Progress Panel

Use the Install Progress Panel control settings to configure settings for the image or labels on the left side of the installer panels that show the progress of the installation. The following settings are available under Install Progress Panel.

Table 9-40 ▪ Install Progress Panel Controls

Install Progress Panel Control	Description
Image	Specify which image you want your installers to show on the left side of the installer panels that show the progress of the installation. Available options are: • **User the Installer's Default Image**—To use the image that is specified in the **File** setting (under the **Panel Image** setting), select this option. • **User the Same Image as the Previous Panel**—To use the image that is specified for the previous panel, select this option. • **Do not Display an Image**—To show no additional image in this panel, select this option. • **Specify an Image (170x305)**—To specify the image that you want to use for the installer step images, select this option, and then use the **File** setting to select the image file. The size of the install progress panel is 170 pixels wide by 305 pixels tall. installer dimensions may change slightly, depending on the platform, to better display text and different fonts. This setting is enabled if **Images** is selected in the **Installer Steps Type** setting. This setting is disabled if **List of installer steps** is selected in the **Installer Steps Type** setting.
File	If you select the Specify an Image (170x305) option in the **Image** setting (under **Install Progress Panel**), click the ellipsis button (...) in this setting to select the appropriate file.

Table 9-40 ▪ Install Progress Panel Controls

Install Progress Panel Control	Description
Install Progress Label	Use this setting to specify which label you want your installers to show on the Install Progress panel. To specify the label, click the **Install Step Label Settings** button in this setting. For more information, see Install Step Label Settings Dialog Box. This setting is enabled, if the **List of installer steps** is selected in the **Installer Steps Type** setting and disabled, if the **Images** is selected in the **Installer Steps Type** setting.
Uninstall Progress Label	Use this setting to specify which label you want your uninstallers to show on the Uninstall Progress panel. To specify the label, click the **Uninstall Step Label Settings** button in this setting. For more information, see Uninstall Step Label Settings Dialog Box. Refer the following to enable/disable this setting: ● Selecting the **List of installer steps** in the **Installer Steps Type** setting and selecting **Yes** in the **Enable Uninstall Label Settings** setting, enable this setting**.** ● Either selecting the **images** in the **Installer Steps Type** setting or selecting **No** in the **Enable Uninstall Label Settings** setting, disable this setting**.**

For information on specifying how the progress of a merge module should be displayed, see Specifying How to Show the Progress of a Merge Module Installation.

Outer Installer Frame

Choosing **Outer Installer Frame** from the **Installer Area** drop-down list lets you customize the look and feel of the outer installer frame sections of the installer UI panels.

Table 9-41 ▪ Outer Installer Frame settings

Outer Installer Frame Settings	Description
Background Image	To include a background image in the UI of your installer, select the **Use background image** check box in this setting. The following subsettings are available under **Background Image**: ● **File**—This setting indicates the path and file name of the image file that you want to use for the background of your installer panels. To browse to a different image, click the ellipsis button (...) in this setting. To revert back to the default image file, click the revert button in this setting. ● **Scale to Fit**—This setting lets you specify whether you want the image to be scaled to fit the width of the panel, the height of the panel, or both the width and height of the panel. Select the appropriate check boxes as needed. ● **Mirror for RTL**—This setting is for languages, such as Arabic and Hebrew, that are read from right to left. If your project includes support for these locales, and if you want the background image that you specify in the File setting to be mirrored on its vertical axis (flipped left-right) for those locales, select this check box.  ***Note ▪*** *You can use and configure animated or static GIF file as a background image for installer panels.*
Titles, Labels and InstallAnywhere Font	Use this setting to customize the colors for the titles, labels, and InstallAnywhere branding on your UI's panels. To override the default colors, clear the **Use default** check box in this setting, and hen click ellipsis button (...) in this setting and specify the appropriate colors. In the resulting dialog that appears, you can also click **Import Font** to import a true type font to use. The imported font will also be shipped with your installer. To restore the default colors, select the **Use default** check box in this setting. If you select the **Gray beveled text** option in the **Beveled Logo** setting, the InstallAnywhere branding and the horizontal line that is next to the InstallAnywhere branding are displayed in gray; this setting overrides any color choice that you specify in the **Titles, Labels, and InstallAnywhere Font** color settings.

Table 9-41 ▪ Outer Installer Frame settings

Outer Installer Frame Settings	Description
Beveled Logo	Indicate how you want the InstallAnywhere branding to be displayed on your UI's panels. Available options are: • **Gray beveled text**—The branding is shown with a gray bevel effect. If you select the **Gray beveled text** option in the **Beveled Logo** setting, the InstallAnywhere branding and the horizontal line that is next to the InstallAnywhere branding are displayed in gray; this setting overrides any color choice that you specify in the **Titles, Labels, and InstallAnywhere Font** color settings. • **non-beveled text**—The branding is shown without a bevel effect. The color that is configured in the **Titles, Labels and InstallAnywhere Font** color setting is used for the branding.
Installer Size	Specify the width and height (in pixels) that you want to use for your installer UI panels at default operating system resolution or DPI settings. To restore the default size for panels (600 pixels for the width and 400 pixels for the height), click the undo button.  ***Note ▪*** *The size of the Installer UI panels are scaled in accordance with the resolution/DPI settings.*
Frameless Window	Specify whether to remove the frame around the installer window in order to achieve a minimalist look and feel common in Windows 10. By default, the **Frameless Window** setting is set to **No**. When you set the **Frameless Window** setting to **Yes**. The following subsettings are also enabled—allows you to choose an image other than the default images that are used for minimize and close buttons, and to specify whether to wrap the installer's long title across multiple lines within the installer UI: • **Minimize Button**—Click the ellipsis (...) button in this setting to select the appropriate file to replace the default graphic that is used for the installer Minimize button.

Table 9-41 ▪ Outer Installer Frame settings

Outer Installer Frame Settings	Description

(continued)	• **Close Button**—Click the ellipsis (...) button in this setting to select the appropriate file to replace the default graphic that is used for the installer Close button.
	• **Wrap Installer Title—**Specify whether to wrap the installer's long title across multiple lines within the installer UI, without truncation and without any alteration to the inner installer frame dimensions. You can select the following appropriate options:
	  • **Yes**—The installer's long title wraps across multiple lines within the installer UI, ensuring that it is fully displayed—without truncation and without any alteration to the inner installer frame dimensions.
	  • **No**—The installer's long title is truncated and ends with an ellipsis (...) when the title exceeds the single-line space available in the title bar of the installer UI.
	
	Note ▪ *When the* ***Frameless Window*** *setting is set to* ***No****, the* ***Minimize Button****,* ***Close Button****, and* ***Wrap Installer Title*** *subsettings are disabled.*

General UI Settings Area

Use the General UI Settings area in the Look & Feel Settings view to customize the appearance of your installer.

The settings in the General UI Settings area are organized into the following main categories:

- Allowable UI Modes
- Startup Splash Screen
- Choose Language Console Panel

Allowable UI Modes

Select one or more UI modes that you want your installers to support. Available options are:

- **GUI**—The project supports graphical (GUI) installers.
- **Console**—The project supports console installers.
- **Silent**—The project supports the ability to run the installers in silent mode.



Note ▪ *InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers cannot run on these 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 or the Locale selector.)

Table 9-42 ▪ Startup Splash Screen Controls

Control	Description
Image	This setting indicates the path and file name of the splash screen image file that you want to use for your installer. Supported image formats include .gif, .jpg, .jpeg, and .png. Both animated and static GIF files are supported. The splash screen image can be any size; the preferred size is 470 pixels for the width and 265 pixels for the height. To browse to a different image, click the ellipsis button (...) in this setting. To revert back to the default image file, click the Default button in this setting. To see how the splash screen looks with the currently specified background image, click the Preview button in this setting. This setting is disabled if the GUI check box in the Allowable UI Modes setting is cleared.
Title	Specify the text that you want to be displayed in the title bar of the installer's splash screen. This setting is blank by default, and no text is displayed in the title bar. The value that you enter 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 `installer.properties` file. This setting is disabled if the GUI check box in the Allowable UI Modes setting is cleared.
Instructions	Specify the text that you want to be displayed on the splash screen next to the drop-down list that contains the available locales for the installer. This setting is blank by default, and no text is displayed next to the drop-down list. The instruction text is displayed on the splash screen if the installer supports more than one locale. This setting is disabled if the GUI check box in the Allowable UI Modes setting is cleared.

Table 9-42 ▪ Startup Splash Screen Controls (cont.)

Control	Description
Confirmation	Specify the text that you want to be displayed for the confirmation button on the splash screen. The default value is OK. The confirmation button is displayed on the splash screen if the installer supports more than one locale. This setting is disabled if the GUI check box in the Allowable UI Modes setting is cleared.



Note ▪ *For details on localization elements that are related to the startup splash screen (splashScreenGUIImageName, splashScreenGUIImagePath, splashScreenGUIInstructions, splashScreenGUITitle, splashScreenGUIConfirm, splashScreenConsoleTitle, and splashScreenConsolePrompt), see Common Localizable Elements. 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 include a Choose Locale console panel to enable end users to set the language for the installer. The controls in this area enable you to customize the title and prompt on that console.

Table 9-43 ▪ Choose Language Console Controls

Control	Description
Title	Specify the text that you want to be displayed for the title of the Choose Locale console panel. The default text is: `Choose Locale...` The Choose Locale console panel is displayed if the console installer supports more than one locale. This setting is disabled if the Console check box in the Allowable UI Modes setting is cleared.
Prompt	Specify the text that you want to be displayed for the prompt on the Choose Locale console panel. The default text is: `CHOOSE LOCALE BY NUMBER` The Choose Locale console panel is displayed if the console installer supports more than one locale. This setting is disabled if the Console check box in the Allowable UI Modes setting is cleared.

Maintenance Mode Runtime Panel Label Settings Area



***Note** ▪ The Maintenance Runtime Panel Label Settings area is enabled only if maintenance mode is enabled in your project. To learn more, see Enabling Maintenance Mode.*

The Maintenance Runtime Panel Label Settings area in the Look & Feel Settings view lets you customize the text and images that are displayed on the Maintenance Mode panel of the Change Installation Launcher.

The following settings are available on the Maintenance Runtime Panel Label Settings area.

Table 9-44 ▪ Maintenance Runtime Panel Label Settings area

Setting	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:`
Add Features **Remove Features** **Repair Installation** **Uninstall Product**	Use the following subsettings to define various elements on the Maintenance Mode panel: • **Title**—Enter the text that identifies the selection. This text appears in bold. • **Description**—Enter the text that describes the selection. This text appears in plain text directly under the title. • **Icon**—This setting identifies the default icon that is displayed for the selection. To select a different icon, click the ellipsis button (...) in this setting and then select an image (.gif, .jpg, .jpeg, or .png). To revert back to the default image, click the Default button.

Installer Icon Area



***Note** ▪ This information applies to installers that target Windows platforms.*

Use the Installer Icon area in the Look & Feel Settings view to specify a custom Windows icon file (.ico) for your project. InstallAnywhere uses the icon that you specify for the self-extractor .exe files that it builds for Windows-based build targets.

InstallAnywhere incorporates a custom installer icon at build time by changing the icon of self-extractor .exe files in the InstallAnywhere installed location. Therefore, if you specify a custom icon file, it is recommended that you build one project at a time. 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 wrong icon for one or both projects may be used.

The Installer Icon area includes the following setting.

Table 9-45 ▪ Setting in the Installer Icon Area

Setting	Description
Executable Icon Path	This setting specifies the path and file name for the icon file (.ico). To specify a different file, click the ellipsis button (...) in this setting. To revert to the default icon, click the undo button in this setting. The default value for this setting is: `com/zerog/ia/installer/images/Icon1.ico`  ***Note ▪*** *This setting is available if an installer builds only on Windows.*
Installer Title Image	This setting specifies the path and file name for the title image of your installer that is placed on the Start screen. To specify a different file, click the ellipsis button (...) in this setting. To revert to the default icon, click the undo button in this setting. The default value for this setting is: `com/zerog/ia/installer/images/iaIcon.png`  ***Note ▪*** *This setting is available if an installer builds only on Windows.*

Billboards View

Billboards are images that appear in the large right pane of the installer while files are being installed. You can add billboards to your projects to display information to end users during the installation process. The billboards can be used to communicate, advertise, educate, and entertain end users. For example, billboards can present overviews on new features of the product being installed or other products from your company. Each billboard is a file that you or your company's graphics department creates for control over the look and feel of your installers.

Use the Billiards view on the Installer UI page to add billboards to your project's installer panels.

Several billboard graphics may be added for larger (and longer) installations. For small installations, like the tutorial OfficeSuite example, only one billboard is displayed.

You can optionally assign billboards to features; a billboard that is associated with a specific feature is displayed only if the feature that it is associated with is going to be installed.

If you add more than one billboard, the billboards are displayed at run time in the order in which they are listed in the Billboard List of the Billboards view, from top to bottom.

The Billboards view includes the following controls:

Table 9-46 ▪ Controls on the Billboards Tab

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.
Display billboards for every *nn* seconds	To display a billboard for a specified number of seconds, select this check box and then enter a number in the seconds box. For more information, see Customization of the User Interface.
Add Billboard	To add a billboard to your project, click this button. The Choose an Image File dialog box opens, prompting you to select the billboard image.  ***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.*
Remove	To remove the selected billboard from your project, click this button.
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.

Help View

Use the Help view on the Installer UI page to add help functionality to your installer. To add help, select the **Enable installer help** check box in this view. Selecting HTML instead of plain text for the format allows for greater formatting control of the message.

If the **Use the same help text for all panels** option in this view is selected, you can define a single help message for your installer.

To have different help instructions for each installer panel, select the **Use different help for each text panel** option. The action customizer, available at the bottom of the Install, Pre-Install, Post-Install, Pre-Uninstall, and Post-Uninstall views on the Sequence page, 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.



Important ▪ *If you enable installer help and supply help for installer panels, do not include* `<META>` *tags in the HTML code of the help panels. An InstallAnywhere installer is unable to display help pages that contain* `<META>` *tags.*

Organization Page

Use the Organization page in the Advanced Designer to arrange install sets, product features, components, and merge modules. Install sets and features enable end users to customize which parts of your product they install. Install sets are groupings of features such as Typical Install or Minimal Install. Each feature in a project is made up of one or more components. A component is the smallest installable part of a product from the installation developer's perspective; components are invisible to the end user. Components may be much more than files; they can be sophisticated actions that are required to install and run applications or features properly.

Once you add an install set in the Install Sets view of your project, you can use the Features view to assign features to that install set. When you add a feature in the Features view of your project, you can use the Components view to assign components to that feature. Once you add a component in the Components view of your project, and once files and actions are included in the Install view on the Sequence page, you can use the Components view to assign the files and actions to the component.

For more information on installer organization, see Organizing Files for Your Installer.

The Organization page includes the following views

Table 9-47 ▪ Views on the Organization Page

View	Description
Install Sets View	Define your project's install sets.
Features View	Define your project's features and assign them to your project's install sets.
Components View	Define your project's components and assign them to your product's features.
Modules View	Import merge modules into your project.
Hosts View	Define application server hosts and database server hosts for your project. You can use these hosts to target application servers and database servers with your installers.

Install Sets View

An install set is a set of product features. The Install Sets view on the Organization page is where you add, name, remove, and order install sets in the installer. Use the Install Set List in this view to define the install sets that you want to make available to end users and mark which ones you want to be selected for installation by default.

When the installer requests install set information, each install set is represented by a graphic element. The Choose Image button enables you to select the image that you want your installer to display.

Rules may be associated with an install set, and that association is created by clicking the Rules tab 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 is not displayed.

To assign features to install sets, use the Features view on the Organization page.

Features View

Features are meant to identify distinct parts of the product so that end users may choose whether to install them. In an InstallAnywhere project, product features are groupings of components. Use the Features view on the Organization page to add, name, remove, and order the features in your project. This view is also where you assign the features to the appropriate install sets in your project.

Rules may be associated with a feature set, and that association is created by clicking the Rules tab 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 is not displayed.

To assign components to features, use the Components view on the Organization page.

Components View

Components are the smallest installable unit of organization from the perspective of the installation developer. Unlike features and install sets, components may be versioned. Components are uniquely identified; this enables you to update a specific component or use the Find Component in InstallAnywhere Registry action to locate a particular component.

Use the Components view on the Organization page to add, remove, and configure components. This view is also where you assign the components to the appropriate features in your project.

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. You can include these components as part of a suite installer, use them to define prerequisite dependencies for your package, and even enable multiple applications to share common components across a system.

The settings in the Components view are organized by category on the following tabs:

- Properties Tab
  - Component Subtab
  - Dependency Subtab
- Rules Tab
- Tags Tab

Properties Tab

When you select a component in the Components view, the following settings are available on the Properties tab:

Table 9-48 ▪ Settings on the Properties tab of the Components View

Setting	Description
Unique ID	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. 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.  ***Note ▪*** *The Find Component in InstallAnywhere Registry action enables you to detect a component based on its UUID and version.*
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	Select the type of component. Available options are: ● **Standard Component**—This component type is always installed, regardless of previous installations or dependencies. The uninstaller for the product with which a standard component is installed removes standard components. ● **Shared Component**—This component type is installed if it has not already been installed on the system. A shared component can be made available to other products. During uninstallation or upgrade, a shared component is removed only if no other product on the target system references it. The uninstaller for the last product that references the shared component removes it. ● **Dependency**—If you configure a component to be a dependency, instead of installing this component, the installer requires that this component has already been installed on the system. Dependencies are not elements of your installer, but rather are prerequisite requirements that the installer enforces.
Optimize registry entry	Use this option to instruct InstallAnywhere how to update the global registry when installing a new version of a previously installed component. ● **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

When you select a standard component or a shared component in the Components view, the following settings are available on the Component subtab of the Properties tab:

Table 9-49 ▪ Settings on the Component Subtab

Setting	Description
Version	Enter a 4-digit version.
Key File	A key file is a single file that identifies the component. A key file should always be included in a component. 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.  ***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 InstallAnywhere Registry action is used.*
Comment	Enter a comment to identify the purpose of this component.

Dependency Subtab

When you select a shared component or a dependency component in the Components view, the following settings are available on the Dependency subtab of the Properties tab:

Table 9-50 ▪ Settings on the Dependency Subtab

Setting	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	You can use the this 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.
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.*

Rules Tab

Use the Rules tab of the Component customizer in the Components view to associate rules with a component. The installer evaluates rules for a component before installing the component. For more information, see Rules Reference.

Tags Tab

Use the Tags tab of the Component customizer in the Components view to associate build configuration tags with a component. To learn more, see Assigning Tags to Project Elements.

Modules View

Use the Modules view on the Organization page to import merge modules into your project and configure their settings.



***Tip** ▪ Merge modules are created as a distribution option in the Build Installers view of the Build page.*

The Modules view consists of two main areas:

- The Modules area at the top of this view contains the list of merge modules that have been imported into your project. This area also has buttons that let you import merge modules into or remove merge modules from your project.
- The customizer area at the bottom of this view contains the settings that are applicable to the merge module that is selected in the Modules area. The settings that are displayed varies, depending on whether the selected merge module is static or dynamic.

The following elements comprise the Modules area at the top of the Modules view:

Table 9-51 ▪ Modules Area in the Modules View

Option	Description
Imported Merge Module List	List of merge modules that have been imported to this project.

Table 9-51 ▪ Modules Area in the Modules View (cont.)

Option	Description
Import Merge Modules	To select a static merge module to merge into the current installer, click this button, and then customize it on the Merge Module customizer. Note the following details about static merge modules: ● All of the merge module's features, components, files, actions, and panels (optionally) are 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 flexibility for scheduling actions. ● Changes made to a static merge module are not reflected in the parent project.
Import Dynamic Merge Module	To select a dynamic merge module to merge into the current installer, click this button, and then customize it on the Dynamic Merge Module customizer. Note the following details about dynamic merge modules: ● All of the merge module's files, actions, and panels (optionally) are combined into current project. ● The dynamic merge module's actions are imported together as an action group in the Pre-Install and Post-install sequences. ● Dynamic merge modules are recommended if you have merge modules whose contents frequently change. A parent project automatically refreshes dynamic merge modules when the parent project is loaded and built.
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

If you add a merge module to the Imported Merge Modules List, first the InstallAnywhere Merge Module Import Assistant dialog box opens, and then the Merge Module customizer opens. The Merge Module customizer includes the following settings:

Table 9-52 ▪ Merge Module Customizer

Setting	Description
Module Name	This read-only setting identifies the name of the merge module.
Module Path	This read-only setting lists the path to the selected merge module.
Creation Date/Time	This read-only setting shows the date and time when the merge module was created.
Vendor	This read-only setting shows the name of vendor that created the merge module.

Table 9-52 ▪ Merge Module Customizer (cont.)

Setting	Description
Version	This read-only setting shows the 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

If you add a dynamic merge module to the Imported Merge Modules List, the Dynamic Merge Module customizer opens. The Dynamic Merge Module customizer includes the following tabs:

- General Tab
- Pre-Install Tab
- Install Tab
- Post-Install Tab

General Tab

The General tab on the Dynamic Merge Module customizer includes the following settings:

Table 9-53 ▪ General Tab on the Dynamic Merge Module Customizer

Setting	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 when 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 Tab

The Pre-Install tab on the Dynamic Merge Module customizer includes the following settings:

Table 9-54 ▪ Pre-Install Tab on the Dynamic Merge Module Customizer

Setting	Description
Import Pre-Install Actions	To import the Pre-Install actions in this merge module into the main project, select this check box.

Table 9-54 ▪ Pre-Install Tab on the Dynamic Merge Module Customizer (cont.)

Setting	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.

Install Tab

The Install tab on the Dynamic Merge Module customizer includes the following settings:

Table 9-55 ▪ Install Tab on the Dynamic Merge Module Customizer

Setting	Description
Import Install Actions	To import the Install actions in this merge module into the main project, select this check box.
Uninstall Merge Module when parent is uninstalled	If you want this merge module to be uninstalled when the main project is uninstalled, select this check box.  ***Note ▪*** *The point at which merge modules are uninstalled can also be configured using the Uninstall Merge Modules action in the Uninstall view on the Sequences page.*
Add merge module log to parent log	To append the installation log of this merge module to the main project log, select this check box. By default, this check box is not selected.  ***Note ▪*** *For this merged logging functionality 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.* *To append the merge module logs to the parent log, you must select* ***Plain text format*** *in the Log Format setting, which is available in the General Settings view of the Project page. Adding the merge module log to the parent log is not available for the XML log format.*
Show progress dialog	Specify the way you want the merge module's installation progress to be displayed by selecting or clearing this check box: • To display an intermediate progress bar while this merge module is being installed, select this check box. • To suppress the intermediate merge module progress bar and integrate and the progress of the merge module installation into the progress bar of the parent installer, clear this check box.
Input Variables	Listing of the input variables in this merge module that can be set.

Table 9-55 ▪ Install Tab on the Dynamic Merge Module Customizer (cont.)

Setting	Description
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 Tab

The Post-Install tab on the Dynamic Merge Module customizer includes the following settings:

Table 9-56 ▪ Post-Install Tab on the Dynamic Merge Module Customizer

Setting	Description
Import Post-Install Actions	To import the Post-Install actions in this merge module into the main project, select this check box.
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.

Hosts View

The Hosts view on the Organization page is where you define hosts other than the default operating system host: application server and database server hosts. You can use these hosts to target application servers and database servers with your installers.

InstallAnywhere displays the hosts that you define in this view at the root level of the visual tree in the Install view of the Sequence page. Use the Install view to assign files, launchers, and actions to your operating system, application server, and database server hosts.

Table 9-57 ▪ Settings in the Host View

Setting	Description
Host List	Lists operating system, application server, and database server hosts. Every project includes a default operating system host that cannot be removed.
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. Note that if an action has been added to the host on the Sequence page, this button is disabled. In order to remove the host from the list, you must first delete the action from the host.

Table 9-57 ▪ Settings in the Host View (cont.)

Setting	Description
Host Customizer	Shows configuration controls for the currently selected host. (Operating system hosts require no configuration.) • For details about the Application Server customizer, see Application Server Host Settings. • For details about the Database Server customizer, see Database Server Host Settings.



Note ▪ *If you enter a password value for an application server host or a database server host, the password value is automatically encrypted according to the security settings that are specified in the Variables view on the Project page.*

Application Server Host Settings

When you add an application server host in the Hosts view of the Organization page, you can customize its settings. The settings that are available vary slightly, depending on the type of server.



Tip ▪ *Note that the host settings that are available in the Hosts view are also available when you select the host in the Install view on the Sequence page. Any changes that you make to the settings in one view are reflected instantly in the other.*

The Application Server customizer includes the following tabs:

- General Settings
- Optional Settings

General Settings

When you select an application server host in the Hosts view, the following settings are available on the General Settings tab.

Table 9-58 ▪ Settings on the General Settings Tab of the Application Server Customizer

Setting	Description
Server Type	Select the type of application server to which you want to deploy.

Table 9-58 ▪ Settings on the General Settings Tab of the Application Server Customizer (cont.)

Setting	Description
Server Path	 ***Note ▪*** *This setting is available for the following server types:* ● *Geronimo* ● *JBoss* ● *WebLogic* Enter the file path to the server. Note the following guidelines for this setting: ● For Geronimo and WebLogic: If the Bundle Connection Libraries from Local Server Install check box is selected, 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.) ● For Geronimo and WebLogic: If the Bundle Connection Libraries from Local Server Install check box is cleared, 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.)
Connection Name	 ***Note ▪*** *This setting is available for WebSphere servers.* Select the name of the WebSphere server. This is typically useful if your installer deploys Web applications to more than one application server host. The name should be unique across all WebSphere application server hosts.
Server Name	 ***Note ▪*** *This setting is available for Tomcat servers.* Specify a name for the selected Tomcat server. This is typically useful if your installer deploys Web applications to more than one application server host. The name should be unique across all Tomcat application server hosts.

Table 9-58 ▪ Settings on the General Settings Tab of the Application Server Customizer (cont.)

Setting	Description
Deployment Options	 ***Note ▪*** *This setting is available for Tomcat and WebSphere servers.* Select the check boxes of the deployment options that you want your installer to support. You can offer all or any combination of the following options for Tomcat servers: • Local Tomcat server • Remote Tomcat server • Save the WAR file locally on the machine on which the installer is running for later deployment To learn more, see Specifying Which Deployment Options to Support for Apache Tomcat Servers. You can offer either or both of the following options for WebSphere servers: • Local or remote WebSphere server • Save the EAR/WAR file locally on the machine on which the installer is running for later deployment • Save the EAR/WAR locally on the target system for later deployment
Dependency Library	 ***Note ▪*** *This setting is available for WebSphere servers.* Specify the path to the WebSphere dependency library. To download one, click the link.
Host Name	 ***Note ▪*** *This setting is available for the following server types:* • *Geronimo* • *WebLogic* Enter the host name of the server to which you want to deploy software. The value that you enter must be an IP address or resolvable name—for example, **`localhost`**, **`127.0.0.1`**, or **`mycompany.com`**.
Port	 ***Note ▪*** *This setting is available for the following server types:* • *Geronimo* • *WebLogic* Enter the port for the application server to which you want to deploy. For example, **`1099`**, **`8880`**, **`7001`**, or **`4848`**.

Table 9-58 ▪ Settings on the General Settings Tab of the Application Server Customizer (cont.)

Setting	Description
Bundle Connection Libraries from Local Server Install	 ***Note ▪*** *This check box is available for the following server types:* • *Geronimo* • *WebLogic* To bundle the connection libraries from this server at build time, select this check box. 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. • If the Bundle Connection Libraries from Local Server Install check box is selected, 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 the Bundle Connection Libraries from Local Server Install check box is cleared, the installer looks, at install time, for the libraries based on the path that you specified for Server Path setting. 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

When you select certain types of application server hosts (Geronimo or WebLogic) in the Hosts view, the Option Settings tab is available. The following settings are available on the Optional Settings tab.

Table 9-59 ▪ Settings on the Optional Settings Tab of the Application Server Customizer

Control	Description
Authenticate with Server	If you want your installer to handle server authentication for the selected server host, select this check box and then enter the appropriate credentials in the subsettings.
Username	Provide a user name for an account with sufficient privileges to perform the WAR/EAR deployment tasks for your installer.
Password	Provide a password for the account specified by Username.  ***Note ▪*** *If you enter a password, the password value is automatically encrypted according to the security settings that are specified in the Variables view on the Project page.*

Database Server Host Settings

When you add a database server host in the Hosts view of the Organization page, you can customize its settings.



Tip ▪ *Note that the host settings that are available in the Hosts view are also available when you select the host in the Install view on the Sequence page. Any changes that you make to the settings in one view are reflected instantly in the other.*

The Database Server customizer includes the following settings.

Table 9-60 ▪ Database Server Customizer Settings

Setting	Description
Server Type	Select the type of database server with which your installer must connect.  ***Note*** ▪ *Generic JDBC connections enable your installer to connect to database servers for which InstallAnywhere does not have built-in support.*
Custom Connection String	This setting is available if you select Generic JDBC Connection option for the Server Type setting. Enter the string that your installer should use to connect to the database that you are targeting. The connection string must work with the database server's syntax requirements.
Database Host Name	Specify a name for the selected database host. This is typically useful if your installer runs SQL scripts on more than one database host. The name should be unique across all database server hosts.
Custom Driver Settings	To provide custom values for the JDBC Driver Class setting or the Dependencies setting, select this check box.
JDBC Driver Class	Indicate the class for the JDBC driver that the database server supports. This setting is enabled if the Custom Driver Settings check box is selected.
Dependencies	Specify the libraries that you want to bundle as dependencies for the selected database server host. This setting is enabled if the Custom Driver Settings check box is selected. If InstallAnywhere does not provide the driver for the selected database type, a **More on drivers** link is displayed in this area. You can click this link to visit the driver vendor's Web site.
Add JAR or ZIP	To add to your project dependencies for the selected database server, click this button and then browse to the .jar or .zip file that you want to include.

Table 9-60 ▪ Database Server Customizer Settings (cont.)

Setting	Description
Remove	To delete a dependency in the Dependencies setting, select it and then click the Remove button.



Note ▪ *For information about running a SQL script on a database server, see Run SQL Script Action.*

Sequence Page

The Sequence page contains links to views that you can use to configure the order of panels and actions that occur at run time. The Sequence page includes the following views.

Table 9-61 ▪ Views on the Sequence Page

View	Description
Pre-Install View	Defines the actions that execute before the installer transfers the files to be installed on target systems. Actions that are scheduled during the Pre-Install sequence typically determine whether target systems meet any required system requirements; they also determine what to install on target systems. In general, the Pre-Install sequence is the sequence in which most configuration options are presented to end users. It is common to require this information at this stage of the installation, and to automate later configurations based on input from end users. Actions that occur during the Pre-Install sequence are not uninstalled.
Install View	Defines the actions that occur when the installer transfers files to target systems. Actions that are scheduled for the Install sequence make changes to target systems, such as creating folders, expanding archives, or moving files.
Post-Install View	Defines the actions that occur after the installer has made the required changes to target systems. Post-Install actions are generally actions that require files to be installed; examples are showing the Readme file or launching the product that was just installed. Actions that occur during the Post-Install sequence are not uninstalled.
Pre-Uninstall View	Defines the actions that execute before the installer removes the product's files from target systems.
Uninstall View	Defines the actions that execute during uninstallation. The Uninstall sequence enables you to add, remove, or change some of the uninstall actions.
Post-Uninstall View	Defines the actions that occur after the uninstaller has removed the product's files from target systems.

Pre-Install View

Use the Pre-Install view on the Sequence page to schedule the panels and actions that occur before the installer transfers the files to be installed on target systems. Actions that are scheduled during the Pre-Install sequence typically determine whether target systems meet any required system requirements; they also determine what to install on target systems.

In general, the Pre-Install sequence is the sequence in which most configuration options are presented to end users. It is common to require this information at this stage of the installation, and to automate later configurations based on input from end users.

When an action is scheduled for the Pre-Install sequence, it may be executed more than once if an end user clicks Previous and then Next repeatedly. This could cause several errors for actions that modify files, such as Modify Text File - Single File or Register Windows Service. For these types of actions, it is recommended that you schedule them during the actual installation (within the Install sequence) to prevent this type of error.

For more information, see:

- Customizing the Pre-Install Sequence
- Adding Pre-Install Actions

Install View

Use the Install view on the Sequence page to schedule actions that occur as the installer is transferring files to the target system; this view is also where you assign files, launchers, and actions to components and features. Actions that are scheduled for the Install sequence make changes to target systems, such as creating folders, expanding archives, or moving files.



Important ▪ *Actions (including files, launchers, folders, and other actions) that occur in the Install sequence are uninstalled by the project's uninstaller. Actions that occur in the Pre-Install and Post-Install sequences are not automatically uninstalled.*

The Install view contains the following controls:

Table 9-62 ▪ Install View Controls

Control	Description
Visual Tree	The Visual Tree displays the hierarchy of hosts, files, folders, and actions that comprise the Install sequence. The items are organized according to destination folder. Instead of displaying hard-coded destinations, destinations are represented by magic folders such as $USER_INSTALL_DIR$, which represents the main product installation directory on a target system. You can add files to the Visual Tree by clicking the Add Files button in this view or dragging and dropping directly from your file manager. (Windows Explorer, the OS or OS X Finder, and some UNIX/Linux File Managers are supported.) Once an action is listed in the Visual Tree, you can move items around in the hierarchy by dragging and dropping them or by clicking the arrow buttons. Right-clicking in the Visual Tree shows a context menu that you can use to manage the files and actions in the sequence. For example, right-clicking a file enables you to specify a magic folder where you want to place the file.
Assign To	This setting lets you specify whether the items in the Visual Tree are displayed according to the product features with which they are associated, or according to the components with which they are associated. Depending on what you select in this list, the columns to the right of this setting display the names of features or components. Note that you can associate a file with a maximum of one component. When you add a file to the Visual Tree, InstallAnywhere automatically associates it with a component. You can override the association if appropriate.
Set Classpath	The Set Classpath button determines the folders and archives that 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.

Table 9-62 ▪ Install View Controls (cont.)

Control	Description
Add Files	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. The files and directories that you select in the Add Files to Project dialog box are added to the Visual Tree: ● 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 subfolder is selected, the location of the added files depends on whether the subfolder is expanded or collapsed: The files are inserted into the subfolder when the subfolder is expanded but they are inserted into the parent folder if the subfolder is collapsed. 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.
Add Launcher	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 that your installer deploys.  ***Note ▪*** *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 is natural for their platform.* *A LaunchAnywhere application must point to a file with a main class.* *For more information about LaunchAnywhere, see LaunchAnywhere and Creating Launchers for Java Applications.*
Add Action	The Add Action button opens the Choose an Action dialog box. For more information about actions, see Actions, Customizing the Pre-Install Sequence, and Action Customizers and the Action Execution Sequence.
Remove	The Remove button deletes the currently selected action, file, or directory from the Visual Tree.
Properties Tab	The Properties tab displays the customizers for the selected project element.
Rules Tab	Use the Rules tab to add rules to the selected project element.
Tags Tab	Use the Tags tab to add build configuration tags to the selected category or action. To learn more, see Assigning Tags to Project Elements.

Table 9-62 ▪ Install View Controls (cont.)

Control	Description
Rollback Tab	If the Enable Rollback check box is selected in the Advanced view of the Project page, you can select the check boxes on this tab to specify rollback behavior for the selected project element: • **If rollback is triggered at install time, uninstall this action/resource**—If you want the selected project element to be uninstalled if the product installation is cancelled or encounters a fatal error, select this check box. If this check box is not selected and a rollback occurs, the project element is not uninstalled. • **If a fatal error occurs on this action/resource at install time, trigger uninstall**—If you want a rollback to be triggered if the selected project element fails to execute properly or to be installed properly, select this check box. If this check box is not selected and the execution of this action results in a fatal error during installation, a rollback is not triggered.

Post-Install View

Use the Post-Install view on the Sequence page to schedule the panels and actions that occur after the installer has made the required changes to target systems. Post-Install actions are generally actions that require files to be installed; examples are showing the Readme file or launching the product that was just installed.

Actions that occur during the Post-Install sequence are not uninstalled.

When an action is scheduled for the Post-Install sequence, it may be executed more than once if an end user clicks Previous and then Next repeatedly. This could cause several errors for actions that modify files, such as Modify Text File - Single File or Register Windows Service. For these types of actions, it is recommended that you schedule them during the actual installation (within the Install sequence) to prevent this type of error.

For more information, see:

- Customizing the Post-Install Sequence
- Adding Post-Install Actions

The Post-Install view contains the following controls:

Table 9-63 ▪ Post-Install View Controls

Control	Description
Action List	The Action List displays the hierarchy of action groups and actions that comprise the Post-Install sequence. You can add action groups and actions to the Action List by clicking the Add Action button. Once an item is listed in the Action List, you can move items around in the hierarchy by dragging and dropping them or by clicking the arrow buttons. Right-clicking in the Action List shows a context menu that you can use to manage the action groups and actions.
Add Action	The Add Action button opens the Choose an Action dialog box, where you can choose to add action groups and actions.
Remove	The Remove button deletes the currently selected action, file, or directory from the Visual Tree.
Properties Tab	The Properties tab displays the customizers for the selected project element.
Rules Tab	Use the Rules tab to add rules to the selected project element.
Tags Tab	Use the Tags tab to add build configuration tags to the selected category or action. To learn more, see Assigning Tags to Project Elements.

Pre-Uninstall View

Use the Pre-Uninstall view on the Sequence page to defines the actions and panels that execute before the installer removes the product's files from target systems.

Uninstall View

InstallAnywhere automatically creates an uninstaller for projects. 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 view on the Sequence page to customize the uninstaller by adding, removing, or changing some of the uninstall actions.

The Uninstall view contains the following settings:

Table 9-64 ▪ Uninstall Settings

Setting	Description
Visual Tree	The Visual Tree displays the hierarchy of uninstall categories and actions that make up the uninstaller. By default, the following uninstall categories are available: • Files • LaunchAnywheres • Shortcuts/Links/Aliases • Registry Entries • Built-in Packages • Folders • Others Category By default, each of these uninstall categories contains one Uninstall action that would run the uninstallation of the category. Right-clicking in the Visual Tree shows a context menu that you can use to manage the files and actions.
Assign to	This setting lets you specify whether the items in the Visual Tree are displayed according to the product features with which they are associated, or according to the components with which they are associated. Depending on what you select in this list, the columns to the right of this setting display the names of features or components.
Add Action	Click the Add Action button to open the **Choose an action** dialog box, where you can select an action to add to the Visual Tree.
Remove	The Remove button deletes the currently selected category or action from the Visual Tree.
Properties Tab	The 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 the Rules tab 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 the Tags tab to add build configuration tags to the selected uninstall category or action. To learn more, see Assigning Tags to Project Elements.

Post-Uninstall View

Use the Post-Uninstall view on the Sequence page to schedule the actions that occur after the uninstaller has removed the product's files from target systems.

The Post-Uninstall view follows the same organization and has the same options as the Post-Install view. The differences between the views are determined solely by the time of their occurrence in the installation or uninstallation process.

Build Page

The Build page contains links to views that you can use to build an installer or a virtual appliance.

Table 9-65 ▪ Views on the Build Page

View	Description
Build Installers View	Provides the options for building an installer including creating and selecting build configurations, setting build targets, bundling VM packs, and defining distribution options.
Build Appliances View	Provides the options for building a virtual appliance. This includes creating and selecting appliance configurations, specifying the target hypervisor, adding installers, and defining other virtual appliance options.

Build Installers View

The Build Installers view on the Build page is where you configure settings for building an installer to multiple targets in multiple build configurations. The Build Installers view includes the following tabs:

Table 9-66 ▪ Build Installers View 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, JVMs, locales, and other settings.
Build Log Tab	Reports the results of the most recent build.

Build Configurations Tab

The Build Configurations tab in the Build Installers view 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 a particular set of locales, platforms, files, build distributions, JVMs, and other settings. You create and modify build configurations on the Build Configurations tab of the Build Installers view on the Build page. On this tab, you can add, edit, or delete a build configuration.

The Build Configurations tab contains the following subtabs:

- Build Targets Subtab
- Distribution Subtab
- Tags Subtab
- Locales Subtab

The Build Configurations tab includes the following controls:

Table 9-67 ▪ Build Configurations Tab

Control	Description
Select Build Configuration	Select a build configuration in this list to view and modify its settings as needed. In addition to new build configurations that you create, the following two values may be listed in the Select Build Configuration list: • **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 that are defined in the template are copied to the Default Configuration. • **Migrated Configuration**—Projects that are upgraded from earlier versions of InstallAnywhere have only one build configuration called *Migrated Configuration*, which has the same build settings that were found in the original project.
Add to project build	To include the selected build configuration in the project builds, select this check box. Each time that you click the Build Project button when this check box is selected, InstallAnywhere builds this build configuration.
Add Build Configuration	To create a new build configuration, click this button. The Add Configuration dialog box opens, prompting you to enter a name for the new build configuration.
Copy Build Configuration	To copy an existing build configuration, click this button. The Copy Configuration dialog box opens, prompting you to enter a name for the new build configuration.
Rename Build Configuration	To edit the name of an existing build configuration, click this button. The Rename Build Configuration dialog box opens, prompting you to enter a new name for the build configuration.
Remove Build Configuration	To delete the selected build configuration, click this button. InstallAnywhere displays a message box, prompting you to confirm that you want to delete the selected build configuration.
Add Build Target	Click to create a new build target. See Build Targets Subtab.

Table 9-67 ▪ Build Configurations Tab (cont.)

Control	Description
Build Output Location	For all of the build configurations in the project (regardless of how many are being built at the moment), there will be one subdirectory for each build configuration under the build output location. These subdirectories are named after the build configuration name. For migrated projects, the build output location of the migrated configuration is as follows: $IA_PROJECT_DIR$/$IA_PROJECT_NAME$_Build_Output
Open in Explorer	Opens a window to show the build output directory.
Build Project	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** check box selected), click this button.
Build Selected	To build the selected build configuration, click this button.
Build All	To build all build configurations that have been added to the project, click this button.
[Target Installer Selection List]	This list appears between the Build All and Try Installer buttons if, after the project has been built, more than one supported target installer is available for the authoring platform. If there are none or only one supported target installer, this list is hidden.  ***Note ▪*** *The name that is displayed for each target installer uses one of the following formats:* • **Without VM**—*OutputFolderName_BuildConfigurationName* • **With VM**—*OutputFolderName_BuildConfigurationName - BundledVMName*
Try Installer	To run the default installer for your operating system, click this button. 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) that you can try.  ***Note ▪*** *To assist in troubleshooting when running an installer on a Windows platform, it is possible to display debug output in a console window. To display debug output during installation, hold down the Control (Ctrl) key when clicking the Try Installer button.*
Try Web Install	To test the Web installation, click this button. InstallAnywhere launches the InstallAnywhere Web install page that is generated by the build process in a Web browser, enabling you to test the Web installer.

Build Targets Subtab

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.

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.

- OS X Build Target Customizer
- Windows and UNIX Platform Build Targets Customizer
- Other Java-Enabled Platforms Build Target Customizer

OS X Build Target Customizer

The OS X build target customizer includes controls to define a OS or OS X-based build target.

The OS X build target customizer includes the following controls:

Table 9-68 ▪ OS X 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 that the project builds. 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 _1 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 in the list.  ***Note ▪*** *If you change the value of this setting to another type, the controls that InstallAnywhere displays on the customizer change.*
Without VM	If you do not want to include a Java virtual machine with this build target, select this check box. The installer searches for Oracle JRE 7 or later. If it is not found on the target system, the installer exits.
With VM	If you want to include a Java virtual machine with this build target, select this check box. If you select this check box, you also need to select the VM that you want to bundle with your installer. If multiple required versions of JRE are present, the installer uses the one that is set by the JRE_HOME environment variable.

Table 9-68 ▪ OS X Build Target Customizer (cont.)

Control	Description
Delete	Delete this build target from the project.

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.

The Windows and UNIX platforms build target customizer includes the following controls:

Table 9-69 ▪ 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. • 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 _1 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 9-69 ▪ Windows and UNIX Platforms Build Target Customizer (cont.)

Control	Description
Without VM	If you do not want to include a Java virtual machine with this build target, select this check box. If you select this check box, you also need to select one of the following options to specify how the installer will find the VM it will use: • **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. Optionally, you can also select the Verify downloaded VM with MD5 hash and enter the MD5 hash (in lowercase 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. • **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. Optionally, you can also select the **Verify downloaded VM with MD5 hash** and enter the MD5 hash (in lowercase 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.  ***Important ▪*** *You can obtain VM packs by downloading them from the Revenera Web site (from the page that opens when you select Downloads > Download Additional VM Packs from the Tools menu). However, these VM packs are provided for your convenience only and should not be referenced by your installation project.* *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 http://www.flexerasoftware.com Web site. 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.*  ***Note ▪*** *On the Windows platform, when the installer or uninstaller is launched with administrative privileges, only the JVMs located in unsecured folders are considered invalid and are not used. This behavior can be turned off by using one of the following methods:* • *Configure the* `designer.jvm.securefolder.check` *property to* `false` *in the* `com.zerog.ia.Designer.properties` *file.*

Table 9-69 ▪ Windows and UNIX Platforms Build Target Customizer (cont.)

Control	Description
	• *Configure the **Secure JVM Folder Check** setting to **Disable** in the **Project** > **Platforms** >**Windows** area.* • *Run the installer or uninstaller using the `<installer/uninstaller_name> -Djvm.securefolder.check=<false>` command line, where `<installer/uninstaller_name>` refers to your installer or uninstaller executable.*
With VM	If you want to include a Java virtual machine with this build target, select this check box. If you select this check box, 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. • **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	If you select the With VM check box, select a bundled VM in this list.
JVM Search Settings	If the installer searches for a VM, it uses the JVM search instruction files that are specified in this list. • **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.  ***Note ▪** InstallAnywhere provides several pre-written JVM Specs, which are listed on the Choose JVM Spec dialog box. You can also write you own JVM spec. For more information, see JVM Spec Files.*
Windows Installer Launcher Type	(Windows only) Identify the type of launcher you want to use. Choose from Graphical Launcher and Console Launcher.  ***Note ▪** InstallAnywhere supports Arabic and Hebrew locales in GUI mode only. Console mode installers will not run under those locales.*
Delete	Click to delete the build target.



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. If any other protocol is used, the build fails.*

Other Java-Enabled Platforms Build Target Customizer

The Other Java-Enabled Platforms build target customizer includes controls to define a Java-enabled platform build target.

The Other Java-Enabled Platforms build target customizer includes the following controls:

Table 9-70 ▪ Other Java-Enabled 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 that the project builds. 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 _1 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 in the list.  ***Note*** ▪ *If you change the value of this setting to another type, the controls that InstallAnywhere displays on the customizer change.*
Without VM	If you do not want to include a Java virtual machine with this build target, select this check box.
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-ROMs. The Distribution subtab also enables you to configure settings for building merge modules and templates.

The Distribution subtab of the Build Configurations tab includes the following options:

Table 9-71 ▪ Distribution Subtab Controls

Control	Description
Build Web Installers (Self-Extracting Installers, Suitable for Web-based Distribution)	To build a Web installer, which is a single executable file that contains all of the necessary installation logic, select this check box. 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	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, select this check box.
Bundle JRE as Directory	To prevent the quarantine of the Java command line tool by the Gatekeeper in case of DMG distribution of macOS installers, select the **Bundle JRE as Directory** option. By default, this option is **unchecked**, which means that the `JRE` is bundled as `jre.zip`. If the **Bundle JRE as Directory** option is selected, then the `JRE` is bundled as a directory inside the macOS installer.  ***Note*** ▪ *This new feature is applicable to DMG distribution with VM installers to macOS operating system.*
Web page displays in	Select in which language to build the target web page.
Build CD-ROM Installers (Suitable for Distribution on Removable Media)	These installers consist of multiple files that are meant to be burned onto a CD, DVD, or multiple CDs or DVDs. They can also be placed on network volumes to provide easy access to large installers. The output of this build process can be directly burned onto disk. For more information, see Creating CD-ROM/DVD Installers.
Optimize Installer Size by Platform and Tags	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, select this check box.
Build Merge Module/Template	To integrate merge modules into an installer project at design time, select this check box.

Table 9-71 ▪ Distribution Subtab Controls (cont.)

Control	Description
Optimize Merge Module/ Template Size by Platform and Tags	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, select this check box.
Read Only (May not be imported, but may be used in an 'Install Merge Module' action)	To create a merge module that is read-only, meaning that it can be installed but not altered, select this check box. 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.

Tags Subtab

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.

The Tags subtab of the Build Configurations tab includes the following controls:

Table 9-72 ▪ Tags Subtab Controls

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 its check box is selected.

The Locales subtab includes the following controls:

Table 9-73 ▪ Locales Subtab Controls

Control	Description
Locale List	Lists all locale options. 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.
Default Locale for Targets with Unsupported Languages	Select the language to use if the target machine's operating system uses a language that is not supported by the installer. If the installer is started on a platform that uses an unsupported language, then the selected language will be used.  ***Note*** *▪ The default locale selection that you make has the following impact depending upon whether the installer is launched using the command-line locale (`-l`) option:* • *If the installer is launched without using the command-line locale option, the default locale would be selected as the language for installation in the list of supported locales.* • *If the installer is launched using the command-line locale option (`install.exe -l nn`) and if the locale specified in this command is not supported, then the installer immediately launches in the language of the default locale. For example, suppose an installer was built for the locales of English, Arabic, and Chinese, with Arabic set as the default locale. If the user attempts to launch this installer using the command-line locale option to specify an unsupported language (such as entering* **`install.exe -l ja`** *to attempt to launch the installer in Japanese), the installer would automatically start in Arabic (the default locale).*



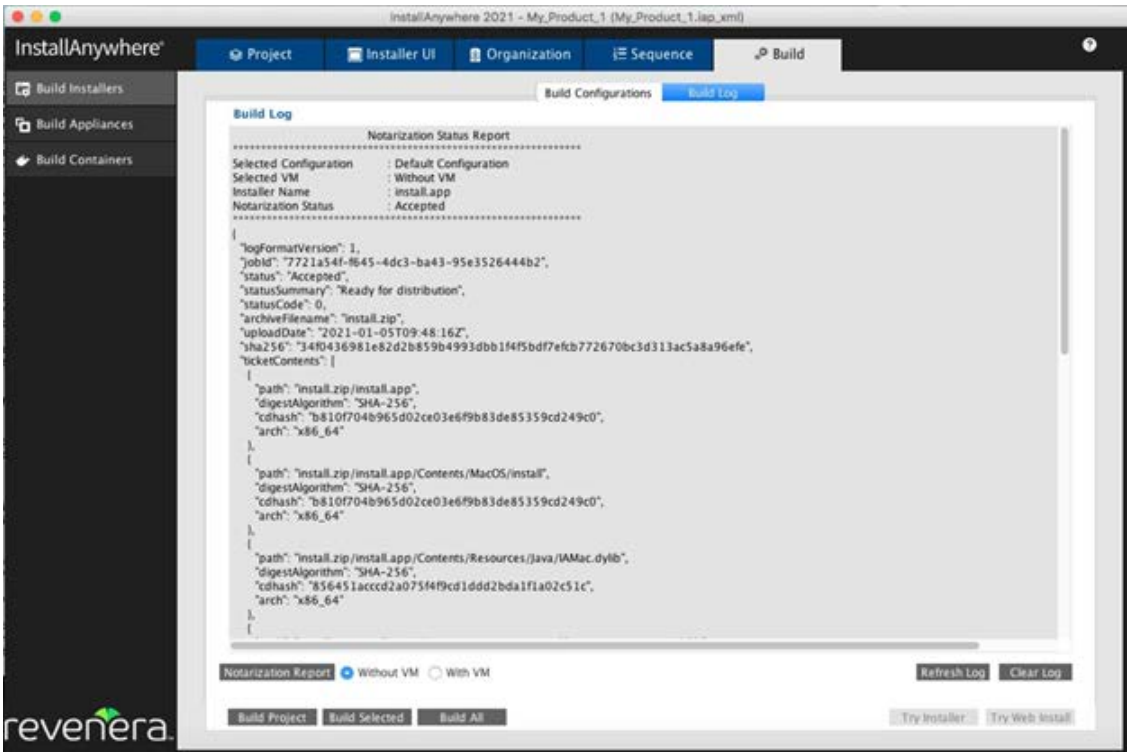
Note *▪ Avoid customizing the locale files of an InstallAnywhere project while that project is open in InstallAnywhere. InstallAnywhere may overwrite your manual changes to the locale file when you later save the open project.*

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. In the Build Log tab, you can view complete notarization report on Mac if notarization option is enabled while building the Mac Installer.

Task ***To view Notarization Report,***

1. In the Build Log tab, select Notarization Report option **Without VM** or **With VM** as required.
2. Click **Notarization Report** button.
   Based on the installer configured, Notarization report is displayed:



Note ▪ *This feature is specific to MAC version of InstallAnywhere. In Windows and Linux system, this option is disabled.*

Build Appliances View

The Build Appliances view on the Build page provides the options for building virtual appliances for multiple target hypervisors in multiple appliance configurations. The Build Appliances view includes the following tabs:

Table 9-74 ▪ Build Appliances View Tabs

Tab	Description
Appliance Configuration Tab	Create and modify appliance configurations, each of which defines how the appliance will be built for a particular target hypervisor.

Table 9-74 ▪ Build Appliances View Tabs (cont.)

Tab	Description
VM Configuration Tab	Select a VM template, add installers, specify basic hardware settings, select OS packages to install, set up a local OS repository, and specify custom scripts to run.
Manage VM Tiers Tab	Configure settings for a multi-tier virtual appliance.
Build Log Tab	View the results of the most recent build.

Appliance Configuration Tab

On the Appliance Configuration tab of the Build Appliances view on the Build page, you can create and modify appliance configurations, each of which defines how the appliance is built for a particular target hypervisor.

The Appliance Configuration tab includes the following options.

Table 9-75 ▪ Appliance Configuration Tab

Option	Description
Select Appliance Configuration	Select an appliance configuration from this list and then perform one of the following actions: ● Select the Add to project build option and then select Save on the File menu to add the selected appliance configuration to the project build. Then, each time you click the Build Appliance button, this appliance configuration will be built. ● Edit the options (described below) on this tab and then select Save on the File menu. ● Click Add to create a new appliance configuration. ● Click Copy to copy the selected appliance configuration using a new name. ● Click Rename to edit the name of the selected appliance configuration. ● Click Remove to delete the selected appliance configuration. ● Click Build Selected to build just the selected appliance configuration. **Default Appliance Configuration** When a new InstallAnywhere project is created, from any of the templates, it is assigned the default appliance configuration, which is named `Default_Appliance_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 appliance configuration.

Table 9-75 ▪ Appliance Configuration Tab (cont.)

Option	Description
Add to project build	Select this option to include the selected appliance configuration to the project build. Then, each time you click the Build Appliance button, this appliance configuration will be built. When a new project is created from any of the templates, it is assigned the default appliance configuration, which is named `Default_Appliance_Configuration`. The **Add to project build** option will be enabled automatically for this configuration.
Add Appliance Configuration	To create a new appliance configuration, click this button. The Add Configuration dialog box opens, prompting you to enter a name for the new appliance configuration.
Copy Appliance Configuration	To copy an existing appliance configuration, click this button. The Copy Configuration dialog box opens, prompting you to enter a name for the new appliance configuration.
Rename Appliance Configuration	To edit the name of an existing appliance configuration, click this button. The Rename Appliance Configuration dialog box opens, prompting you to enter a new name for the appliance configuration.
Remove Appliance Configuration	To delete the selected appliance configuration, click this button. InstallAnywhere displays a message box, prompting you to confirm that you want to delete the selected appliance configuration.
Appliance Version	Enter a version number to identify the version of the virtual appliance that you will build with the selected appliance configuration.
Appliance UUID	When you add a new appliance configuration, an alphanumeric string is automatically entered to uniquely identify this appliance. Click Generate UUID to generate a new ID.
EULA	Click Choose EULA File and select a text (`.txt`) file that contains the end user license agreement information for the virtual appliance. Your customers will be prompted to accept this EULA before they will be permitted to use the virtual appliance.  ***Note ▪*** *The EULA must be a text-only file and will not support markup of any kind.*  ***Note ▪*** *Specifying a EULA is mandatory for the creation of a virtual appliance.*
Description	Enter text to describe this virtual appliance.

Table 9-75 ▪ Appliance Configuration Tab (cont.)

Option	Description

Appliance URL	In the Appliance URL field, enter the start URL by which the appliance can be accessed post deployment. Enter the Appliance URL using the following format:
	`<protocol>#hostname#<context_root>`
	where:
	● `<protocol>` is of type `http://`
	● `#hostname#` will be replaced by the IP of the appliance obtained after deployment of the virtual appliance
	● `<context_root>` is the context root of the virtual appliance
	A typical virtual appliance URL can be specified by the appliance author in the InstallAnywhere GUI as:
	`http://#hostname#/sample/demo.html`
	If the IP of the deployed appliance is `x.y.a.b`, then this URL will be resolved as:
	`http://x.y.a.b/sample/demo.html`
	and will be presented as the start URL of the appliance to the end user.
	
	Note ▪ *For more information, see Specifying the Appliance URL.*

Target Hypervisor	Select one of the following options to identify the target hypervisor of this virtual appliance: ● Amazon EC2 ● VMware vSphere 5
Credentials	Select the set of defined credentials to identify and connect to the target hypervisor. To create a new set of credentials, click Add New Credential to open the Add New/Manage Connection Dialog Box. To edit an existing set of credentials, select the credentials from the list and click Manage Credential to open the Add New/Manage Connection Dialog Box.
Deploy this Appliance	To enable automatic deployment of this virtual appliance to the selected target hypervisor at build time, select this check box.

Table 9-75 ▪ Appliance Configuration Tab (cont.)

Option	Description
OVF Appliance Type	Select one of the following options to define the OVF appliance type: ● **VMware vCenter compatible appliance**—Appliance is deployed in OVF 1.1 format on a licensed VMware vSphere server which is managed by a licensed VMware vCenter. This appliance will be deployed as a pure virtual appliance and will have its own resource pool. Such virtual appliances are also capable of having multiple virtual machines under the appliance. InstallAnywhere currently supports only a single-VM virtual appliance. ● **VMware vSphere compatible appliance**—Appliance is also deployed in OVF 1.1 format on a licensed or free version of VMware vSphere, which may or may not be managed by a VMware vCenter. This appliance will be deployed as a single virtual machine.  ***Note ▪*** *Only displayed if VMware vSphere 5 is selected in the Target Hypervisor list.*
Appliance output options	Select one of the following options: ● **OVF (Open Virtualization Format) 1.1**—Contains the VM disk files and the OVF descriptor in a single directory. ● **OVA (OVF Archive)**—The OVF 1.1 format output is archived as a `tar` file, which enables easier distribution of the virtual appliance. ● **Both**—The virtual appliance output is provided in both OVF 1.1 and OVA formats. This option will consume a larger amount of disk space.  ***Note ▪*** *Only displayed if VMware vSphere 5 is selected in the Target Hypervisor list.*

Table 9-75 ▪ Appliance Configuration Tab (cont.)

Option	Description
Root Device Type	The AMI image used to launch Amazon EC2 instances is stored in the root device volumes. To specify the type of disk storage that your AMI image will be stored in, select one of the following root device storage-backed instance types: • **EC2 instance store backed**—Temporary block level storage (also known as an "instance store" volume). An instance store is dedicated to a particular instance and the data on it persists only during the lifetime of its associated EC2 instance. Data on such instance store volumes are lost when the instances are stopped, fail or are terminated and cannot be restored. It is ideal to use such instance-store-backed instances for temporary storage of information or content. • **Elastic block store backed**—Reliable storage volumes that can be attached to a running instance in the same availability zone. EBS volumes attached to an instance persist independently from the life of the instance and are recommended to be used when data changes frequently and requires long term persistence.  ***Note ▪*** *For more information, see Support for Elastic Block Store (EBS) for Amazon.* *Only displayed if Amazon EC2 is selected in the Target Hypervisor list.*

Table 9-75 ▪ Appliance Configuration Tab (cont.)

Option	Description
Select the region for deployment	Select one of the following regions to upload and run your virtual appliance: ● US East (Northern Virginia) Region ● US West (Northern California) Region ● US West (Oregon) Region ● South America (Sao Paulo) Region ● EU (Ireland) Region ● Asia Pacific (Tokyo) Region ● Asia Pacific (Singapore) Region  ***Note ▪*** *It is always beneficial to deploy the AMI to the region closest to your customer base. For example, if your business runs in South America only, then it is better to base your deployed AMI in the Sao Paulo region. This will result in faster response times. Also, both the S3 bucket and AMI instance would be deployed to the corresponding region, which would help customers reduce costs.* You can create an S3 bucket in any of these regions. Amazon guarantees that the S3 bucket will be created in the respective region and that the user-uploaded files will be stored in the specified region. The same region specification applies to EC2 as well. Launching and running of EC2 instances can be done in any of these regions.  ***Note ▪*** *For more information, see About EC2 and S3 Regions of Deployment for Amazon and Deploying an Amazon Virtual Appliance Through the Amazon Web Services (AWS) Management Console.* *Only displayed if Amazon EC2 is selected in the Target Hypervisor list.*
EBS Volume size (GiB)	Specify the size in gigabytes of your EBS volume. The limit is up to 1 TiB depending on your AWS account limits.  ***Note ▪*** *Only displayed if Amazon EC2 is selected in the Target Hypervisor list. Field is enabled when Elastic block store backed is selected under Root Device Type.* *See also Support for Elastic Block Store (EBS) for Amazon.*

Table 9-75 ▪ Appliance Configuration Tab (cont.)

Option	Description
Device	Enter the name of the root device that the volumes will be attached to. For example, the recommended connection is: `/dev/sdf` through `/dev/sdp`  ***Note ▪*** *For more information, see Support for Elastic Block Store (EBS) for Amazon.* *Only displayed if Amazon EC2 is selected in the Target Hypervisor list. Field is enabled when* ***Elastic block store backed*** *is selected under Root Device Type.*
Image Name	Enter the name of the EBS backed image created from the EBS snapshot.  ***Note ▪*** *For more information, see Support for Elastic Block Store (EBS) for Amazon.* *Only displayed if Amazon EC2 is selected in the Target Hypervisor list. Field is enabled when* ***Elastic block store backed*** *is selected under Root Device Type.*
Same as the build credentials	To deploy the virtual appliance to the same machine on which you are building the virtual appliance, select this check box. To deploy the virtual appliance to a different server—not the one on which you are building the virtual appliance—clear this check box, and then select the appropriate machine in the box under this check box. The box contains all of the machines that are defined in the Credentials list.  ***Note ▪*** *This functionality is available if VMware vSphere 5 is selected in the Target Hypervisor setting.*
Appliance Output Location	After the virtual appliance is built, the output location is listed here. Click Open in Explorer to open the containing folder.
Build Appliance	Click to initiate the building of virtual appliances based on all of the appliance 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 appliance configuration.
Build All	Click to build all appliance configurations that have been added to the project.

VM Configuration Tab

On the VM Configuration tab of the Build Appliances view on the Build page, you specify the VM that is installed with this virtual appliance.

Table 9-76 ▪ VM Configuration Tab

Property	Description
Select VM Tier	 ***Note ▪*** *This setting is available if VMware vSphere 5 is selected in the Target Hypervisor setting for the selected appliance configuration.* Select a VM tier in this list to view and modify its settings as needed.
Add VM Tier	To create a new VM tier, click this button. The Add VM Tier dialog box opens, prompting you to enter a name for the new VM tier.
Copy VM Tier	To copy an existing VM tier, click this button. The Copy VM Tier dialog box opens, prompting you to enter a name for the new VM tier.
Rename VM Tier	To edit the name of an existing VM tier, click this button. The Rename VM Tier dialog box opens, prompting you to enter a new name for the VM tier.
Remove VM Tier	To delete the selected VM tier, click this button. InstallAnywhere displays a message box, prompting you to confirm that you want to delete the selected VM tier.
Name of VM	Enter a name to identify the VM that will be used in this virtual appliance.
VM Version	Enter the version of the specified VM.
UUID of VM	An alphanumeric string is automatically entered to uniquely identify this virtual appliance's VM. Click Generate UUID to generate a new ID.
Select OS	Select the operating system of the VM.
Select InstallAnywhere VM Template	 ***Note ▪*** *This setting is available if Amazon EC2 is selected in the Target Hypervisor setting for the selected appliance configuration.* Select a VM template to use to build this virtual appliance. For information on obtaining and configuring VM templates, see Obtaining VM Templates for Virtual Appliances.

Table 9-76 ▪ VM Configuration Tab (cont.)

Property	Description
Chosen VM	 ***Note*** ▪ *This setting is available if VMware vSphere 5 is selected in the Target Hypervisor setting for the selected appliance configuration.* This read-only setting identifies the VM template that will be used for the selected VM tier. To identify the VM, click the Choose a VM button next to this setting.
JVM to Bundle	 ***Note*** ▪ *This setting is available if VMware vSphere 5 is selected in the Target Hypervisor setting for the selected appliance configuration.* Specify the JVM that you want to bundle with the virtual appliance.

On the subtabs of the VM Configuration tab, you can define the virtual appliance installer properties, the virtual appliance hardware requirements, the operating systems to install, the installers to associate with the virtual appliance, the boot and login scripts to run, and the repository settings.

- Product Properties Subtab
- Hardware Subtab
- OS Packages Subtab
- Installers Subtab
- Script Info Subtab
- Repository Settings Subtab

Product Properties Subtab

The Product Properties subtab on the VM Configuration is where you view and define properties for the installers (which are identified on the Installers subtab) that you are adding to the VM for your virtual appliance. InstallAnywhere uses the default values that you set here when running the installers on the VM.

The following settings are available on the Product Properties subtab of the VM Configuration tab.

Table 9-77 ▪ Product Properties Subtab

Setting	Description
Product Name	Name of product installer. The default value is the value that was entered in the Installer Title setting in the General Settings view on the Project page.
Product Version	Product version.
Product Vendor	Vendor of product.

Table 9-77 ▪ Product Properties Subtab (cont.)

Setting	Description
Product Description	Describe purpose of product.
Product Properties	To view and edit this product's properties, click the View/Edit button in this column. The Edit Installer Properties Table dialog box opens, enabling you to manage properties (or response options) that need to be supplied to the installer in order for it to silently install on the virtual appliance. When you add an installer to the virtual appliance project, InstallAnywhere automatically discovers these properties. For more information, see Edit Installer Properties Table Dialog Box. If you are creating a multi-tier virtual appliance for VMware vSphere, ensure that you define the properties that each tier in your virtual appliance needs in order to connect to each other. For example, you may need to configure the IP address, port number, user name, and password of your database tier so that your application tier can connect to it.

Hardware Subtab

Instead of offering different editions (such as Basic, Deluxe, and Premier) of your virtual appliance with different sets of functionality for each edition, you may want to offer virtual appliances with varying sets of hardware requirements or compute capacity.

Use the Hardware subtab of the VM Configuration tab to specify the hardware requirements for the selected target hypervisor. The options that are listed on this subtab depend on which target hypervisor is selected on the Appliance Configuration tab in the Build Appliances view:

- Amazon EC2 Target Hypervisor Requirements
- VMware vSphere 5 Target Hypervisor Hardware Requirements

Amazon EC2 Target Hypervisor Requirements

Amazon EC2 provides the flexibility to choose from a number of different instance types to meet your computing needs. Each instance provides a predictable amount of dedicated compute capacity and is charged per instance-hour that is consumed. When you are configuring an Amazon EC2 virtual appliance, you are prompted to select one of the following instances types on the Hardware subtab of the VM Configuration tab of the Build Appliances view:

- Small Instance
- High CPU Medium Instance
- Medium Instance

The hardware requirements for each of these instance types is constantly being adjusted and updated by Amazon. For the most up-to-date information, see http://aws.amazon.com/ec2/instance-types/.

VMware vSphere 5 Target Hypervisor Hardware Requirements

When VMware vSphere 5 is selected in the Target Hypervisor list on the Appliance Configuration tab, and you open the Hardware subtab of the VM Configuration tab, you are prompted to specify the hardware requirements for the VMware vSphere 5 VM.

Table 9-78 ▪ Hardware Subtab / VMware vSphere 5

Property	Description
Number of CPUs	Minimum number of CPUs that the machine is required to have. The default value is 1 CPU.
Hard Disk Size (in GB)	Minimum amount of free space required to be available on the machine's hard drive. The default value is 4 GB.  ***Note*** ▪ *InstallAnywhere will create the SCSI controller for the hard drive.*  ***Note*** ▪ *The hard disk size needs to be greater than or equal to the size of the Baseline VM selected for the project.*
RAM Size (in MB)	Minimum amount RAM required to be installed on the machine. The default value is 1024 MB.
Number of CDROMs	Minimum number of CDROM drives that are required on the machine. The default value is 0.  ***Note*** ▪ *InstallAnywhere will create the IDE controllers for the CDROMs.*
Number of Ethernet Adapters	Minimum number of Ethernet adapters that the machine is required to have. The default value is 1.  ***Note*** ▪ *The network adapters are all set to default PCNET32 type.*

OS Packages Subtab

VM Templates created by InstallAnywhere include several operating system packages, such as `SSH`, `wget`, `perl`, and `openssl`. These packages are used by the virtual appliance creation process.

However, there might be a need to install additional operating system packages for your specific virtual appliance to work. On the OS Packages subtab of the VM Configuration tab, you can choose additional operating system packages to be installed during the creation of the virtual appliance.

The OS Packages subtab includes the following options:

Table 9-79 ▪ OS Packages Subtab

Option	Description
Packages Selected to Install	Operating system packages in this list will be installed during the creation of the virtual appliance. To clear the selection of one of these operating system packages, select it in this list and click the Remove button to move it to the Packages Available to Install list.
Packages Available to Install	This is a list of all of the available operating system packages. To select one of these operating system packages to be installed during the creation of this virtual appliance, select it in this list and click the Add button to move it to the Packages Selected to Install list.



Note ▪ *For detailed instructions on how to add a standalone RPM or Debian OS package to the OS Packages tab, see Using RPM/Debian Packages for Proprietary Operating Systems on Virtual Appliances.*

Installers Subtab

Adding installers is the crux of the virtual appliance build process. You can add a prebuilt InstallAnywhere installer, an InstallAnywhere installer out of the current project, or other installers. You can also specify a response file, if you have one. Use the Installers subtab of the VM Configuration tab in the Build Appliances view to add installers to an appliance configuration.

Once you add an installer to the Installers subtab, InstallAnywhere adds a default entry that corresponds with the installer on the Product Properties subtab of the VM Configuration tab. This tab keeps track of all installers (which are called Products in OVF terminology) that are installed onto your appliance.



Note ▪ *All installers that you specify must be capable of unattended silent installations; otherwise the appliance build might stop at the point these installers prompt for user interaction.*

The Installers subtab of the VM Configuration tab includes the following options:

Table 9-80 ▪ Installers Subtab

Option	Description
Build Time	Select this tab and click Add Installer to add an installer that will be run during build time. Build time installers are mostly used to push payload for the software appliance.
First Boot Time	Select this tab and click Add Installer to add an installer that will be run the first time the virtual appliance is booted. You would add an installer to this tab if there is a need for some configuration to be performed prior to installation.

Table 9-80 ▪ Installers Subtab (cont.)

Option	Description
Add Installer	Click to open the Add Installer/Edit Installer Dialog Box where you can add an installer.
Remove Installer	Click to remove the selected installer from the appliance configuration.
Edit Installer	Click to open the Add Installer/Edit Installer Dialog Box where you can edit an existing installer.
Up **Down**	Use to move the installers up or down in the list. Installers listed first will be installed first.



Important ▪ *The First Boot Time/Build Time installers are not capable of detecting and installing* `InstallTimeMerge` *modules for virtual appliances.*



Note ▪ *For detailed instructions on how to add a standalone RPM or Debian OS package to the Installers tab, see Using RPM/Debian Packages for Proprietary Operating Systems on Virtual Appliances.*

Script Info Subtab

Virtual appliances built by InstallAnywhere have the capability of automatically calling scripts at specific lifecycle stages of your appliance. You have the option of running custom shell scripts at First Boot, First Login, Subsequent Boot, and Subsequent Login of the virtual appliance.

For example, if you have installed an Apache web server in your virtual appliance, and this needs to be started every time your appliance boots up, then you need to define a small shell script that will start your Apache web server each time it boots up.



Note ▪ *InstallAnywhere appliances also run some internal scripts at each of these lifecycle stages. These internal scripts help safeguard the security of your appliances by regenerating the different SSL keystores and helping the end user set up a default password for the virtual appliance during deployment.*

Because user interaction is not supported during boot time, custom boot scripts (first boot and subsequent boot) should not have any sections that have user interaction. Also, the display of output user messages from the first/subsequent boot are not visible on the console during boot time.

Custom boot scripts (first boot and subsequent boot) support only shell script.

The Script Info subtab of the VM Configuration tab includes the following options:

Table 9-81 ▪ Script Info Subtab

Option	Description
First Boot Script	Click Choose and select a script that will run the first time the virtual appliance is launched.
First Login Script	Click Choose and select a script that will run the first time a user logs into this virtual appliance.
Subsequent Boot Script	Click Choose and select a script that will run each time the virtual appliance is launched.
Subsequent Login Script	Click Choose and select a script that will run each time a user logs into this virtual appliance.

Repository Settings Subtab

The virtual appliance creation process requires that the build system be connected to the Internet. However, in some cases, because of security reasons or low Internet bandwidth, such a connection might not be possible. In these cases, you can choose to set up an OS repository of OS packages and OS-related components on the build machine, and have InstallAnywhere use this repository at build time. The VM Configuration tab in the Build Appliances view on the Build page is where you specify whether you want to use Internet-based repositories or local repositories. Note that the repositories need to be accessible to the virtual appliance build system in order to be able to resolve the IP addresses and install the OS packages.

The Repository Settings subtab includes the following options:

Table 9-82 ▪ Repository Settings Subtab

Option	Description
Internet-Based Repositories Provided by OS Vendors	Select this option to install OS packages and OS related components from Internet-based repositories provided by OS vendors.
Local Repositories	Select this option to specify that you want to install OS packages and OS related components from a local repository.

Table 9-82 ▪ Repository Settings Subtab (cont.)

Option	Description
Location of Local Repository Specification File	Click Choose and select the location of your local OS repository specification file. ● The specification file is of the same format as `/etc/apt/sources.list` file in a standard Ubuntu installation or `/etc/yum.repos.d/Centos-base.repo` in the case of a CentOS installation. ● As part of the specification, you can also point to multiple repositories including repositories that include your proprietary packages in the specification file. ● For baseline VM creation, the repository settings will serve as an advanced option which can be pointed to by using the `Baseline.vm.local.repository.location` inside the appropriate property files.

Manage VM Tiers Tab



Note ▪ *The Manage VM Tiers tab is available if VMware vSphere 5 is selected in the Target Hypervisor setting on the Appliance Configuration tab (Build page > Build Appliances view).*

The Manage VM Tiers tab in the Build Appliances view on the Build page is where you configure settings for a multi-tier virtual appliance.

The following settings are available on the Manage VM Tiers tab.

Table 9-83 ▪ Manage VM Tiers Tab Settings

Setting	Description
VM Tier Name	This read-only column lists the name of each VM tier.
Start/Stop Order	In this column, specify the order in which the VM tiers should be powered on and powered off. For example, a tier that has a Start/Stop Order value of 0 is powered on before a tier that has a Start/Stop Order of 3. The tier with the Start/Stop Order value of 3 is powered off before the tier with the Start/Stop Order value of 0.
Start Delay (in seconds)	Specify the number of seconds before each VM tier in the virtual appliance should be started.
Stop Delay (in seconds)	Specify the number of seconds before each VM tier in the virtual appliance should be stopped.

Build Log Tab

The Build Log tab displays an XML log of the build once an appliance is successfully built. Click Refresh Log to display the current log. Click Delete Log to remove the log.

Troubleshooting “Out of Memory” Error Upon InstallAnywhere Launch or Command Line Build

Launching the InstallAnywhere Advanced Designer will sometimes lead to an error condition called `OutOfMemory`, which causes the user interface to freeze, and you are unable to proceed further in loading or building a project. This error could also occur when building a project via command line using the `build.exe` or `build-as-invoker.exe` commands

This error occurs when the repository of all kinds of objects in Java, which is called *heap space*, is running low on memory. It could also be caused by a failure to initialize the Java VM because not enough memory was acquired for the object heap. In order to have better performance and avoid such errors, it is advisable to set the initial and maximum JVM heap size for the InstallAnywhere Advanced Designer and for the build command line.

- Setting Initial and Maximum JVM Heap Size for the Advanced Designer
- Setting Initial and Maximum JVM Heap Size for Building via Command Line

Setting Initial and Maximum JVM Heap Size for the Advanced Designer

To set the initial and maximum JVM heap size for the advanced designer, perform the following steps.

Task ***To set the initial and maximum JVM heap size for the Advanced Designer:***

1. Locate the `InstallAnywhere.lax` file in the InstallAnywhere installation directory.
2. Open the file in a text editor and search for the following section:

```
#   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


#   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
```

3. Assign values to these properties in bytes. The example below allocates 536870912 bytes of memory to the initial and maximum heap sizes:

```
lax.nl.java.option.java.heap.size.initial=536870912
lax.nl.java.option.java.heap.size.max=536870912
```

   Ideally, you should enter values in multiples of 1024 (more than 1 mb). For more information on selecting a heap size, you may want to consult Oracle Java documentation.

   Note ▪ *For the Advanced Designer, you must specify heap sizes in bytes. Attempting to specify heap sizes in a different unit of measure by appending* `m`, `k`, *or* `g` *will have no effect.*

4. Save the `InstallAnywhere.lax` file.

5. Close and reopen the InstallAnywhere Advanced Designer.

Setting Initial and Maximum JVM Heap Size for Building via Command Line

To set the initial and maximum JVM heap size for the `build.exe` and `build-as-invoker.exe` commands, perform the following steps.



Task ***To set the initial and maximum JVM heap size for build commands:***

1. Locate the `Build.lax` file in the InstallAnywhere installation directory.
2. Open the file in a text editor and search for the following section:

```
#   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

#   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
```

3. Assign values to these properties.

   The example below allocates 536870912 bytes of memory to the initial and maximum heap sizes:

```
lax.nl.java.option.java.heap.size.initial=536870912
lax.nl.java.option.java.heap.size.max=536870912
```

   The default size for these values is measured in bytes. Append the letter **`k`** or **`K`** to the value to indicate kilobytes, **`m`** or **`M`** to indicate megabytes, and **`g`** or **`G`** to indicate gigabytes. The example below allocates 512 megabytes of memory to the initial and maximum heap sizes

```
lax.nl.java.option.java.heap.size.initial=512m
lax.nl.java.option.java.heap.size.max=512m
```

4. Save the `Build.lax` file.

Wizard Reference

Each wizard that is available in InstallAnywhere is described in this section.

- Choose a VM Wizard
- Create InstallAnywhere VM Template Wizard

Choose a VM Wizard

The Choose a VM Wizard lets you specify which virtual machine template you want to use for your virtual appliance. This wizard is available if VMware vSphere 5 is the target hypervisor. (That is, VMware vSphere 5 is selected in the Target Hypervisor setting on the Appliance Configuration tab on the Build Appliances view in the Build view.)



Task ***To launch the Choose a VM Wizard:***

5. In the Advanced Designer, on the **Build** page, click **Build Appliances**. The **Build Appliances** view opens.
6. Open the **VM Configuration** tab.
7. Click the **Choose a VM** button, which is next to the Chosen VM list.

The Choose a VM Wizard consists of the following panels:

- Select a Virtual Machine/Template Panel
- Summary of Selected VM Panel

Select a Virtual Machine/Template Panel

Use the Select a Virtual Machine/Template panel to indicate which virtual machine, snapshot, or template you want to use for your VMware vSphere/vCenter virtual appliance.The settings that are available on this panel are:

Table 9-84 ▪ Select a Virtual Machine/Template Panel Settings

Setting	Description
An InstallAnywhere VM template	If you want to use a VM template for your virtual appliance, select this option. To learn more, see Obtaining VM Templates for Virtual Appliances.
A VM running on an existing VMware vSphere/vCenter 5 server	If you want to use a VM that is running on an existing VMware vSphere 5 or VMware vCenter 5 server for your virtual appliance, select this option. Note the following requirements for the VM: • SSH should be configured and running on this machine. • The SSH credentials that are specified should have root credentials on this machine. • VMware Tools should already be installed on this VM.

Table 9-84 ▪ Select a Virtual Machine/Template Panel Settings (cont.)

Setting	Description
A VM snapshot running on an existing VMware vSphere/vCenter 5 server	If you want to use a VM snapshot that is running on an existing VMware vSphere 5 or VMware vCenter 5 server for your virtual appliance, select this option. Note the following requirements for the VM snapshot: • SSH should be configured and running on this machine. • The SSH credentials that are specified should have root credentials on this machine. • VMware Tools should already be installed on this VM snapshot.
Select InstallAnywhere VM Template	This setting is available if you select the **An InstallAnywhere VM template** option. This setting lists the VM templates that are available on your machine. To learn more, see Obtaining VM Templates for Virtual Appliances.
Selected VM from Tree	This setting is available if you select the **A VM running on an existing VMware vSphere/vCenter 5 server** option or the **A VM snapshot running on an existing VMware vSphere/vCenter 5 server** option. This setting shows the VM or VM snapshot that is selected in the list of vSphere/vCenter servers and their corresponding VMs in the box below this setting. The VMs in the box below this setting are the VMs that are available on the server that is specified in the Connection Settings area of the project (Build page > Build Appliances view > Appliance Configuration tab).

Summary of Selected VM Panel

Use the Summary of Selected VM panel to view settings for the selected existing VM or VM snapshot and specify a password. The settings that are available on this panel are:

Table 9-85 ▪ Summary of Selected VM Panel Settings

Setting	Description
VM Name	This read-only setting shows the name of the VM.
Virtual Appliance Name	This read-only setting shows the name of the virtual appliance.
Provide Root Password of VM	Enter the root password for the VM.
Power on VM if it is not already running	If you want InstallAnywhere to turn on the VM if it is not already running, select this check box.

Table 9-85 ▪ Summary of Selected VM Panel Settings (cont.)

Setting	Description
Validate VM to Proceed	Click this button to have InstallAnywhere validate the following: • The credentials that were specified enable the SSH connection to occur. • VMWare Tools is present. • The VM is compatible with the option that is selected in the Select OS setting on the VM Configuration tab (Build page > Build Appliances view).

Create InstallAnywhere VM Template Wizard

You can create a VM Template for either VMware vCenter/vSphere Server or AmazonEC2 using the Create InstallAnywhere VM Template Wizard.

Launching the Wizard

The Create InstallAnywhere VM Template Wizard executable found in the following directory:

`$IA_HOME$/VM_Template_CreationUtility`

To launch the Create InstallAnywhere VM Template Wizard, double-click one of the following files:

Table 9-86 ▪ Create InstallAnywhere VM Template Wizard Executable Files

User	Executable
User with administrative privileges on the build system	`CreateVMTemplate.exe`
User without administrative privileges on the build system	`CreateVMTemplate-AsInvoker.exe`

You can also launch the wizard from within the Advanced Designer.



Task ***To launch the Create InstallAnywhere VM Template Wizard:***

On the **Tools** menu, click **Wizards**, and then click **Create VM Template**.

Wizard Panels

The Create InstallAnywhere VM Template Wizard consists of the following panels:

- InstallAnywhere VM Template Information Panel
- VMware OS Information Panel
- Amazon OS Information Panel
- Review EULA for Creating the VM Template Panel
- VMware vCenter/vSphere 5 Information Panel

- Review Summary Information Panel
- Creating the Baseline VM Panel
- Finish Panel

InstallAnywhere VM Template Information Panel

On the InstallAnywhere VM Template Information panel, you specify the type of VM template you are creating, select the target platform, and enter a name for the template.

The InstallAnywhere VM Template Information panel includes the following properties:

Table 9-87 ▪ InstallAnywhere VM Template Information Panel

Property	Description
Create an InstallAnywhere VM Template for	Select the type of VM template that you are creating. Available options are: ● VMware vCenter/vSphere Server ● Amazon EC2
Choose Target Platform	Select the operating system that you want to use on the selected hypervisor.
Guest OS Architecture	Select the architecture of the operating system.
Name of VM Template	Enter a name to identify this new VM template.

Click Next to proceed.

VMware OS Information Panel

On the VMware OS Information panel, which opens if you selected VMware vCenter/vSphere Server on the InstallAnywhere VM Template Information panel, you specify the operating system media, VMware Tools location, and host machine details of the operating system that you are using to create this VM template.

The VMware OS Information panel includes the following settings:

Table 9-88 ▪ VMware OS Information Panel

Property	Description
OS Media ISO Location	Location of an ISO file of the operating system for which a VM template is being created. For more information, see Supported Hypervisors and Platforms for Virtual Appliances.  ***Note*** ▪ *To create a VM template, it is preferable to use a minimal OS / JeOS media from the OS vendor. This will help keep the footprint of the appliance to a minimum.* *Note that the server flavor of the operating system media is required.* *Certain Ubuntu Desktop ISO media are not capable of silent installation and require GUI intervention and hence is not recommended for use.*  ***Important*** ▪ *It is preferable that the OS media be obtained from the OS vendor Web sites so that you can be sure of the authenticity of the source. The ISO files for these operating systems are available at the following sites:* *http://www.ubuntu.com/download/server* *http://www.centos.org/*
VMware Tools Location	Location of the VMware Tools, which have to be in a `*.tar.gz` format.  ***Note*** ▪ *For information on obtaining VMware Tools, see Creating VM Templates Using the Create InstallAnywhere VM Template Wizard.*
Host Name	Enter the IP or the hostname of a machine that is running the same operating system that you are using to create the VM template.  ***Note*** ▪ *You should specify a host machine that is similar to your VM template target platform. In other words, if you require a VM template for Ubuntu, you need to specify an Ubuntu machine as the host machine.*
User Name **Password**	Enter the root login credentials to the specified host machine.
Test Connection	Click this button to: • Test whether the machine is accessible over the network. • Test whether an SSL connection can be made on port 22.

Click Next to proceed.

Amazon OS Information Panel

On the Amazon OS Information panel, which opens if you selected Amazon EC2 on the InstallAnywhere VM Template Information panel, you specify the operating system media, cloud image location, guest OS name, and host machine details of the operating system that you are using to create this VM template.

The Amazon OS Information panel includes the following properties:

Table 9-89 ▪ Amazon OS Information

Property	Description

Operating System Media	Select one of the following options:
	● **Create the baseline VM using the OS Vendor provided cloud image**—Select this option to create the VM template based on a cloud image provided by Ubuntu under the UEC images:
	`http://cloud-images.ubuntu.com/oneiric/`
	`http://cloud-images.ubuntu.com/precise/`
	● **Create the baseline VM using the specified host machine**—Select this option to have InstallAnywhere create an InstallAnywhere-compatible baseline VM template using a specified host machine (a Debian base system that InstallAnywhere creates from scratch).

Guest OS Name	Select the appropriate option.
Host Name	Enter the IP or the hostname of a machine that is running the same operating system (Ubuntu or CentOS) that you are using to create the VM template.
User Name **Password**	Enter the root login credentials to the specified host machine.
Test Connection	Click this button to do the following: ● Test whether the machine is accessible over the network. ● Test whether an SSL connection can be made on port 22.

Click Next to proceed.

Review EULA for Creating the VM Template Panel

On the Review EULA for Creating the VM Template panel, you are required to accept the EULA. You may review the EULAs by clicking the Browse EULA(s) button.

For Ubuntu operating systems, there is a single EULA:

- http://www.ubuntu.com/legal

For Centos operating systems, there are two EULAs:

- http://mirror.centos.org/centos/6/os/i386/EULA
- http://mirror.centos.org/centos/6/os/i386/GPL

Select the **I/We accept all the terms and conditions in the EULA(s) above** option and click Next to continue.

VMware vCenter/vSphere 5 Information Panel

On the VMware vCenter/vSphere 5 Information panel, which opens if you selected VMware vCenter/vSphere Server on the InstallAnywhere VM Template Information panel, you specify the credentials to login to the target VMware vSphere 5 hypervisor.

The VMware vCenter/vSphere 5 Information panel includes the following properties

Table 9-90 ▪ VMware vCenter/vSphere 5 Information Panel

Property	Description
Host Name	Enter the name of your VMware vSphere 5 hypervisor.
User Name **Password**	Enter the credentials to logon to the specified VMware / vSphere 5 hypervisor.
The above specified machine is a VMware vCenter instance	If the credentials specified above are to a VMware vCenter instance, then select this option and specify these additional credentials for the VMware vSphere host that is attached to the instance: • Hostname • Username • Password

Review Summary Information Panel

After you have specified the information required to create a VM template for either VMware vCenter/vSphere Server or AmazonEC2 using the Create InstallAnywhere VM Template Wizard, the Review Summary Information panel opens listing all of your selections.

After you have reviewed the listed information, do one of the following:

- Click Back to adjust the settings.
- Click Build to build the VM template using the specified settings.

Creating the Baseline VM Panel

When you click Build on the Review Summary Information panel, the Creating the Baseline VM panel opens and progress messages are displayed.

Finish Panel

When the build is complete, and you click Next on the Creating the Baseline VM panel, the Finish panel opens, displaying the results of the build.

To view messages about the build, click the Show Details button.

To exit the wizard, click Finish.

Menu Reference

The menus in InstallAnywhere are located on the menu bar, which is at the top of the InstallAnywhere interface. Each menu contains a list of commands.

Each of the menus in InstallAnywhere is described in this section:

Table 9-91 ▪ 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.
Tools	Includes commands to launch wizards, download VM packs and templates, and check for product updates.
Help	Includes the About InstallAnywhere command plus commands to open the help library, check for updates, and specify license information InstallAnywhere.

File

The following table lists the File menu commands, as well as associated keyboard shortcuts.

Table 9-92 ▪ File Menu Commands

Command	Shortcut	Description
New	CTRL+N	Opens the Create New Project dialog box, which enables you to create a new InstallAnywhere project.
Open	CTRL+O	Opens an existing InstallAnywhere project.
Open Recent		Opens one of the most recently accessed projects.
Save		Saves the current project.

Table 9-92 ▪ File Menu Commands (cont.)

Command	Shortcut	Description
Save As		Enables you to save the current project file with a new name and location. The original project still exists, but it remains in the state it was at its last save. The renamed project contains any changes and becomes the currently open project.
Save a Copy As		Enables you to save the project with a new name and location. Unlike the Save As command, the Save a Copy As command makes a copy while leaving the original project as the currently open project.
Revert to Saved		Returns to the last saved state of the project.
Search	CTRL+F	Lets you search for an InstallAnywhere variable in any location of the open project. To learn more, see Searching for InstallAnywhere Texts.
Exit	CTRL+Q	Closes the current project and closes InstallAnywhere.

Edit

The following table lists the Edit menu commands, as well as associated keyboard shortcuts.

Table 9-93 ▪ Edit Menu Commands

Command	Shortcut	Description
Cut Action	CTRL+X	Removes the action that is currently selected in a view on the Sequence page, and moves it to the Clipboard.
Copy Action	CTRL+C	Copies the action that is currently selected in a view on the Sequence page to the Clipboard.
Paste Action	CTRL+V	Inserts the action from the Clipboard below the currently selected action.
Clear Action		Removes the currently selected action.

Table 9-93 ▪ Edit Menu Commands (cont.)

Command	Shortcut	Description
Cut Rules		Provides the following commands: • **All**—CTRL+ALT+X—Removes all of the rules that are associated with the installation (on the Project page) or with the selected action (on the Sequence page), and moves them to the Clipboard. • **Selected**—Removes the currently selected rules that are associated with the installation (on the Project page) or with the selected action (on the Sequence page), and moves it to the Clipboard.
Copy Rules		Provides the following commands: • **All**—CTRL+ALT+C—Copies all of the rules that are associated with the installation (on the Project page) or with the selected action (on the Sequence page) to the Clipboard. • **Selected**—Copies the currently selected rules that are associated with the installation (on the Project page) or with the selected action (on the Sequence page) to the Clipboard.
Paste Rules		Provides the following commands: • **Append**—CTRL+ALT+V—Appends the rule or rules from the Clipboard to the list of rules. • **Replace**—Replaces the rules in the list with the rules in the Clipboard.
Clear Rules		Removes all of the rules that are associated with the installation (on the Project page) or with the selected action (on the Sequence page).
Preferences		Displays the Preferences dialog box, which enables you to specify preferences for creating projects and working in InstallAnywhere. To learn more, see InstallAnywhere Preferences Dialog Box.

Tools

The following table lists the **Tools** menu commands.

Command	Description
Wizards > Create JRE VM Pack	Creates a new VM pack. For more information, see Creating JRE VM Packs.
Wizards > Create VM Template	Launches the Create InstallAnywhere VM Template Wizard, which enables you to create a VM template for virtual appliances.
Downloads > Download Additional VM Packs	Launches a Web page that you can visit to download Java virtual machines. You can bundle the Java virtual machines into the installations that you create in InstallAnywhere.
Downloads > Download VM Templates (Appliance)	
Check for Updates	Checks for service packs and other updates to InstallAnywhere. If updates are available, you can select which ones you would like to download and install. To learn more, see Obtaining Updates for InstallAnywhere.

Help

The following table lists the Help menu commands.

Table 9-94 ▪ Help Menu Commands

Command	Description
InstallAnywhere Help	Displays the InstallAnywhere Help Library in the default Internet browser.
Specify license information	Opens the licensing wizard, which lets you set up licensing for InstallAnywhere.
About InstallAnywhere	Displays the About InstallAnywhere dialog box, where you can find information such as the version of InstallAnywhere. The dialog box also has a Preferences button that you can click to open the Preferences dialog box.

Dialog Box Reference

Each dialog box that is available in InstallAnywhere is described in this section.

- About InstallAnywhere Dialog Box
- Add Files to Project Dialog Box

- Add Installer/Edit Installer Dialog Box
- Add New/Manage Connection Dialog Box
- Advertise Variables Dialog Box
- Build Time Variables Dialog Box
- Configure Input Items Dialog Box
- Change Disk Space and Name Dialog Box
- Choose an Action Dialog Box
- Choose Icon Dialog Box
- Choose Label Settings Dialog Box
- Choose Variable Dialog Box
- Configure Search and Replace Strings Dialog Box
- Create New Project Dialog Box
- Create JRE VM Pack Dialog Box
- Custom Rules Dependencies Dialog Box
- Edit Advertised Variables Dialog Box
- Edit Build Time Variables Dialog Box
- Edit Installer Properties Table Dialog Box
- Filter File Dialog Box
- Install Step Label Settings Dialog Box
- InstallAnywhere Merge Module Import Assistant Dialog Box
- InstallAnywhere Preferences Dialog Box
- Installer Steps Dialog Box
- LaunchAnywhere Properties Dialog Box
- Manage Instances Dialog Box
- Manage Upgrade Configurations Dialog Box
- Create/Open Project Dialog Box
- Create New Project Dialog Box
- Open Project File Dialog Box
- Prepare the Helper Tool Dialog Box
- Read/Modify XML File Dialog Box
- RPM Specification Settings Dialog Box
- Save New Project As Dialog Box
- Search Results Dialog Box

- SWVPD Registry Settings Dialog Box
- Uninstaller Properties Dialog Box
- Uninstaller Steps Dialog Box
- Uninstall Step Label Settings Dialog Box

About InstallAnywhere Dialog Box

The About InstallAnywhere dialog box shows details such as the InstallAnywhere version number and copyright information. This dialog box also includes a Preferences button that launches the InstallAnywhere Preferences dialog box.



Task ***To access the About InstallAnywhere dialog box:***

On the **Help** menu, click **About InstallAnywhere**.

Add Files to Project Dialog Box

The Add Files to Project dialog box lets you add files or directories to your InstallAnywhere project.



Task ***To launch the Add Files to Project dialog box, do one of the following:***

On the **Sequence** page, in the **Install** view, click the **Add Files** button.

The following settings are available on the Add Files To Project dialog box:

Table 9-95 ▪ Add Files to Project Dialog Box Settings

Setting	Description
Folder	The directory that is selected in this list controls which subdirectories are listed below the Path setting. You can click the arrow in this list and select a different directory from the list.
Path	Lists the full path of the directory that is selected in the Folder list. To select a different directory, click the arrow in this list. As an alternative, you can enter a directory path in the Path setting and then click Go to open that directory.
Arrow buttons	Use the arrow buttons to jump to a different directory in the Folder list. • **Up One Folder (up arrow)**—Go up one directory level. • **Go Back (left arrow)**—Go back to the previous directory. • **Go Forward (right arrow)**—Go forward one directory. • **Open Selected Folder (down arrow)**—Open the selected directory.

Table 9-95 ▪ Add Files to Project Dialog Box Settings (cont.)

Setting	Description
Home button	Click the Home button to open the user-specific Home directory. On Windows-based systems, the Home directory is: `C:\Documents and Settings\User_Name`
Favorites button	Click the Favorites button 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 setting and the click the Go button 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 that are listed below the Path setting, click the Add All button.
Remove	To remove the selected directories from the Files to Add list, click the Remove button.
Files to Add	This box lists files and directories that you have selected.  ***Note ▪*** *This type of file linking adds a static list of files to your project: any files that you add later to this source directory are not automatically 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	To cancel the operation without adding any files to the project, click the Cancel button.
Done	To add the selected files and directories to the project, click the Done button.

Add Installer/Edit Installer Dialog Box

Use the Add Installer dialog box or the Edit Installer dialog box to add an installer to an appliance configuration and edit its settings.

- To open the Add Installer dialog box: On the Build page, in the Build Appliances view, click the VM Configuration tab, and then click the Installers subtab. Next, click the Add Installer button.
- To open the Edit Installer dialog box: On the Build page, in the Build Appliances view, click the VM Configuration tab, and then click the Installers subtab. Next, click installer that you want to edit, and then click the Edit Installer button.

Adding installers is the crux of the virtual appliance build process. You can add a prebuilt InstallAnywhere installer, an InstallAnywhere installer out of the current project, or other installers. You can also specify a response file, if you have one.



***Note** ▪ All installers that are specified here must be capable of unattended silent installations; otherwise, the appliance build might stop at the point these installers prompt for user interaction.*

You can add an installer to either the Build Time subtab or the First Boot Time subtab of the Installers subtab:

- **Build-time installers**—Build-time installers are typically used to push payload for the software appliance.
- **First-boot installers**—If there is a need for some configuration to be performed prior to installation, you can use the first boot time installer option. These installers are executed the first time that the virtual appliance is booted.

The following settings are available on the Add/Edit Installer dialog box:

Table 9-96 ▪ Add/Edit Installer Dialog Box

Setting	Description
A pre-built InstallAnywhere installer	Add a prebuilt InstallAnywhere installer (an installer that was built using a different InstallAnywhere project) to this appliance configuration.
An installer out of the current InstallAnywhere project	Add an InstallAnywhere installer that was built using the same InstallAnywhere project to this appliance configuration.
Other installers	Select one of your own custom RPM/Debian packages installers that are capable of unattended installations. For more information and detailed instructions, see Using RPM/Debian Packages for Proprietary Operating Systems on Virtual Appliances.
Location of the installer	Click the Choose button and select the location of the prebuilt InstallAnywhere installer that you are adding.  ***Note** ▪ This setting is displayed if the **A pre-built InstallAnywhere installer** option is selected.*
Choose installer Build Configuration	Select the build configuration from the existing InstallAnywhere project that generates the InstallAnywhere installer that you are adding. If this option is chosen, first the installer is built, and then the appliance is built.  ***Note** ▪ This setting is displayed if the **An installer out of the current InstallAnywhere project** option is selected.*

Table 9-96 ▪ Add/Edit Installer Dialog Box (cont.)

Setting	Description
Choose Build Target	Select the build target from the selected build configuration that generates the InstallAnywhere installer that you are adding. If this option is chosen, first the installer is built, and then the appliance is built.  ***Note ▪*** *This setting is displayed if the* ***An installer out of the current InstallAnywhere project*** *option is selected.*  ***Note ▪*** *If you do not have any Linux-based build targets enabled in your project, this setting is disabled.*
Choose Executable	Click the Choose button and select a custom RPM/Debian package that is capable of an unattended installation.  ***Note ▪*** *This setting is displayed if the* ***Other installers*** *option is selected.*

Table 9-96 ▪ Add/Edit Installer Dialog Box (cont.)

Setting	Description

Installation Command	Enter the command line that is required to install the executable file that is selected in the Choose executable setting.
	
	If you are using a non-InstallAnywhere installer, use the following format for your command line:
	
	`@@executable@@ /i /s /r @@responsefile@@`
	
	Note the following about this format:
	
	• `@@executable@@` denotes the installation executable file.
	• `@@responsefile@@` denotes the response file.
	
	The following are examples of installation commands for Windows and Linux platforms:
	
	• **Windows**—On a Windows-based machine, if the name of executable file is `setup.exe` and the response file name is `resp.rsp`, the aforementioned installation command is executed as follows:
	
	`$> setup.exe /i /s /r resp.rsp`
	
	• **Linux**—For a Linux box example, the name of the executable file is `abc.rpm`, and the installation command is specified as follows:
	
	`rpm -ivh @@executable@@`
	
	In this example, the installation command is executed as follows:
	
	`$> rpm -ivh abc.rpm`
	
	Note ▪ *This setting is displayed if the* ***Other installers*** *option is selected.*

Location of Response File	If you want to use a response file to provide the variables that the end user is required to provide during installation, click the Choose button and then select a response file. When the appliance is being built, the values that are specified in the response file are used to drive the installation.

Note ▪ *Once the installer is added, InstallAnywhere adds a default entry that corresponds with the installer on the Product Properties subtab, which is available on the Sequence page when you click Build Appliances and then click the VM Configuration tab. The Product Properties subtab keeps track of all installers (which are called products in OVF terminology) that are installed onto your appliance.*

Add New/Manage Connection Dialog Box

Use the Add New Connection dialog box, or the Manage Connection dialog box, to specify the credentials and configuration details for connecting to a VMware vSphere 5 or Amazon EC2 hypervisor.



Task ***To open the Add New Connection dialog box or the Manage Connection dialog box:***

1. On the **Sequence** page, click **Build Appliances**.
2. On the **Appliance Configuration** tab, do one of the following:
   - Click the **Add New Credential** button.
   - In the **Connection Settings** box, select the setting that you want to edit, and then click the **Manage Credential** button.

The settings that are displayed on the Add New Credential dialog box or the Manage Credential dialog box vary according to which option is selected in the Target Hypervisor setting: Amazon EC2 or VMware vSphere 5.

Amazon EC2

When you are configuring a virtual appliance for Amazon EC2, the following settings are available on the Add New Connection dialog box and the Manage Connection dialog box.

Table 9-97 ▪ Amazon EC2 Settings

Setting	Description
Identity Name	Enter a nickname for these credentials. This nickname will be used to store your credentials securely in the InstallAnywhere credential store.
Hostname, Username, Password	Enter the connection information to the Amazon EC2 virtual machine that is similar to your target platform. This machine will be used to create your AMI image to deploy to the Amazon cloud.
Account Number **Secret Key** **Access Key** **Private Key** **x509 Certificate**	If you have already enabled the automatic deployment of Amazon appliances (by selecting the Deploy this Appliance check box on the Appliance Configuration tab), provide these additional required Amazon EC2 credentials.  ***Note ▪*** *If the Deploy this Appliance check box is cleared, these settings are disabled.*

VMware vSphere 5

When you are configuring a virtual appliance for VMware vSphere 5, the following settings are available on the Add New Connection dialog box and the Manage Connection dialog box.

Table 9-98 ▪ VMware vSphere 5 Settings

Setting	Description
Identity Name	Enter a nickname for these credentials. This nickname will be used to store your credentials securely in the InstallAnywhere credential store.

Table 9-98 ▪ VMware vSphere 5 Settings (cont.)

Setting	Description
Hostname, Username, Password	Enter the connection information to the VMware vSphere server.
Datacenter	Enter the name of the datacenter on the VMware vSphere server.  ***Note ▪*** *A datacenter is a container for all the inventory objects that are required to complete a fully functional environment for operating virtual machines. You can create multiple datacenters to organize sets of environments.*
Datastore	Enter the name of the datastore on the VMware vSphere server.  ***Note ▪*** *A datastore is a logical container that holds virtual machine files and other files that are necessary for virtual machine operations.*
Resource Pool	Enter the name of the resource pool on the VMware vSphere server.  ***Note ▪*** *Resource pools are used to hierarchically partition available CPU and memory resources of a standalone host or a cluster. Use resource pools to aggregate resources and set allocation policies for multiple virtual machines, without the need to set resources on each virtual machine.*
The above machine is a VMware vCenter instance	If the credentials specified above are to a VMware vCenter instance, select this option and specify these additional details for the VMware vSphere host that is attached to the instance: • vSphere Hostname • vSphere Username • vSphere Datacenter • vSphere Datastore • vSphere Resource Pool

Advertise Variables Dialog Box

When you are creating a merge module, 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.

Advertised variables are used to inform the parent installer of settings that are 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.



Task ***To open the Advertise Variables dialog box:***

1. On the **Project** page, click **Variables**.
2. Click the **Advertise Variables** button.

The following settings are available on the Advertise Variables dialog box.

Table 9-99 ▪ Advertise Variables Dialog Box

Setting	Description
Pre-Install, Install, Post-Install tabs	Select the appropriate tab to indicate the sequence for which you want to advertise variables: Pre-Install, Install, or Post-Install.  ***Note ▪*** *Advertised variables are defined by sequence:* ● *Variables that are set as advertised as Pre-Install affect only the dynamic merge module in pre-install.* ● *The only advertised variables that the Install Merge Module action checks are those that are set in the Install sequence.*
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 the Add button next to the input variables list, and then enter the following information: ● **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 setting blank. ● **Comment**—Enter text to describe the purpose of this variable. To remove a selected variable from the list, click the Remove button.
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: ● **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. To remove a selected variable from the list, click the Remove button.



Note ▪ *For each build configuration, you can specify whether you want to create a merge module installer as part of the build output. For more information, see Creating Merge Modules.*

Build Time Variables Dialog Box

The Build Time Variables dialog box opens when you click the Build Time Variables button in the Variables view of the Project page. This dialog box lists the build-time variables that have been defined for this project. Build-time variables are variables that have their values set at build time.

On the Build Time Variables dialog box, the variable name and value of each defined build time variable are listed.

When you are defining a build-time variable, instead of using the `dollar` symbol ($) as the variable delimiter (which is used for standard variables, such as `$var$`), you instead use the at sign (@) as the delimiter for build time variables, such as `@var@` or `@Apple@`.

To add a build-time variable, click Edit Variables; this opens the Edit Build Time Variables dialog box.



***Note** ▪ In addition to defining build-time variables using the Advanced Designer interface, you can also configure build-time variables using a property file or as environmental variables. For more information, see Resolving Variables at Build Time.*

Configure Input Items Dialog Box

The Configure Input Items dialog box opens when you click Configure on the customizer of the Get User Input - Simple panel action. This dialog box provides settings that you can use to configure the contents of the Get User Input panel.

You can use the Configure Input Items dialog box to do the following:

- 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.
- Specify details such as the typeface, size, text color, and background color for all panel controls.



***Note** ▪ The specific settings that are available on the Configure Input Items dialog box depend on the Input Method that is selected on the Get User Input panel customizer.*

Labels may be literal values or values that are represented by InstallAnywhere variables (which 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).

Task ***To open the Change Disk Space and Name dialog box:***

1. On the **Build** page, click **Build Installers**.
2. On the **Build Configurations** tab, click the **Distribution** subtab.
3. Click the **Change Disk Space and Name** button.

The Change Disk Space and Name dialog box includes the following settings:

Table 9-100 ▪ Change Disk Space and Name Dialog Box

Setting	Description
Media Number	This column shows a read-only number that uniquely identifies the media volume.
Media Name	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. ***Note ▪*** *For compatibility with UNIX-based systems, enter an 8-character name that does not include any spaces.*
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 that is specified on the combined media volumes, new volumes will automatically be created.*
Add	To add a new media volume to the list, click the Add button.
Remove	To remove the selected media volume from the list, click the Remove button.

Choose an Action Dialog Box

The Choose an Action dialog box presents actions that are available for the sequence within which you are working. For example, if you click the Add Action button from within the Install view on the Sequence page, the Choose an Action dialog box does not include tabs for panels.

The following table describes each of the tabs on the Choose an Action dialog box.

Table 9-101 ▪ Choose an Action Dialog Box Controls

Tab	Description	Applicable Views on the Sequence Page
General	Lists general actions.	• Pre-Install • Install • Post-Install • Pre-Uninstall • Uninstall, • Post-Uninstall
Install	Lists actions that are restricted to the Install sequence.	• Install
Panels	Lists available panel actions. Panels actions accept user input and provide feedback during these sequences.	• Pre-Install • Post-Install • Pre-Uninstall • Post-Uninstall
Consoles	Console actions serve the same purpose as panel actions for console (non-GUI, non-silent) installers.	• Pre-Install • Post-Install • Pre-Uninstall • Post-Uninstall
System i (i5/OS)	Lists actions that are specific to the System i (i5/OS) platform.	• Pre-Install • Install • Post-Install • Pre-Uninstall • Post-Uninstall
Plug-Ins	Shows plug-in actions, if any are installed.  ***Note ▪*** *The Plug-ins tab appears when a custom code plug-in has been added.*	• Pre-Install • Install • Post-Install • Pre-Uninstall • Post-Uninstall



Note ▪ *For information on the individual actions that are 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
- 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.

The Windows tab of the Choose Icon dialog box includes the following controls:

Table 9-102 ▪ Windows Tab of Choose Icon Dialog Box

Control	Description
Choose .ICO File	Click Choose ICO File and select a Windows icon file (.ico). After a file is selected, the Path and Name fields will be populated.  ***Note ▪*** *The Choose Icon dialog box does not display a preview of the selected .ico file.*
32x32 Icon	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.
16x16 Icon	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.

OS X Tab

The OS X tab of the Choose Icon dialog box includes the following controls:

Table 9-103 ▪ OS X Tab of Choose Icon Dialog Box

Control	Description
OS X Icon File (.icns)	Click Choose ICNS File and select a OS or OS X icon resource file (.icns). After a file is selected, the Path and Name fields will be populated.

Choose Label Settings Dialog Box

The Choose Label Settings dialog box is where you change the icons for the installer steps that the installer user interface shows for the installer progress. It is also where you change the font color of the installer step text.



Task ***To access the Choose Label Settings dialog box:***

On the **Installer Steps** dialog box, click the **Icons and Fonts** button.

To learn how to access and use the Installer Steps dialog box, see Installer Steps Dialog Box.

The Choose Label Settings dialog box is organized into the following areas:

- **Active Step Label Settings**—This area is where you configure the icon and the font color of the installer step that is currently being performed at run time.
- **Completed Step Label Settings**—This area is where you configure the icon and the font color of the installer steps that have been completed.
- **Upcoming Step Label Settings**—This area is where you configure the icon and the font color of the installer steps that have not yet been performed.

The following settings are available in each of the aforementioned areas.

Table 9-104 ▪ Choose Label Settings Dialog Box

Setting	Description
Path	This read-only setting shows the path to the folder that contains the selected image that is being used for the designated type of label. To change the path, click the Choose Icon button.
Name	This read-only setting shows the name of the selected image that is being used for the designated type of label. To change the name, click the Choose Icon button.
Default Icon	To revert to the default icon for the designated type of label, click the Default Icon button.
Choose Icon	To change the icon that is being used for the designated type of label, click the Choose Icon button.
Font Settings	To change the font color of the text that is used for the designated type of label, click the Font Settings button. You can also click the **Import Font** button to import a true type font to use. When the font is imported, it will be available from the list of fonts you can select. The imported font will also be shipped with your installer.

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.

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.

Create New Project Dialog Box

Available from the Advanced Designer interface when you click New on the File menu, the Create New Project dialog box enables you to create a new project based on a standard or custom template.

Table 9-105 ▪ Create New Project Dialog Box Settings

Controls	Description
Template list	Lists the standard InstallAnywhere project templates.
Other Template	Opens the Choose Template dialog box, which enables you to 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 JRE VM Pack Dialog Box

You can use the Create JRE VM Pack dialog box, which opens when you click Create JRE VM Pack on the File menu of the Advanced Designer, to create a JRE VM pack for any platform.



***Note** ▪ You can use the JRE VM Pack Utility as a standalone tool on a machine where InstallAnywhere is not installed. To learn more, see Running the Create JRE VM Pack Wizard as a Standalone Tool.*

The following settings are available on the Create JRE VM Pack dialog box.

Table 9-106 ▪ Create JRE 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* `jre`*. For more information, see Directory Name Requirement for Preparing a JRE VM Pack.*
Choose destination folder	Click Choose and select the directory where you want to store the new VM pack.
Platform	Choose the platform of this new VM pack from the list. Available options are: • Windows • Solaris • HP-UX • AIX • Linux
Choose a name for VM pack	Enter a name to uniquely identify the new VM pack.
Create JRE VM Pack	Click to create the VM pack using the specified information.

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 you are 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 opens when you click Edit Variables on the customizer of an Install Merge Module action in the Install view of the Sequence page.

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.

The Edit Advertised Variables dialog box includes the following options:

Table 9-107 ▪ Edit Advertised Variables Dialog Box

Option	Description
Input Variables List	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: • **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 `$VARIABLE_NAME$`. 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 merge module, click Add next to the parent variables list to add an entry to the list, and then enter the following information: • **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 `$VARIABLE_NAME$`. Click Remove to remove a selected variable from the list.
Send stdout to variable **Send stderr to Variable**	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.

Edit Build Time Variables Dialog Box

On the Edit Build Time Variables dialog box, which is opened by clicking the Edit Variables button on the Build Time Variables dialog box, you can add new build time variables to the project and edit existing build time variables. Build time variables are variables that have their values set at build time.

To add a build time variable, perform the following steps:



Task ***To add a build time variable:***

1. Click **Add**. A new, empty variable is added to the **Build Time Variable** list.
2. In the **Build Time Variable** column, enter a name for the new variable using the `@` symbol as the variable delimiter.

   When defining a build time variable, instead of using the `$` symbol as the variable delimiter (which is used for standard variables, such as `$var$`), you instead use the `@` symbol as the delimiter for build time variables, such as `@var@` or `@Apple@`.
3. In the **Value** column, enter a value for the variable.
4. Click **OK**. The **Edit Build Time Variables** dialog box closes. The new build time variable is now listed on the **Build Time Variables** dialog box.
5. Click **OK**.



Note ▪ *In addition to defining build time variables using the Advanced Designer interface, you can also configure build time variables using a property file or as environmental variables. For more information, see* *Resolving Variables at Build Time*.

Edit Installer Properties Table Dialog Box

When you add an installer to a VM configuration for a virtual appliance (that is, when you add an installer to the Installers subtab of the VM Configuration tab in the Build Appliances view of the Build page), InstallAnywhere determines the installer's response variables and displays the Edit Installer Properties Table dialog box. This dialog box is where you enter default values for the response variables. The values (or response options) need to be supplied to the installer in order for it to silently install on the virtual appliance. When the virtual appliances are being built, the default values that you have specified are used to configure the installation.

If you are creating a multi-tier virtual appliance for VMware vSphere, ensure that you define the properties that each tier in your virtual appliance needs in order to connect to each other. For example, you may need to configure the IP address, port number, user name, and password of your database tier so that your application tier can connect to it.



Task ***To open the Edit Installer Properties Table dialog box:***

1. On the **Build** page, click **Build Appliances**.
2. Click the **VM Configuration** tab.

3. In the list of products, select the **View/Edit** button for the product whose properties you want to edit.

On the Edit Installer Properties Table dialog box, you can perform the following tasks:

- **Edit a property**—In the Value setting for each product property, specify a value. When InstallAnywhere runs this installer on the virtual machine, it passes these customized values to the installer.
- **Add a property**—If you want to add a property that was not automatically discovered by InstallAnywhere, click Add.

  ***Note** ▪ This is a required step when using third-party installers because InstallAnywhere is unable to automatically discover the properties in a third-party installer.*
- **Remove a property**—To remove a property from the list, select it and click Remove.



***Note** ▪ Alternatively, to override the discovery algorithm (which could take time for bigger installers), you can specify the response file directly on the Add Installer/Edit Installer Dialog Box; InstallAnywhere will obtain the response variables from the file that you specify.*

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.

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 9-108 ▪ 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: • Begins With • Ends With • Contains • Regular Expression

Table 9-108 ▪ Filter Options (cont.)

Option	Description
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.

Install Step Label Settings Dialog Box

The Install Step Label Settings dialog box is where you specify which label you want your installers to show on the Install Progress panel.



Task ***To access the Install Step Label Settings dialog box:***

1. In the Advanced Designer, on the **Installer UI** page, click **Look & Feel Settings**. The **Look & Feel Setting** view opens.
2. Ensure that one of the allowable UI modes is GUI:
   a. Expand the **General UI Setting** area.
   b. In the **Allowable UI Modes** setting, ensure that the **GUI** check box is selected.
3. Ensure that the installer is configured to show a list of installer steps for the installer progress:
   a. Expand the **UI Panel Settings** area.
   b. In the **Custom UI Designer** setting, click the **Custom UI Designer** button. A themes dialog box appears.
   c. From the **Installer Area** drop-down list, select **Installer Steps**.
   d. Expand the **Installer Steps** area.
   e. Ensure that the **Show Installer Steps** setting is set to **Yes**.
   f. In the **Installer Steps Type** setting, ensure that **List of installer steps** is selected.
4. Expand the **Install Progress Panel** area.
5. In the **Install Progress Label** setting, click the **Install Step Label Settings** button.

The Install Step Label Settings dialog box has two options:

- **Highlight the Same Label as the Previous Panel**—To keep the label the same as shown on the previous panel during the installation, select this option.
- **Choose a Label from the List of Install Step Labels**—To display the selected label during the installation, select this option, and then select the appropriate label in the box.

InstallAnywhere Merge Module Import Assistant Dialog Box

Use the InstallAnywhere Merge Module Import Assistant dialog box to specify which aspects of the selected merge module you want to import and integrate into the currently open project.



Task ***To open the InstallAnywhere Merge Module Import Assistant dialog box:***

1. On the **Organization** page, click **Modules**.
2. Click the **Import Merge Module** button. The **Choose Merge Module** dialog box opens.
3. Select the merge module (.iam.zip) that you want to import, and then click the **Open** button.

The following settings are available on the InstallAnywhere Merge Module Import Assistant dialog box.

Table 9-109 ▪ InstallAnywhere Merge Module Import Assistant Dialog Box

Control	Description
From what sequences contained within the Merge Module should actions be imported?	Select one or more of the following options to specify the sequences in the merge module from which actions should be imported: ● Pre-Install Sequence (Panels, Consoles, and Actions) ● Install Sequence ● Post-Install Sequence (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: ● Import each Feature as a top-level Feature ● Import each Feature as a child of a new Feature
Import Tags	To import the build tags that are defined in the imported merge module into this project, select this check box.

InstallAnywhere Preferences Dialog Box

The InstallAnywhere Preferences dialog box enables you to specify options for creating projects and working in InstallAnywhere.



Task ***To open the InstallAnywhere Preferences dialog box:***

On the **Edit** menu, click **Preferences**.

The InstallAnywhere Preferences dialog box is also available from the About InstallAnywhere dialog box.

The InstallAnywhere Preferences dialog box is organized into the following tabs:

- General Settings Tab
- Resources Tab
- Source Paths Tab
- Updates Tab

General Settings Tab

Use the General Settings tab on the InstallAnywhere Preferences dialog box to select default settings for InstallAnywhere.

Table 9-110 ▪ Settings on the General Settings Tab of the InstallAnywhere Preferences Dialog Box

Setting	Description
Project Loading	Specify whether InstallAnywhere should verify the location of files that are referenced in the Install view of the Sequence page when you open projects. Available options are: • **Always (recommended)**—When you open a project, InstallAnywhere verifies the location of files that are referenced in the Install view of the Sequence page. • **Never**—When you open a project, InstallAnywhere does not verify the location of files that are referenced in the Install view of the Sequence page.
Command Line Builds	Specify how you want a command-line build to proceed if referenced files are missing. Available options are: • **Fail (recommended)**—If referenced files are missing during a command-line build, the build fails. • **Continue without including the missing files**—If referenced files are missing during a command-line build, the build continues without the missing files.
Multiple Build Configurations	Specify the appropriate behavior that results if an error occurs when InstallAnywhere is building multiple build configurations. Available options are: • **Stop Building**—Stop the entire project build. • **Continue Building**—Continue building the remaining build configurations.
Manifest Files	Specify whether InstallAnywhere can successfully build installers for a project that includes a manifest file that references missing files. Available options are: • **Fail (recommended)**—If the project includes a manifest file that references missing files, the build fails. • **Continue without including the missing files**—If the project includes a manifest file that references missing files, the build proceeds without the missing files 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.

Table 9-110 ▪ Settings on the General Settings Tab of the InstallAnywhere Preferences Dialog Box (cont.)

Setting	Description
Default Selection for the Install View	Specify the default option that should be selected in the **Assign to** list when you open the Install view on the Sequence page. Available options are: • Components • Product Features
Save the project before build	Determines whether a project is saved before the build is performed: • **Option is selected**—The project is saved prior to building the installer; all settings are saved. • **Option is not selected**—If you make changes to project settings and then build the installer, the setting changes are in the built installer but they are not saved to the project file.

Resources Tab

Use the Resources tab on the InstallAnywhere Preferences dialog box to provide semicolon-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 predefined 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 settings:

Table 9-111 ▪ Settings on the Resources Tab of the InstallAnywhere Preferences Dialog Box

Setting	Description
Plugin Resource Paths	InstallAnywhere scans the paths listed here and populates the Plug-Ins tab with the plug-ins that it finds in those paths as well as those stored at the standard plug-in location: <*InstallAnywhere*>\plugins\  ***Note ▪*** *The plug-in list in the Choose an Action dialog box is updated every time that the dialog box opens.*

Table 9-111 ▪ Settings on the Resources Tab of the InstallAnywhere Preferences Dialog Box (cont.)

Setting	Description
VM Pack Resource Paths	InstallAnywhere scans the paths listed here and populates the VM to Bundle with Installer list with the VM packs that it finds in those paths as well as those stored at the standard VM pack location: `<InstallAnywhere>\resource\installer_vms\` ***Note ▪*** *The VM packs listed in the VM list on the Build Targets tab (VM to Bundle with Installer) are refreshed each time that 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.*
JVM Specs Resource Paths	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. 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 setting. In this field, enter a semicolon-delimited list of paths 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 that it finds in these paths as well as those stored at the default JVM spec file location (`<IA_HOME>\resource\jvms`). ***Note ▪*** *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—for example:* 

Source Paths Tab

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 that is listed on the Source Paths tab of the InstallAnywhere Preferences dialog box, 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 9-112 ▪ Settings on the Source Paths Tab of the InstallAnywhere Preferences Dialog Box

Setting	Description
Enable Source Paths	To use source path variables instead of absolute hard-coded paths, select this check box.
Source Path List	This box on the Source Paths tab lists all of the defined source paths. The box contains the following columns: • **Access Path Name**—Variable name to use in the source path. • **Folder**—Location of the source path.
Add	To add an entry to the source paths box, click this button. InstallAnywhere adds a new row to the box, enabling you to specify the source path name and location. To refer to a source path elsewhere in your InstallAnywhere project, use the following notation: `$<access_path_name>$` For example, if the Access Path Name column is **RESOURCE**, refer to it elsewhere in the project as **$RESOURCE$**.
Remove	Click to delete the selected row.
Default Source Paths	This area lists all of the default source paths for the currently open project. Select the check boxes of the source paths that you want to be able to use for the currently open project.



Note ▪ *For more information about source paths, see Working with Source Paths.*

Updates Tab

Use the Updates tab on the InstallAnywhere Preferences dialog box to enable FlexNet Connect to automatically check for updates that are available for InstallAnywhere. For more information on configuring InstallAnywhere to automatically check for updates and how to install updates, see Obtaining Updates for InstallAnywhere.

You can also use the Updates tab to manage your participation in Revenera'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** 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.

Installer Steps Dialog Box

The Installer Steps dialog box is where you configure the text, the text color, and the icons for the installer steps that the installer UI shows for the installer progress. You can use this dialog box to add, edit, remove, and reorder the text and their corresponding icons, and to change the color of the text.



Task ***To access the Installer Steps dialog box:***

1. In the Advanced Designer, on the **Installer UI** page, click **Look & Feel Settings**. The **Look & Feel Setting** view opens.
2. Ensure that one of the allowable UI modes is GUI:
    a. Expand the **General UI Setting** area.
    b. In the **Allowable UI Modes** setting, ensure that the **GUI** check box is selected.
3. Ensure that the installer is configured to show a list of installer steps for the installer progress:
    a. Expand the **UI Panel Settings** area.
    b. In the **Custom UI Designer** setting, click the **Custom UI Designer** button. A themes dialog box appears.
    c. From the **Installer Area** drop-down list, select **Installer Steps**.
    d. Expand the **Installer Steps** setting.
    e. Ensure that the **Show Installer Steps** setting is set to **Yes**.
    f. In the **Installer Steps Type** setting, ensure that the **List of installer steps** is selected.
4. Expand the **Panel Image** setting.
5. In the **Configure Install Labels** setting, click the **Installer Step Labels setting** button.

The following settings are available on the Installer Steps dialog box.

Table 9-113 ▪ Settings on the Installer Steps Dialog Box

Setting	Description
Configure Install Labels	This box shows the list of items that the installer UI shows for the installer progress. The items are listed in the order that they appear in the list of installer steps, from top to bottom.
Add Label	To add a progress step to the list of installer steps, click this button. This button is enabled when the **Auto populate labels when saving** check box is cleared.
Edit Label	To edit the text for the selected installer step, click this button. This button is enabled when the **Auto populate labels when saving** check box is cleared.

Table 9-113 ▪ Settings on the Installer Steps Dialog Box (cont.)

Setting	Description
Remove Label	To remove the selected installer step, click this button. This button is enabled when the **Auto populate labels when saving** check box is cleared.
Auto Populate	To add a label for every panel action that is added to the Pre-Install and Post-Install sequences, click this button.
Move Label Up arrow	To move up the label that is selected in the List of Labels for Installer Steps box, click the up arrow button. This button is enabled when the **Auto populate labels when saving** check box is cleared.
Move Label Down arrow	To move down the label that is selected in the List of Labels for Installer Steps box, click the down arrow button. This button is enabled when the **Auto populate labels when saving** check box is cleared.
Icons and Fonts	To specify the small graphics that are displayed next to the text labels that show the progress of the installation, or to change the color of the font that is used for the text labels, click this button.
Top Offset	To add space above the installer steps, specify the number of pixels in this field to include as an offset of space. This will offset the installer steps from the top of the panel by the amount of pixels you specify.  ***Tip ▪*** *An example use of this setting might be instances where you want include a personalized company logo directly above the installer steps. The offset would then push the installer steps down to accommodate space needed to include addition of a logo.*
Auto populate labels when saving	To create a list of labels that matches the current set of panel actions each time that you save changes to your project, click this button.
Allow label text to wrap	To avoid the possibility that any long installer step labels are truncated in the installer UI, select this check box. If you select this check box, any long label text continues on successive lines.
Allow Vertical Scroll	To navigate through a list of labels by moving up and down using the scroll bar in the Panel. If you select this check box, the Vertical Scroll is available when the number of labels exceeds in the Installer Step Frame.

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.

Manage Instances Dialog Box

The Manage Instances dialog box opens when an end user launches the installer for a product that has already been installed using an installation in which instance management is enabled. The end user is prompted to choose to either install a new instance or modify an existing instance.

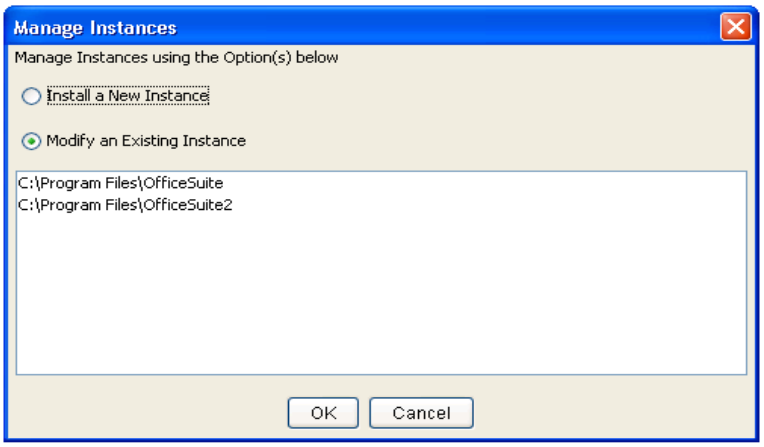


Figure 9-1: Manage Instances Dialog Box

The action that takes place depends upon what the user selects on this dialog box:

- If the end user selects the Install a New Instance option, the installation continues and prompts the end user to select a location for the new instance of the product.
- If the end user selects the Modify an Existing Instance option and selects an instance from the list, maintenance mode is launched, and the end user is prompted to choose whether to add features, remove features, repair the product, or uninstall the product.

Manage Upgrade Configurations Dialog Box

The Manage Upgrade Configurations dialog box is where you define how your upgrade should detect earlier versions of the product that need to be updated.



Task ***To access the Manage Upgrade Configurations dialog box:***

1. In the Advanced Designer, on the **Project** page, click **Upgrades**. The **Upgrades** view opens.
2. Find the **Upgrade Configuration** setting.
3. In the **Upgrade Configuration** setting, click the **Manage Upgrade Configurations** button.

The following settings are available on the Manage Upgrade Configurations dialog box.

Table 9-114 ▪ Manage Upgrade Configurations Dialog Box

Setting	Description
List of Upgrade Configurations	Use this list to define one or more configurations that specify how your upgrade should detect earlier versions of the product that need to be updated. By default, this list shows a Default Upgrade Configuration item. To manage the items in this list: ● To add a new configuration, click the Add button next to the list. ● To view or edit the settings for a particular configuration, select the configuration and then review and modify the settings below the list as needed. ● To remove an existing configuration, select it in the list, and then click the Delete button.
Criterion for Detecting Which Products to Upgrade	Select the code that you want the current upgrade configuration to target. Available options are: ● **Products that Share My Upgrade Code**—The installer searches target systems for the presence of a product that has the same upgrade code. ● **Products that Share My Product Code**—The installer searches target systems for the presence of a product that has the same product code. If the product code option is selected, the following check box is available: ● **Product Code for Earlier Version Does Not Match that of the Current Project**—The installer searches target systems for the presence of a product that has a different product code that you specify. If you select this option, ensure that you use the Product Code of the Earlier Version box to indicate the product code for which you want the installer to search. The upgrade code is listed in the Upgrades view on the Project page of a project. The product code is listed in the General Settings view on the Project page of a project. To learn more about selecting an option for this setting, see Detecting Installed Earlier Versions that Need to Be Updated.
Range of Product Versions to Upgrade	Enter a range of version numbers that you want the current upgrade configuration to target. The range of version numbers for the upgrade configuration must be lower than the version number of the upgrade. That is, the range of version numbers must be less than the version number that you specify in the General Settings view on the Project page of the current project. To learn more, see Detecting Installed Earlier Versions that Need to Be Updated.

Create/Open Project Dialog Box

The Create/Open Project dialog box is provided to choose to either create a new project or open an existing project. This dialog box is the first dialog you see when you open InstallAnywhere.

The Create/Open Project dialog box includes the following options:

Table 9-115 ▪ Create/Open Project Dialog Box

Option	Description
Create new project	Select this option to open the Create New Project Dialog Box, where you select a template, and enter a project name and location.
Open existing project	Select this option to display the Open Project File Dialog Box where you can choose an existing InstallAnywhere project to open.
Exit	Click to exit InstallAnywhere.

Create New Project Dialog Box

On the **Create New Project** dialog box, which opens when you click **Create new project** on the Create/Open Project Dialog Box, you select a template, and enter a project name and location.

The Create/Open Project dialog box includes the following options:

Table 9-116 ▪ Create/Open Project Dialog Box

Option	Description
Template Buttons	Choose one of the following templates to base your new project on: • Basic Project Template • Typical Project Template • Quick Install Template • Sample Dependency Template • Console Project Template A template is simply a merge module that has been placed within the `iatemplates` folder inside the InstallAnywhere installation folder. 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.
Other Template	Click to open the **Choose Template** dialog box, where you can select a custom template (`iam.zip`).

Table 9-116 ▪ Create/Open Project Dialog Box (cont.)

Option	Description
Project Name	Enter a name for this new project. By default, the project name is `My_Product_n.iap_xml`.
Project Path	Enter or browse to the location where you want to save this project file.
Create project file in 'Project Name' subfolder	Select this option to save the project file in a subfolder of the selected **Project Path** that matches the **Project Name**. For example, if the name of your project file was `ABCProduct.iap_xml`, and the path was `C:\InstallAnywhere 2022 Projects`, the full path to the new project file would be: `C:\InstallAnywhere 2022 Projects\ABCProduct\ABCProduct.iap_xml`
OK	Click to create the project.

Open Project File Dialog Box

Use the Open Project File dialog box to browse to the InstallAnywhere project file (.iap_xml or .iap) or project manifest (.imf_xml) that you want to open in InstallAnywhere.



Task ***To open the Open Project File dialog box, do one of the following:***

- On the Create/Open Project Dialog Box, click **Open existing project**.
- On the **File** menu of the InstallAnywhere user interface, click **Open**.

The following setting are available on the **Open Project File** dialog box.

Table 9-117 ▪ Open Project File Dialog Box Settings

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.
Files of Type	Filters the files that appear in the directory list. Available options are: - InstallAnywhere Project Files (*.iap_xml, *.iap) - Project Manifest (*.imf_xml)

Table 9-117 ▪ Open Project File Dialog Box Settings (cont.)

Control	Description
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. The project opens on the **Project** page of the InstallAnywhere interface.

Prepare the Helper Tool Dialog Box

The Prepare the Helper Tool dialog box is where you specify the location of the OS or OS X SDK that you want to use to build the helper tool for code-signed OS or OS X–based installers that require authentication. It is also where you specify details about the Developer ID Application certificate that you are using to code sign your helper tool and OS or OS X–based installers. When you click the Build button on this dialog box, InstallAnywhere prepares and code signs your helper tool using the information that you specified.

Once you have code signed your helper tool, verify that it was signed properly. To learn how, see Verifying that Your Code-Signing Output Files Are Working as Expected on OS or OS X–Based Target Systems.



Task ***To access the Prepare the Helper Tool dialog box on a OS or OS X–based machine:***

On the **File** menu, click **Prepare Helper Tool**.

The Prepare Helper Tool command on the File menu and the Prepare the Helper Tool dialog box are available only on OS or OS X–based systems, since all code signing for OS X–based target systems must be performed on this platform. For more details, see Requirements for Code-Signing Support for OS or OS X–Based Installers.

The following settings are available on the Prepare the Helper Tool dialog box.

Table 9-118 ▪ Settings on the Prepare the Helper Tool Dialog Box

Setting	Description
Select the OS or OS X SDK for Building the Helper Tool	The list for this option contains all of the OS or OS X SDKs that are installed with Xcode and available to the user who is currently logged on to the machine. The default path is for the latest version of the OS or OS X SDK. Select the SDK that you want to use to prepare the helper tool. If you want to use a different OS or OS X SDK, select the **Choose the Path to OS X SDK** option, and then browse to and select the appropriate path.
Choose the Path to the OS X SDK	This option enables you to browse to and select an OS or OS X SDK that is not listed in the other option's list of paths. For example, if your machine has OS X 10.9, but you want to use the OS or OS X 10.10 SDK to build the helper tool, you can use this option to specify the path to the newer OS or OS X 10.10 SDK.

Table 9-118 ▪ Settings on the Prepare the Helper Tool Dialog Box (cont.)

Setting	Description
Common Name of the Certificate (as displayed in Keychain Access)	Specify the common name of the certificate. This name is displayed in Keychain Access, as shown in the screen shot below. The common name must use syntax such as this: **Developer ID Application: AAA Software LLC**
User ID of the Certificate (as displayed in Keychain Access)	Specify the user ID of the certificate. This ID is displayed in Keychain Access, as shown in the screen shot below.

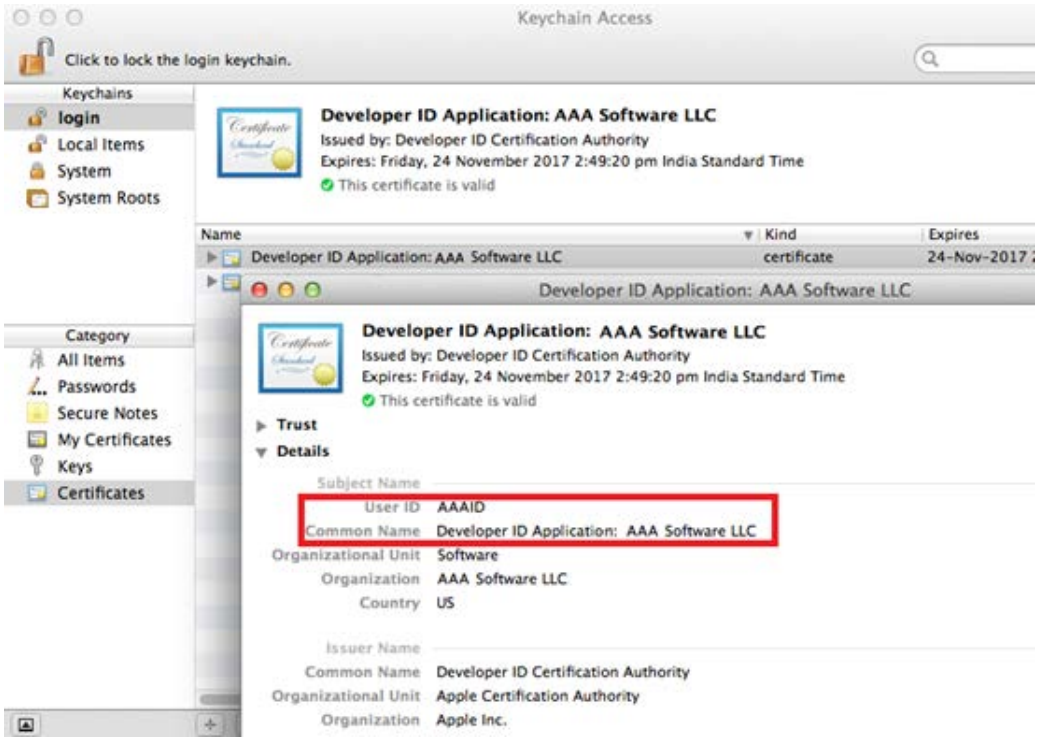


Figure 9-2: User ID and Common Name in Keychain Access

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.

RPM Specification Settings Dialog Box

On supported Linux systems, Red Hat Package Management (RPM) registration creates a virtual package and uses it to make entries to the RPM database. Use the RPM Specification Settings dialog box to configure settings for RPM registration.



Task ***To launch the RPM Specification Settings dialog box:***

1. On the **Project** page, click **General Settings**. The **General Settings** view opens.
2. Expand the **UNIX Tag** area of the view.
3. In the **Enable RPM Registration (Linux Only)** setting, select **Yes**, and then click the ellipsis button (**...**) in this setting.

The following settings are available on the RPM Specification Settings dialog box.

Table 9-119 ▪ RPM Specification Settings Dialog Box

Control	Description
Name	Enter the name string from the RPM file name that you plan to use.
Version	Enter the version number. This version number is included in the package name.
Release	Enter the release number. The release number distinguishes two different packages with the same version number.
Description	Enter a brief description of the functionality or application.
Summary	Enter a one-line description of the package functionality.
Copyright	Enter the copyright notice for the package. On target systems that are running RPM version 4 or later, the Copyright setting is not used.
License	Specify the license (GPL, LGPL, commercial) for the package to be installed. On target systems that are running versions of RPM prior to 4, the License setting is ignored.
URL	Enter the URL that you want the package to reference.
Distribution	Enter the name of the distribution to which this package belongs, if any.
Vendor	Enter 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. The default value is **Applications/System**.
Packager	Enter 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 opens when you select **Save As** from the **File** menu.

Table 9-120 ▪ 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.
Arrow buttons	Use to jump to different directory in the Folder list. • **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.
Home button	Click to open the user-specific Home directory which, in Windows is: `C:\Documents and Settings\`*`User_Name`*
Favorites button	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 • 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.

Table 9-120 ▪ Save New Project As Dialog Box (cont.)

Options	Description
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 InstallAnywhere interface.)

Search Results Dialog Box

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 sequence (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.

Variables Tab

The Variables tab of the Search Results dialog box includes the following controls.

Table 9-121 ▪ Variables Tab of Search Results Dialog Box

Control	Description
Enter Variable Name	Enter the name of the variable to search for. 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.  ***Note ▪*** *It is always recommended that you enter the* ***$*** *symbols before and after the variable name.*
Search for variable in	Select the installation sequences (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.*

Table 9-121 ▪ Variables Tab of Search Results Dialog Box (cont.)

Control	Description
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.
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 ▪*** *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$***.*
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 9-122 ▪ 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 sequences (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.
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 tags found listed at the top, and the found tags grouped by category.

SWVPD Registry Settings Dialog Box

Use the SWVPD Registry Settings dialog box to specify 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.



Task ***To launch the SWVPD Registry Settings dialog box:***

1. On the **Project** page, click **General Settings**. The **General Settings** view opens.
2. Expand the **UNIX Tag** area of the view.
3. In the **Enable SWVPD Registration Integration (AIX Only)** setting, select **Yes**, and then click the ellipsis button (**...**) in this setting.



Note ▪ *If you enable SWVPD registry integration and leave one or more settings on the SWVPD Registry Settings dialog box blank, the installer uses corresponding values from the General Settings view on the Project page.*

The SWVPD Registry Settings dialog box includes the following settings:

Table 9-123 ▪ SWVPD Registry Settings Dialog Box

Setting	Description
Product Name	Enter the name under which you want this product to be logged in SWVPD.
Product Version	Enter the version under which you want this product to be logged in SWVPD.
Product Description	Enter the description to be entered for this product in SWVPD.

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.

Uninstaller Steps Dialog Box

The Uninstaller Steps dialog box is where you configure the text for the uninstaller steps that the uninstaller UI shows for the uninstaller progress. You can use this dialog box to add, edit, remove, and reorder the text.



Task ***To access the Uninstaller Steps dialog box:***

1. In the Advanced Designer, on the **Installer UI** page, click **Look & Feel Settings**. The **Look & Feel Setting** view opens.
2. Ensure that one of the allowable UI modes is GUI:

a. Expand the **General UI Setting** area.

b. In the **Allowable UI Modes** setting, ensure that the **GUI** check box is selected.

3. Ensure that the uninstaller is configured to show a list of uninstaller steps for the uninstaller progress:

a. Expand the **UI Panel Settings** area.

b. In the **Custom UI Designer** setting, click the **Custom UI Designer** button. A themes dialog box appears.

c. From the **Installer Area** drop-down list, select the **Installer Steps**.

d. Expand the **Installer Steps** setting.

e. Ensure that the **Show Installer Steps** setting is set to **Yes.**

f. In the **Installer Steps Type** setting, ensure that the **List of installer steps** is selected.

g. Ensure that the **Enable Uninstall Label Settings** setting is set to **Yes**.

4. Now expand the **Panel Image** setting.

5. In the **Configure Uninstall Labels** setting, click the **Uninstaller Step Labels setting** button.

The following settings are available on the Uninstaller Steps dialog box.

Table 9-124 ▪ Settings on the Uninstaller Steps Dialog Box

Setting	Description
Configure Uninstall Labels	This box shows the list of items that the uninstaller UI shows for the uninstaller progress. The items are listed in the order that they appear in the list of uninstaller steps, from top to bottom.
Add Label	To add a progress step to the list of uninstaller steps, click this button. This button is enabled when the **Auto populate labels when saving** check box is cleared.
Edit Label	To edit the text for the selected uninstaller step, click this button. This button is enabled when the **Auto populate labels when saving** check box is cleared.
Remove Label	To remove the selected uninstaller step, click this button. This button is enabled when the **Auto populate labels when saving** check box is cleared.
Auto Populate	To add a label for every panel action that is added to the Pre-Uninstall and Post-Uninstall sequences, click this button.
Move Label Up arrow	To move up the label that is selected in the List of Labels for the Uninstaller Steps box, click the up arrow button. This button is enabled when the **Auto populate labels when saving** check box is cleared.

Table 9-124 ▪ Settings on the Uninstaller Steps Dialog Box

Setting	Description
Move Label Down arrow	To move down the label that is selected in the List of Labels for the Uninstaller Steps box, click the up arrow button. This button is enabled when the **Auto populate labels when saving** check box is cleared.
Top Offset	To add space above the uninstaller steps, specify the number of pixels in this field to include as an offset of space. This will offset the uninstaller steps from the top of the panel by the amount of pixels you specify.  ***Tip ▪*** *An example use of this setting might be instances where you want include a personalized company logo directly above the uninstaller steps. The offset would then push the uninstaller steps down to accommodate space needed to include addition of a logo.*
Auto populate labels when saving	To create a list of labels that matches the current set of panel actions each time that you save changes to your project, click this button.
Allow label text to wrap	To avoid the possibility that any long uninstaller step labels are truncated in the uninstaller UI, select this check box. If you select this check box, any long label text continues on successive lines.
Allow Vertical Scroll	To navigate through a list of labels by moving up and down using the scroll bar in the Panel. If you select this check box, the Vertical Scroll is available when the number of labels exceeds in the Uninstaller Step Frame.



Note ▪ *Icons and texts font colour for the uninstaller steps are same as those configured for the installer steps.*

Uninstall Step Label Settings Dialog Box

The Uninstall Step Label Settings dialog box is where you specify which label you want your uninstallers to show on the Uninstall Progress panel.



Task ***To access the Uninstall Step Label Settings dialog box:***

1. In the Advanced Designer, on the **Installer UI** page, click **Look & Feel Settings**. The **Look & Feel Setting** view opens.
2. Ensure that one of the allowable UI modes is GUI:
   a. Expand the **General UI Setting** area.
   b. In the **Allowable UI Modes** setting, ensure that the **GUI** check box is selected.

3. Ensure that the uninstaller is configured to show a list of uninstaller steps for the uninstaller progress:
    a. Expand the **UI Panel Settings** area.
    b. In the **Custom UI Designer** setting, click the **Custom UI Designer** button. A themes dialog box appears.
    c. From the **Installer Area** drop-down list, select **Installer Steps**.
    d. Expand the **Installer Steps** area.
    e. Ensure that the **Show Installer Steps** setting is set to **Yes**.
    f. In the **Installer Steps Type** setting, ensure that the **List of installer steps** is selected.
    g. Ensure that the **Enable Uninstall Label Settings** setting is set to **Yes**.
4. Expand the **Install Progress Panel** area.
5. In the **Uninstall Progress Label** setting, click the **Uninstall Step Label Settings** button.

The Uninstall Step Label Settings dialog box has two options:

- **Highlight the Same Label as the Previous Panel**—To keep the label the same as shown on the previous panel during the uninstallation, select this option.
- **Choose a Label from the List of Install Step Labels**—To display the selected label during the uninstallation, select this option, and then select the appropriate label in the box.

Actions

Actions represent operations for the installer to perform. InstallAnywhere supports an extensible action architecture the InstallAnywhere 2026 R1 User Guide 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 9-1 ▪ InstallAnywhere Actions By Type

Action Type	Description
Install Actions	Available only in the Install sequence, these actions deploy the installer payload to the target machine.
Uninstall Actions	Available only in the Uninstall sequence, you can use these actions to customize the flow of the InstallAnywhere uninstaller.
General Actions	Available in the Install sequence, as well as the Pre-Install, Post-Install, Post-Install, and Post-Uninstall sequences, 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 sequences, 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 `<InstallAnywhere>/plugins` folder.



Tip ▪ *Action customizers share many properties and controls. See Common Action Settings 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 sequence, Install actions deploy the installer payload to the target machine. The following table lists the Install actions available in InstallAnywhere.

Table 9-2 ▪ Install Actions

Action	Description
Create Alias, Link, Shortcut Action	Create an alias (OS or OS X), symbolic link (UNIX and Linux), or shortcut (Windows).
Create Folder Action	Create a new folder on the end user's system.
Create LaunchAnywhere for Java Application Action	Create a launcher to start the installed Java application.
Create Uninstaller Action	Create the uninstaller and several additional files needed by the uninstaller.
Deploy WAR/EAR Archive Action	Deploy a WAR or EAR archive to an application server.
Download File Action	Download a file during installation using FTP, HTTP, HTTPS, SFTP, or Anonymous FTP protocol.
Enable Update Notifications Action	Include the FlexNet Connect Java agent in installers built from this project.
Expand Archive Action	Expand an archive file (.zip, .jar, .sit, .war, or .ear) or decode a Binary file (.bin) on target systems.  ***Important ▪*** *.zip files must be zipped with the DEFLATE compression method. No other compression method is supported.*
Expand Archive (7-zip) Action	Expand a 7-ZIP archive file (`*.7z` or `*.xz`) with the LZMA and LZMA2 compression methods on target systems.
Expand Archive (TAR) Action	Expand a TAR file (`.tar`) on target systems.
Install Archive Action	Install a ZIP file (.zip, .jar) on target systems.
Install File Action	Install a file from the installer onto target systems.
Install from Manifest Action	Install all of the files and folders specified in the manifest file on the end user's system. See Manifest Files for more information.

Table 9-2 ▪ Install Actions (cont.)

Action	Description
Install HP-UX Depot Action	Install and uninstall HP-UX depot files. You must specify the package name within the depot file, since it may contain multiple packages. In order to install multiple packages in the same depot, add one action for every package that you want to install (This method does not increase the size of the installer).
Install Linux Package Action	Install and uninstall Linux RPM Package Manager (.rpm) files, Debian (.deb) packages, or a file you want to install from a default or a custom repository. The package files can either be bundled with the installer or pre-existing on the system. If you choose to install from a repository, there are two options: ● Install from Repository—Install from the default distribution repository on the machine you are targeting ● Install from Repository / Custom Repository—Install from a specified location of a custom repository. When you click the Custom Repository check box, the Choose Repository button is enabled, allowing you to search for the custom repository location, choosing either a *.list file for Debian file distributions or *.repo for Linux. If the RPM or DEB is relocatable, and the Relocatable check box is selected in the action customizer, the file is installed to its location in the file tree. Additionally, the RPM or DEB can be set to Ignore Dependencies (similar to the `--nodeps` option for the command-line RPM tool) and to Force Installation (`--force`). You can also choose Do Not Uninstall. For custom repositories, the only check box that is available is to Do Not Uninstall. The Install Linux Package action allows you to add only one package at a time. If you want to add additional packages, click Add Action again.
Install Merge Module Action	Install a merge module as if the merge module were run as a separate silent installer.
Install Solaris Package Action	Install and uninstall Solaris package files. 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 Action	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.
Move File Action	Move or rename a file from one location to another location on the end user's system.

Table 9-2 ▪ Install Actions (cont.)

Action	Description
Move Folder Action	Move or rename a folder from one location to another location on the end user's system.
Set System Environment Variable Action	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 Action	Run a SQL script on a database server.
Trigger Rollback Action	Add to the Install sequence to specifically initiate a rollback if a certain rule or condition is met.

Create Alias, Link, Shortcut Action

You can use the Create Alias, Link, Shortcut action to create an alias (OS or 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 9-3 ▪ Create Alias, Link, Shortcut Action Customizer

Property	Description
Destination	Specify the following information to identify the destination information: • **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. • **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 9-3 ▪ Create Alias, Link, Shortcut Action Customizer (cont.)

Property	Description
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 this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Create Folder Action

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 9-4 ▪ Create Folder Action Customizer | Properties Tab

Property	Description
Destination	Specify the following information to identify the destination information: • **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.
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 executable files 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.*

Table 9-4 ▪ Create Folder Action Customizer | Properties Tab (cont.)

Property	Description
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 / 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 this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Create LaunchAnywhere for Java Application Action

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 9-5 ▪ 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. You can use either the magic folder list or the Visual Tree's arrow buttons to change the path.
Name	Sets the name for the launcher.

Table 9-5 ▪ Create LaunchAnywhere for Java Application | Properties > General Settings Tab (cont.)

Option	Description
Main Class	Specifies the main class in your Java application. This is the class that the launcher invokes. To choose from a list of the available classes in the project, click the Automatically Find Main Classes button next to this setting.
Arguments	Specifies the list of arguments to pass to the Java application's main method. The default value is `$CMD_LINE_ARGUMENTS$`.
Launcher Type	Defining the launcher type allows you to specify whether the Java application is a GUI-based application that calls `javaw.exe` (stdout and stderr suppressed by default), or a console application that calls `java.exe` (stdout and stderr directed to the console by default). In addition, on Windows-based systems, console launchers redirect stdout and stderr output to the same console window from which they were invoked. If your application has a graphical interface, select Graphical Launcher. Otherwise, select Console Launcher.
Icon	To select a different icon for the LAX executable file on OS or OS X–based and Windows-based target systems, click the Change button next to the icon image. To revert the launcher icon to the InstallAnywhere default image, click the Default button. Custom icons are not available for UNIX-based installers. OS or OS X icons can be 128 X 128. On Windows-based target systems, the LaunchAnywhere executable file does not display the specified icon directly. Instead, the LaunchAnywhere executable file has the generic .exe icon, and an .ico file is placed in the same folder as the executable file; all shortcuts that are created during installation and that target the LaunchAnywhere inherit the custom icon. InstallAnywhere does not support interlaced GIF files for the icon.
Launcher Properties	To configure various settings for the launcher, click the Edit Properties button.

Table 9-5 ▪ Create LaunchAnywhere for Java Application | Properties > General Settings Tab (cont.)

Option	Description
Windows Execution Level	Select the execution level that your application requires for Windows-based systems. Available options are: • **As Invoker**—The launcher is run with the same execution level as its parent process. This execution level is typically standard user privileges, since Windows Explorer runs as a standard user. If the launcher does not require administrative privileges and all users can run it without administrative privileges, select this option. • **Highest Available**—The launcher is run with the highest Windows privileges and user rights that are available to the current user. Administrators must authorize it; non-administrators run it without administrative privileges. • **Administrator**—The launcher requires local administrative 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/ OS X Permission	To set the file permissions to a specific value for this launcher, select this check box. 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 9-6 ▪ Create LaunchAnywhere for Java Application | Properties > Advanced Settings Tab

Option	Description
Configure LaunchAnywhere with	Select one of the following options: • 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)  ***Note ▪*** *See Java Virtual Machines for a details about VM selection. See Creating Launchers for Java Applications for instructions on using this action.*

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Create Uninstaller Action

InstallAnywhere automatically adds this Create Uninstaller action to create an uninstaller. The Create Uninstaller action is basically a special LaunchAnywhere executable file 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 9-7 ▪ Create Uninstaller | Properties > General Settings Tab

Option	Description
Path	Identifies the location of the uninstaller executable file to be created on the target system.  ***Tip*** ▪ *Install the uninstaller into its own folder.*
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.

Table 9-7 ▪ Create Uninstaller | Properties > General Settings Tab (cont.)

Option	Description
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	Select the execution level that the launcher requires for Windows-based systems. Available options are: • **As Invoker**—The launcher is run with the same execution level as its parent process. This execution level is typically standard user privileges, since Windows Explorer runs as a standard user. If the launcher does not require administrative privileges and all users can run it without administrative privileges, select this option. • **Highest Available**—The launcher is run with the highest Windows privileges and user rights that are available to the current user. Administrators must authorize it; non-administrators run it without administrative privileges. • **Administrator**—The launcher requires local administrative 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.
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/ 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 9-8 ▪ Create Uninstaller | Properties > Advanced Settings Tab

Option	Description
Configure LaunchAnywhere with	Select one of the following options: • 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)  ***Note ▪*** *See Java Virtual Machines for a details about VM selection. See Creating Launchers for Java Applications for instructions on using this action.*

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Deploy WAR/EAR Archive Action

The Deploy WAR/EAR Archive action deploys a Web application to an application server. This action requires an existing application server host in the Hosts view on the Organization page. Once the application server has been added in the Hosts view, you can add the Deploy WAR/EAR Archive action to an application server host in a view on the Sequence page.

Properties Tab

The Properties tab of the Deploy WAR/EAR Archive action customizer has the following subtabs:

- General Settings Subtab
- Optional Settings Subtab
- Tomcat Settings Subtab
- WebSphere Settings Subtab

General Settings Subtab

The General Settings subtab on the Properties tab includes the following settings:

Table 9-9 ▪ General Settings Tab in the Deploy WAR/EAR Archive Action Customizer

Setting	Description
Application Name	Enter a name to identify the archive file.
Source	Specify the source for the EAR or WAR file. Available options are: • **Selected Archive**—To bundle the EAR or WAR file with the installer, select this check box and click the Choose Archive button to select the appropriate archive. • **Existing Archive**—To deploy an EAR or WAR file that is on target systems, select this check box and enter the path and file name for the archive on the target system.

Table 9-9 ▪ General Settings Tab in the Deploy WAR/EAR Archive Action Customizer (cont.)

Setting	Description
Undeploy During Uninstall	To ensure that the archive is undeployed when the uninstaller runs, select this check box. This check box is selected by default. Note that undeploying WAR files is not supported for remote Tomcat servers.

Optional Settings Subtab

The Optional Settings subtab is available for the following types of Web servers: Geronimo, JBoss, and WebLogic.

The Optional Settings subtab on the Properties tab includes the following settings:

Table 9-10 ▪ Optional Settings Tab in the Deploy WAR/EAR Archive Action Customizer

Setting	Description
Deployment Plan	For servers that support the JSR-88 standard, you can optionally choose an archive for the deployment plan. A deployment plan is an XML file that contains server-specific information that is not included in the EAR or WAR file's deployment descriptor. Available options are: ● **Selected Archive**—To choose an archive that you want to bundle with the installer, select this check box and click the Choose Archive button to select the appropriate deployment plan file. ● **Existing Archive**—To choose an archive that is on target systems, click the Existing Archive button and enter the path and file name for the deployment plan file on target systems.

Tomcat Settings Subtab

The Tomcat Settings subtab is applicable for Tomcat application server hosts.

The Tomcat Settings subtab on the Properties tab includes the following settings:

Table 9-11 ▪ Tomcat Settings Tab in the Deploy WAR/EAR Archive Action Customizer

Setting	Description
Deployment Option	Specify what kind of server to use for deploying WAR files. By default, the entry in this setting is the user-defined variable $TOMCAT_DEPLOYMENT_OPTION$. You can specify a different variable if needed. To learn more, see Specifying Which Deployment Options to Support for Apache Tomcat Servers.

Table 9-11 ▪ Tomcat Settings Tab in the Deploy WAR/EAR Archive Action Customizer

Setting	Description
Server Path	Enter the path on the server where you want to deploy the Web application. By default, the entry in this setting is the user-defined variable $TOMCAT_SERVER_PATH$. You can specify a different variable if needed. This setting applies to local Tomcat servers; this setting is available if the Local Tomcat check box is selected for the associated Tomcat server host.
Host Name	Enter the name of the Tomcat server or its IP address. By default, the entry in this setting is the user-defined variable $TOMCAT_HOSTNAME$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.
Port	Enter the port number on which Tomcat is running. By default, the entry in this setting is the user-defined variable $TOMCAT_PORT$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.
User Name	Enter the user name for the account that is specified in the `tomcat-users.xml` file (which is in the conf folder in the Tomcat directory). By default, the entry in this setting is the user-defined variable $TOMCAT_USERNAME$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.
Password	Enter the password that corresponds with the specified user name for an account that is specified in the `tomcat-users.xml` file (which is in the conf folder in the Tomcat directory). By default, the entry in this setting is the user-defined variable $TOMCAT_PASSWORD$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.

Table 9-11 ▪ Tomcat Settings Tab in the Deploy WAR/EAR Archive Action Customizer

Setting	Description
SSL Enabled	Indicate whether SSL is enabled on the remote Tomcat server. By default, the entry in this setting is the user-defined variable $TOMCAT_SSL_CONNECTION_STATUS$. You can specify a different variable if needed. If this variable is set to Yes at run time, the action uses SSL for deployment to the server. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.

WebSphere Settings Subtab

The WebSphere Settings subtab is applicable for WebSphere application server hosts.

The WebSphere Settings subtab on the Properties tab includes the following settings:

Table 9-12 ▪ WebSphere Settings Tab in the Deploy WAR/EAR Archive Action Customizer

Setting	Description
Deployment Option	Specify what kind of server to use for deploying EAR or WAR files. By default, the entry in this setting is the user-defined variable $WEBSPHERE_DEPLOYMENT_OPTION$. You can specify a different variable if needed. To learn more, see Specifying Which Deployment Options to Support for IBM WebSphere Servers.
Host Name	Enter the name of the WebSphere server or its IP address. By default, the entry in this setting is the user-defined variable $WEBSPHERE_HOSTNAME$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
SOAP Port	Enter the SOAP port number on which WebSphere is running. By default, the entry in this setting is the user-defined variable $WEBSPHERE_PORT$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.

Table 9-12 ▪ WebSphere Settings Tab in the Deploy WAR/EAR Archive Action Customizer

Setting	Description
Admin Security Enabled	Enter the value that indicates whether administrative security is enabled on the WebSphere application server. If this variable is set to Yes at run time, the action uses the credentials in the User Name and Password settings to connect to the server. By default, the entry in this setting is the user-defined variable $WEBSPHERE_IS_SECURITY_ENABLED$. You can specify a different variable if needed. The available values are Yes and No. This setting applies to remote WebSphere servers.
User Name	Enter the user name for the account that should be used to connect to the WebSphere server if administrative security is enabled. By default, the entry in this setting is the user-defined variable $WEBSPHERE_USERNAME$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
Password	Enter the password that corresponds with the specified user name for an account that should be used to connect to the WebSphere server. By default, the entry in this setting is the user-defined variable $WEBSPHERE_PASSWORD$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
SSL TrustStore File Path	Specify the path to the SSL TrustStore file. By default, the entry in this setting is the user-defined variable $WEBSPHERE_TRUSTSTORE_PATH$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
SSL TrustStore Password	Specify the password to the SSL TrustStore. By default, the entry in this setting is the user-defined variable $WEBSPHERE_TRUSTSTORE_PASSWORD$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.

- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Download File Action

You can use the Download File action to download a file during installation using FTP, HTTP, HTTPS, SFTP, or Anonymous FTP protocol. You specify information regarding this action on the Download File Action customizer.

Properties Tab

The Properties tab of the Download File Action customizer includes the following options:

Table 9-13 ▪ Download File Action Customizer

Option	Description
Choose Protocol Type	Select one of the following protocol types to identify the method that will be used to download the specified file: • FTP • HTTP • HTTPS • SFTP • Anonymous FTP
Enter URL	Enter the URL of the file that will be downloaded.
User Name **Password**	Enter the login credentials for the selected protocol type.  ***Note ▪*** *If HTTP or HTTPS is selected for Choose Protocol Type, the User Name and Password fields are disabled. If Anonymous FTP is selected, the User Name field is disabled but the Password field is enabled.*  ***Important ▪*** *To encrypt the password using Project Automation, use the following API call and then set it to the action:* `ProjAutoRuntimeFacade.getEncryptedPassword(String <password_to_be_encrypted>, String <absolute_project_file_path>)`
Size of the file (in bytes)	Enter the size of the file to be downloaded. This helps in determining the necessary total space required by the installer for complete installation.

Table 9-13 ▪ Download File Action Customizer (cont.)

Option	Description
Port	Enter the appropriate port number for the selected protocol type. The default values are as follows: • FTP—Port 21. • HTTP—Port 80. • HTTPS—Port 443. • SFTP—Port 22. • Anonymous FTP—Port 21.
Destination	Specify the following information to identify the destination information: • **Path**—Specify the location on the target machine where the downloaded file will be installed. 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 downloaded file.
Show indeterminate dialog	Choose this option to show a message dialog while the file is downloading that displays the message you enter in the associated text box. The default message is `Downloading File...`
Do not uninstall	Select this option if you do not want the downloaded file to be uninstalled when the application is uninstalled.

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Enable Update Notifications Action

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 automatically notifies your Web-connected end users when patches, updates, and product information for your product are available. FlexNet Connect helps you reduce the number of end users running old releases of your products and prevent them from installing the wrong updates from your Web site.

By default, the Enable Update Notifications action also creates a launcher for the Java agent so that your end users have the option of checking for updates on demand.



***Note** ▪ The use of the Enable Update Notifications action requires a FlexNet Connect account in either a Revenera–hosted or self-hosted environment. You must create a product on the FlexNet Connect Publisher site with a product code that matches the product code that is configured in your project (in the Advanced Designer, on the Project page, click the General Settings link).*

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 9-14 ▪ Enable Update Notifications Customizer | Properties > General Settings

Option	Description
Have you registered your product with FlexNet Connect?	Opens the login page for the Revenera-hosted Publisher site. You must register your product with FlexNet Connect to take advantage of the Enable Update Notifications action. 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.
Destination for FlexNet Connect files	Defines the install location for FlexNet Connect files. This shows where your installer will place the `us.jar` file (source: `<InstallAnywhere>\isus\us.jar`). Click the drop-down list to specify the location you want the FlexNet Connect files to be installed.  ***Note** ▪ 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.*

Table 9-14 ▪ Enable Update Notifications Customizer | Properties > General Settings (cont.)

Option	Description
Create LaunchAnywhere to execute update check	Select this option to enable or disable the creation of a LaunchAnywhere executable file. By default, this option is selected. If you do not want to include a LaunchAnywhere executable file 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 file that initiates the check for notifications.
Windows Execution Level	Select the execution level that the Update Notifications launcher requires for Windows-based systems. Available options are: • **As Invoker**—The launcher is run with the same execution level as its parent process. This execution level is typically standard user privileges, since Windows Explorer runs as a standard user. If the launcher does not require administrative privileges and all users can run it without administrative privileges, select this option. • **Highest Available**—The launcher is run with the highest Windows privileges and user rights that are available to the current user. Administrators must authorize it; non-administrators run it without administrative privileges. • **Administrator**—The launcher requires local administrative 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 9-15 ▪ 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 General Settings view on the Project page. 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.
FlexNet Connect Server Settings	Define the server on which you have a FlexNet Connect account. Choose **Use Revenera hosted FlexNet Connect server** if you use FlexNet Connect via a Revenera-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 this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.

- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.



Note ▪ *For more information, see Preparing Your Installer for Update Notifications.*

Expand Archive Action

You can use the Expand Archive action to expand an archive file (.zip, .jar, .sit, .war, .ear) or decode a Binary file (.bin) on the end user's system.

The archive files can be installed as part of your installer, or they can be present on target systems.



Important ▪ *.zip files must be zipped with the DEFLATE compression method. No other compression method is supported.*

Properties Tab

The Properties tab of the Expand Archive action customizer includes the following properties:

Table 9-16 ▪ Expand Archive Action Customizer

Property	Description
Source	Specify the archive file that you want to expand. Available options are: • **Selected Archive**—To include the archive in your installer, select this option and then click the Choose Archive button to select the archive that you want to expand. • **Existing Archive**—To expand an archive that is present on target systems, select this option and then enter the path and file name for the archive on target systems. The Expand Archive action supports the following archive file formats: • ZIP files (.zip) • Java archives (.jar) • Binary files (.bin) • StuffIt files (.sit) • WAR files (.war) • Enterprise archives (.ear)

Table 9-16 ▪ Expand Archive Action Customizer (cont.)

Property	Description
Destination	In the Path list in this area, select the platform-specific directory or location. Note that the text field next to the Path list is read-only. If the destination location of the archive should be in a subfolder, use the Create Folder action to create the parent folder. Then move the archive action into the Create Folder action. To learn more, see Adding Folders to a Project.

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Expand Archive (7-zip) Action

You can use the **Expand Archive (7-zip)** action to expand a 7-Zip archive (.7z or .xz) with the LZMA and LZMA2 compression methods. This action is available in the Install phase.

The archive file can be installed as part of your installer, or it can be present on target systems.

Properties Tab

The Properties tab of the **Expand Archive (7-zip)** action customizer includes the following properties:

Table 9-17 ▪ Expand Archive (7-zip) Action Customizer

Property	Description
Source	Specify the 7-zip archive file that you want to expand. Available options are: • **Selected Archive**—To include the archive in your installer, select this option and then click the Choose Archive button to select the archive that you want to expand. • **Existing Archive**—To expand an archive that is present on target systems, select this option and then enter the path and file name for the archive on target systems.

Table 9-17 ▪ Expand Archive (7-zip) Action Customizer (cont.)

Property	Description
Destination	In the Path list in this area, select the platform-specific directory or location. Note that the text field next to the Path list is read-only. If the destination location of the archive should be in a subfolder, use the Create Folder action to create the parent folder. Then move the archive action into the Create Folder action. To learn more, see Adding Folders to a Project.

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Expand Archive (TAR) Action

You can use the Expand Archive (TAR) action to expand a TAR archive file (.tar, .gz, .Z) on the target system. This action has the advantage of preserving the permissions of all files that are part of the TAR archive.

The archive file can be installed as part of your installer, or it can be present on target systems.

Properties Tab

The Properties tab of the Expand Archive (TAR) action customizer includes the following properties:

Table 9-18 ▪ Expand Archive (TAR) Action Customizer

Property	Description
Source	Specify the TAR archive file that you want to expand. Available options are: • **Selected Archive**—To include the archive in your installer, select this option and then click the Choose Archive button to select the archive that you want to expand. • **Existing Archive**—To expand an archive that is present on target systems, select this option and then enter the path and file name for the archive on target systems.

Table 9-18 ▪ Expand Archive (TAR) Action Customizer (cont.)

Property	Description
Destination	In the Path list in this area, select the platform-specific directory or location. Note that the text field next to the Path list is read-only. If the destination location of the archive should be in a subfolder, use the Create Folder action to create the parent folder. Then move the archive action into the Create Folder action. To learn more, see Adding Folders to a Project.

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Install Archive Action

The Install Archive action installs 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 you add a ZIP file to your project.

Install File Action

The Install File action installs a file onto target systems. The Install File customizer includes the following options:

Table 9-19 ▪ 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 / 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 9-19 ▪ Install File Customizer (cont.)

Option	Description
If the file already exists on the end user's system	Select one of the following options to define installer behavior when it attempts to install a file that already exists on the target system: • 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  ***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* `-fileOverwrite` *command. See Recording User Response to File Overwrite Prompts in Response File for more information.*



Note ▪ *This action is not available from the Choose an Action dialog box, but its customizer appears automatically when files are added.*

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Install HP-UX Depot Action

You can use the Install HP-UX Depot action to Install and uninstall HP-UX depot files. A depot file may include mutiple packages, thus package name is required within the depot file. In order to install multiple packages in the same depot, add one action for every package that you want to install.

Properties Tab

The Properties tab of the Install HP-UX Depot customizer includes the following properties:

Table 9-20 ▪ Install HP-UX Depot Customizer

Option	Description
Bundled with installer	To bundle a depot file with the installer, select this option and then click the **Choose depot** button to select a depot file.
On end users's sytem	Select this option to specify the path to the depot file that you want to bundle.
Package Name	Select this option to enter the package name.
Show Indeterminate Dialog	Select this option to show a message dialog while the depot file is downloading that displays the message you enter in the associated text box. The default message is `Installing...`
Do Not Uninstall	Select this option to prevent the uninstaller from removing the downloaded depot file from the target system .

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Install Merge Module Action

Use the Install Merge Module 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 9-21 ▪ 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.

Table 9-21 ▪ Install Merge Module Customizer (cont.)

Option	Description
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.
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	Specify how you want the merge module's installation progress to be displayed: ● To display an intermediate progress bar on a separate pop-up dialog box while this merge module is being installed, select this check box. ● To suppress the separate merge module progress bar and integrate the progress of the merge module installation into the progress bar of the parent installer, clear this check box.
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	Select this option to uninstall this merge module when the main project gets uninstalled.  ***Tip ▪*** *The point at which merge modules are uninstalled can also be configured using the Uninstall Merge Modules action in the Uninstall sequence.*

Table 9-21 ▪ Install Merge Module Customizer (cont.)

Option	Description
Add Merge Module log to parent log	Select this option to append the Merge Module installation log to the main project log. By default, this option will not be selected.  ***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.*  ***Note*** ▪ *You can append merge module logs to the parent log only if Plain text format is selected in the Log Format setting in the General Settings view on the Project page. An XML log format is not supported.*

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Install Solaris Package Action

You can use the Install Solaris Package action to install and uninstall Solaris package files. These packages can either be bundled with the installer or pre-existing on the system.

Properties Tab

The Properties tab of the Install Solaris Package customizer includes the following properties:

Table 9-22 ▪ Install Solaris Package Customizer

Option	Description
Bundled with installer	To bundle a package with the installer, select this option and then click the **<No bundled package chosen>** button to select a package file.
On end users's sytem	Select this option to specify the path to the package file that you want to bundle.
Package Name	Select this option to enter the package name.

Table 9-22 ▪ Install Solaris Package Customizer

Option	Description
Response File	To bundle the response file for the package with the installer, select this option and click the **Choose file** button to select the response file.
Admin File	To bundle the admin file for the package with the installer, select this option and click the **Choose file** button to select the admin file.
Show Indeterminate Dialog	Select this option to show a message dialog while the package file is downloading that displays the message you enter in the associated text box. The default message is `Installing...`
Do Not Uninstall	Select this option to prevent the uninstaller from removing the downloaded package file from the target system .

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Install SpeedFolder Action

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 9-23 ▪ 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: ● Choose a platform-specific directory or location from the list. ● Enter subdirectory information in the box.

Table 9-23 ▪ Install SpeedFolder Customizer (cont.)

Option	Description
Options	Select one of the following options to specify whether to install the folder and its contents or only the folder's contents. • 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: • 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.
In classpath	Puts the item on the classpath for all LaunchAnywhere executable files 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.
Override default UNIX/ OS X permissions	Sets the file permissions to a specific value for this action.

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Set System Environment Variable Action

You can use the **Set System Environment Variable** action to set the value for an environment variable on the end user's system. This action is compatible with Windows and UNIX platform only.

Properties Tab

The Properties tab of the **Set System Environment Variable** Action customizer includes the following options:

Table 9-24 ▪ Set System Environment Variable Action customizer

Option	Description
Variable Name	Specify the name of the environment variable whose value you want to create or modify. Name may be either a variable (such as *$ENV_VARIABLE_NAMES$*) or a literal (such as CLASSPATH)
Set Value To	Specify the value for the environment variable (specified in the **Variable Name** setting). Value may be either a variable (such as *$USER_MAGIC_1$*) or a literal (such as `C:\Program Files`)
When Setting This Variable	Use this option to instruct the installer to adjust the value in the **Set Value To** setting relative to the specified environment variable's existing value. Available options are: • **Replace existing value**—Select this option to replace the value of specified environment variable with the new value that is specified in the **Set Value To** setting. • **Prepend to existing value**—Select this option to add the new value that is specified in the **Set Value To** setting to the beginning of the existing value. • **Append to existing value**—Select this option to add the new value that is specified in the **Set Value To** setting to the end of the existing value.
Set This Variable For	Use this option to instruct the installer to set the value of environment variable for only the current user or all users. Available options are: • **Current user**—Select this option to set the value of specified environment variable for the current user only. • **All users**—Select this option to set the value of specified environment variable for all users.

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Run SQL Script Action

The Run SQL Script action runs a SQL script on a local or remote database server. This action requires an existing database server host in the Hosts view on the Organization page. Once the database server has been added in the Hosts view, you can add the Run SQL Script action to a database server host in a view on the Sequence page.

The Run SQL Script action enables you to optionally create a SQL database on MySQL servers. When you are configuring this action, you can specify separate SQL scripts (install script and uninstall script) to run during installation and uninstallation.



Tip ▪ *The Choose Database Connection panel action and the Choose Database Connection console action enable end users to specify information for connecting to the database on which the SQL script is going to be run on databases. For more information, see Enabling End Users to Specify Database Connection Information.*

Properties Tab

The Properties tab of the Run SQL Script action customizer has the following subtabs:

- Server Settings Subtab
- Install Subtab
- Uninstall Subtab

Server Settings Subtab

The Server Settings subtab on the Properties tab includes the following settings:

Table 9-25 ▪ Server Settings Tab in the Run SQL Script Action Customizer

Setting	Description
Server Host	Specify the server host to which your installer is connecting. By default, the entry in this setting is the user-defined variable $DB_SERVERHOST_VARIABLE$. You can specify a different variable if needed.
Server Port	Specify the port for the server connection. By default, the entry in this setting is the user-defined variable $DB_SERVERPORT_VARIABLE$. You can specify a different variable if needed.

Table 9-25 ▪ Server Settings Tab in the Run SQL Script Action Customizer (cont.)

Setting	Description
Database Name	Specify the name of the database on which your installer is running the SQL script. By default, the entry in this setting is the user-defined variable $DB_NAME_VARIABLE$. You can specify a different variable if needed.
User Name	Specify the user name for the account that should be used to connect to the server through server authentication. By default, the entry in this setting is the user-defined variable $DB_USERNAME_VARIABLE$. You can specify a different variable if needed.
Password	Specify the password that corresponds with the specified user name for the account that should be used to connect to the server through server authentication. By default, the entry in this setting is the user-defined variable $DB_PASSWORD_VARIABLE$. You can specify a different variable if needed.
Create Database	 ***Note ▪*** *This check box is applicable for MySQL, Microsoft SQL Server, and PostgreSQL databases.* To have the Run SQL Script action create the database on target systems before running the SQL script, select this check box. This check box is cleared by default. Note the following details about the run-time behavior that occurs if you select this check box: ● The user account that is used to connect to the server must have database administrator privileges. ● The action creates a basic database with default parameters. For example, for a MySQL database, the default character set and collation are used. For DB2, default values of the following parameters are used: buffer pool name for table spaces, buffer pool name for indexes, the storage group, and the encoding scheme for database data. ● The action does not verify whether the database name already exists for a database on the target system.

Install Subtab

The Install subtab on the Properties tab includes the following settings:

Table 9-26 ▪ Install Tab in the Run SQL Script Action Customizer

Setting	Description
Selected Script	To specify a script file to bundle with your installer, select this option. 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, select this option. In the Existing Script text box, enter the path to the script file that you want your installer to run.
SQL Statements Delimiter	Identify the character that separates SQL statements. To override the default delimiter (a semicolon), enter 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*** ▪ *To view the currently selected script if you are bundling the script with your installer, click the View Script button. Note that the SQL script cannot be modified in the SQL Script Preview dialog box.*

Uninstall Subtab

The Uninstall subtab on the Properties tab includes the following settings:

Table 9-27 ▪ Uninstall Tab in the Run SQL Script Action Customizer

Setting	Description
Selected Script	To specify a script file to bundle with your uninstaller, select this option. Click Choose File to open the Choose a File dialog box. Then, browse to the script file you want your uninstaller to run and click Open.
Existing Script	To specify a script file that exists on the target system, select this option. In the Existing Script text box, enter the path to the script file that you want your uninstaller to run.

Table 9-27 ▪ Uninstall Tab in the Run SQL Script Action Customizer (cont.)

Setting	Description
SQL Statements Delimiter	Identify the character that separates SQL statements. To override the default delimiter (a semicolon), enter 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 ▪*** *To view the currently selected script if you are bundling the script with your uninstaller, click the View Script button. Note that the SQL script cannot be modified in the SQL Script Preview dialog box.*

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Trigger Rollback Action

You can add a Trigger Rollback action to the Install sequence 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.

Task ***To add a Trigger Rollback action to the Install sequence:***

1. In the Advanced Designer, on the **Sequence** page, click **Install**. The **Install** view opens.
2. Click the **Add Action** button. The **Choose an Action** dialog box opens.
3. Click the **Trigger Rollback Action** action, and then click the **Add** button. InstallAnywhere adds the action to the **Visual Tree**. The **Choose an Action** dialog box remains open, enabling you to add additional actions as needed.
4. Click the **Rules** tab.
5. Click the **Add Rule** button.
6. Add the rule or rules that you want to use to define the conditions for rollback.

For a description of all available rules, see Rules Reference.

If the conditions that are defined in the rules are not met, the installation is rolled back.

Uninstall Actions

The following table lists the Uninstall actions that are available in InstallAnywhere. You can use them to customize the flow of the InstallAnywhere uninstaller. These actions are available only in the Uninstall sequence.

Table 9-28 ▪ Uninstall Actions

Action	Description
Uninstall Category Action	Groups sets of uninstall actions.
Uninstall AIX Entries Action	Uninstalls all entries made in the SWVPD registry of the AIX machine.
Uninstall DB Scripts Action	Runs the Uninstall scripts mentioned in the Run SQL action against the databases.
Uninstall Files Action	Uninstalls all of the files installed or created by the installer during installation.
Uninstall Folders Action	Uninstalls all of the folders installed or created by the installer during installation.
Uninstall JEE Archives Action	Undeploys WAR/EAR files deployed during installation.
Uninstall LaunchAnywheres Action	Uninstalls all of the LaunchAnywheres installed or created by the installer during installation.
Uninstall Merge Modules Action	Uninstalls all associated merge modules installed by a parent installer during installation. For more information, see Uninstaller for Merge Modules and Multiple Products.
Uninstall RAIR Entries Action	Uninstalls the RAIR component entries in i5/OS machines.
Uninstall Registry Entries Action	Uninstalls all of the registry entries installed or created by the installer during installation.
Uninstall RPM Packages Action	Uninstalls all RPM entries made by the installer.
Uninstall Shortcuts/ Links/Aliases Action	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 9-29 ▪ General Actions

Action	Description
Action Group Action	Adds a folder to the Action List in which you can group sets of InstallAnywhere actions. This action is available in the Install sequence as well as the Pre-Install, Post-Install, Pre-Uninstall, and Post-Uninstall sequences.  ***Note*** *▪ For more information, see Assigning a Rule to a Group of Actions.* **Comment Action Groups** To prevent an action group (and all of the actions that it contains) from being bundled with the installer, select the Comment Action Group option on the Properties tab of the Action Group customizer. When this option is selected, a overlay icon appears on top of the action group icon in the Visual Tree to indicate its change in status:  The Comment Action Group option gives you an easy way to retain the actions in an action group in your installation project but to not include them in the built installer. If, at some future time, you wish to include these actions in the installer, you can clear the selection of the Comment Action Group option.
Add Jump Label Action	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. Use this action in conjunction with the **Jump to Target** action. (Available during the Pre-Install and Post-Install sequences.)
Comment Action	The Comment action is designed to allow developers to add a simple comment to the installer.
Copy File Action	Copy a file from one location to another location on the end user's system.
Copy Folder Action	Copy a folder from one location to another location on the end user's system.

Table 9-29 ▪ General Actions (cont.)

Action	Description
Delete File Action	Delete a file from the end user's system. • **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 Multiple Files**—Select the checkbox to delete multiple files with the same extension from the target system, and enter the path (fixed or relative), wildcard, and file extension for the files you want to delete.
Delete Folder Action	Delete a folder from the end user's system.

Evaluate Dependencies Action	Evaluate dependencies on which the installer is based. This action sets the following InstallAnywhere variables:
	`$DEPENDENCY_SUCCESSES$`
	`$DEPENDENCY_FAILURES$`
	`$DEPENDENCY_REPORT$`
	`$DEPENDENCY_STATUS$`

Execute Ant Script Action	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 Action	Execute Command allows developers to run executable files as they would from the target system's command line.
Execute Custom Code Action	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 Action	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 9-29 ▪ General Actions (cont.)

Action	Description
Execute Target File Action	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.  ***Note ▪*** *This action is available only during the installation of files and after files have been installed.*  ***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.*
Execute Uninstaller Action	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.
Find Component in InstallAnywhere Registry Action	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.
Get Windows Registry Entry Action	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.  ***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.*
Jump to Target Action	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 sequences.)
Launch Default Browser Action	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.  ***Note ▪*** *This action is available during the Pre-Install, Post-Install, Pre-Uninstall, and Post-Uninstall sequences.*

Table 9-29 ▪ General Actions (cont.)

Action	Description
Modify Text File - In Archive Action	This action alters text files within an archive (ZIP or JAR).
Modify Text File - Multiple Files Action	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 Action	This action modifies a text file on the target system.
Output Debug Information Action	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. Output data options include: ● **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. Output destination options include: ● **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. For more information, see Using the Output Debug Information Action.
Output Text to Console Action	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.  ***Note*** ▪ *This action is available in the Pre-Install, Install, and Post-Install sequences.*
Perform XSL Transform Action	This action allows developers to specify an XSLT and target for an Extensible Stylesheet Language transform. Predefined XSL Transforms can be found in `<InstallAnywhere>\resource\extras\presets`

Table 9-29 ▪ General Actions (cont.)

Action	Description
Perform XSL Transform - In Archive Action	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 Action	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 Action	This action enables developers to read or modify XML files on the target system.
Refresh Windows Environment Action	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 Action	Registers a Windows Service on the end user's system.  ***Note ▪*** *This action is available only in the Install sequence.*
Restart Windows Action	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 Install, Post-Install, Pre-Uninstall, and Post-Uninstall sequences.*
Set InstallAnywhere Variable - Multiple Variables Action	Use this action to create, and specify values for InstallAnywhere variables. This action can be used to control nearly any aspect of the installation. To learn more, see Evaluation of InstallAnywhere Variables and Setting Variables in the Advanced Designer.  ***Important ▪*** *Run-time variables are not supported for the uninstaller on OS or OS X–based systems. To learn more, see About Uninstallers and Variables.*
Set InstallAnywhere Variable - Single Variable Action	Set the value of a single InstallAnywhere variable. To learn more, see Evaluation of InstallAnywhere Variables and Setting Variables in the Advanced Designer.  ***Important ▪*** *Run-time variables are not supported for the uninstaller on OS or OS X–based systems. To learn more, see About Uninstallers and Variables.*

Table 9-29 ▪ General Actions (cont.)

Action	Description
Set Windows Registry - Multiple Entries Action	Set multiple Windows registry keys, data, and values on target systems.  ***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.*
Set Windows Registry - Single Entry Action	Set an individual Windows registry key, data, and value on target systems.  ***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.*
Show Message Dialog Action	This action creates a modal dialog that requests end user input. The message dialog box appears over the currently displayed panel. For more information, see Show Message Dialog Action.
Start, Stop, Pause Windows Service Action	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.
Uninstall InstallShield Universal Software Action	Uninstalls software previously installed by InstallShield Universal or InstallShield MultiPlatform installers.
Enable 8.3 File Naming Action	This action enables 8.3 naming on Windows target systems for all volumes and enables 8.3 file creation permanently for whole system when Installer has elevated privileges. This action can be performed in any sequence phase.
Associate File Extension/URL Prefix - Windows Action	Associates extensions to Installed Applications and existing Applications on Windows. New extensions or existing extensions can be associated.

Delete Folder Action

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 9-30 ▪ 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. • **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 this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Execute Ant Script Action

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 9-31 ▪ Execute Ant Script Customizer

Option	Description
Ant Build Script	Identify the Ant build script you want to execute by choosing one of the following options: • **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.

Table 9-31 ▪ Execute Ant Script Customizer (cont.)

Option	Description
Specify Build Properties	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.
Substitute IA Variables in Build Properties	Select this option to resolve any InstallAnywhere variables in the build properties file you selected.
Ant Target	To specify an Ant target, choose one of the following options: ● **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	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.
Store process's exit code in	Specify the variable name which is used to store the value or exit code returned from an Ant script during installation. You can specify any customized variable name. By default, variable name is set to **$EXECUTE_ANT_EXITCODE$**.  ***Note ▪*** *By default, an exit code or value is not returned by the Ant script. To return a value or exit code, the variable name specified in this option, must be included in the ANT script by either using the `Exec` tag with the `resultproperty` attribute or by using the `Property` tag with the `name` attribute.* *For instance, if the $EXECUTE_ANT_EXITCODE$ variable name is specified in the **Store process's exit code in** option, the ANT script includes the $EXECUTE_ANT_EXITCODE$ variable name either by using `failonerror="true"resultproperty="EXECUTE_ANT_EXITCODE"` in the `Exec` tag or by using `<property name="EXECUTE_ANT_EXITCODE" value="1" />`.*
Dependencies	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.  ***Note ▪*** *InstallAnywhere automatically adds dependencies required by the Execute Ant Script action.*



Note ▪ *The Execute Ant Script action is only for developers familiar with Ant. For more information, visit the Apache Ant site (http://ant.apache.org/).*

Execute Command Action

You can use the Execute Command action to run executable files as you would from the target system's command line. The Execute Command action customizer has the following options:

Table 9-32 ▪ Execute Command Customizer

Option	Description
Command Line	Enter the syntax to run the executable. Type the name of the executable (command) or the full path to the executable. • 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 `\"`. • All elements in **Command Line** may be represented either as literal values or by InstallAnywhere variables.
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).  ***Note*** ▪ *For more information, see Preventing the Substitution of Unknown Variables.*
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 `Executing...`
Store process's stdout in	Enter the variable to store the `stdout` from your command. The default value is `$EXECUTE_STDOUT$`.
Store process's stderr in	Enter the variable to store the `stderr` from your command. The default value is `$EXECUTE_STDERR$`.
Store process's exit code in	Enter the variable to store exit codes. The default value is `$EXECUTE_EXITCODE$`.

Table 9-32 ▪ Execute Command Customizer (cont.)

Option	Description
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 executable files.



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 Action

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 9-33 ▪ Execute Custom Code Customizer

Option	Description
Path	Identifies the archive that contains all classes needed by the custom code action. • **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`**. 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.
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 sequence.*

Table 9-33 ▪ Execute Custom Code Customizer (cont.)

Option	Description
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 uninstall() method	Choose whether to set the `uninstall()` method for the custom code to run before or after files or folders are uninstalled.  ***Note ▪*** *This option is available only in the Install sequence.*
Dependencies	Custom code tasks sometimes are dependent on 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 Action

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 9-34 ▪ Execute Script/Batch File Customizer

Option	Description
Comment	Enter a comment to describe the function of the script or batch file.

Table 9-34 ▪ Execute Script/Batch File Customizer (cont.)

Option	Description
Script	Type the script you want the installer to run. Start your script with a **`cd <directory>`** command to establish the directory from which the script will run. By default, scripts and batch files are run in a `temp` 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).  ***Note ▪*** *For more information, see Preventing the Substitution of Unknown Variables.*
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 **`Executing Installation Script...`**
Store process's stdout in	Enter the variable to store the `stdout` from your command. The default value is **`$EXECUTE_STDOUT$`**.
Store process's stderr in	Enter the variable to store the `stderr` from your command. The default value is **`$EXECUTE_STDERR$`**.
Store process's exit code in	Enter the variable to store exit codes. The default value is **`$EXECUTE_EXITCODE$`**.

Find Component in InstallAnywhere Registry Action

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 InstallAnywhere Registry action to locate that component. The Find Component in InstallAnywhere 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 use 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 InstallAnywhere Registry action customizer includes the following options:

Table 9-35 ▪ Find Component in InstallAnywhere 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 `$REG_COMPONENT_LOCATION$`.
Search in both 32-bit and 64-bit InstallAnywhere registries	If you want the action to search for a component in both 32-bit and 64-bit InstallAnywhere registry locations (Program Files (x86)\Zero G Registry and Program Files\Zero G Registry) on 64-bit target systems, select this check box.

Modify Text File - In Archive Action

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 9-36 ▪ 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. (Available only in the Install and Post-Install sequences.)
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. The path of the text file to modify should be starting at the root of the archive specified as the target of the action. For example, ● META-INF\someProps.properties ● someFolder\file1.txt ● someFolder\someSubFolder\file2.txt  ***Note*** ▪ *The Archive Path value should not be the full, absolute path beginning at the root of the archive. For example: C:\sample\META-INF\someProps.properties.*
Change Line Endings To	Specify the type of line ending to use in the text file: ● User's Platform (Default) ● OS ● UNIX ● DOS/Windows To ensure that the line endings conform to the built-in line endings for the target system, choose User's Platform.

Table 9-36 ▪ Modify Text file - In Archive Customizer (cont.)

Option	Description	
Source File Encoding	Enter the encoding type of the source file that you are modifying. For a list of supported encoding types, view the Supported Encodings page for the version of Java you are using:	
	Java 11	https://docs.oracle.com/en/java/javase/11/intl/supported-encodings.html
	Java 17	https://docs.oracle.com/en/java/javase/17/intl/supported-encodings.html
	Java 21	https://docs.oracle.com/en/java/javase/21/intl/supported-encodings.html
	If the encoding type you enter in the Destination File Encoding field is invalid or unsupported, then system default encoding will be applied. If you leave the Destination File Encoding field blank, then UTF-8 encoding will be attempted. If the installer fails to apply UTF-8, then system default encoding will be applied.  ***Note ▪*** *Note the following regarding file encoding:* ● *Retention of the existing encoding type is not supported.* ● *There will not be any build time validation.* ● *The result of applying an encoding type will be written to the log file.*	

Table 9-36 ▪ Modify Text file - In Archive Customizer (cont.)

Option	Description
Destination File Encoding	Enter the encoding type of the destination file to be saved after it has been modified by the Modify Text File action. For a list of supported encoding types, view the Supported Encodings page for the version of Java you are using: **Java 11** https://docs.oracle.com/en/java/javase/11/intl/supported-encodings.html **Java 17** https://docs.oracle.com/en/java/javase/17/intl/supported-encodings.html **Java 21** https://docs.oracle.com/en/java/javase/21/intl/supported-encodings.html If the encoding type you enter in the Destination File Encoding field is invalid or unsupported, or if you leave this field blank, then UTF-8 encoding will be attempted. If the installer fails to apply UTF-8, then system default encoding will be applied.  ***Note*** ▪ *Note the following regarding file encoding:* • *Retention of the existing encoding type is not supported.* • *There will not be any build time validation.* • *The result of applying an encoding type will be written to the log file.*
Additional Text	Use this option to instruct the installer to add additional text to text file. Choose one of the following options: • **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.
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** texts. When this action runs, the installer locates all instances of the **Search For** text and replaces them with their corresponding **Replace With** text.

Table 9-36 ▪ Modify Text file - In Archive Customizer (cont.)

Option	Description
Warn if search string is not found	If you want InstallAnywhere to log a warning message in the log file whenever a text string—specified under the **Search For** field on the **Configure Search and Replace Strings** dialog box—is not found in the target text file, select this option. The warning message notifies that expected text replacement is not applied to the text file since the specified search string does not exist in that file. By default, this option is cleared
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).

Modify Text File - Multiple Files Action

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 9-37 ▪ 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. Available only in the Install and Post-Install sequences.
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.
Change Line Endings To	Specify the type of line ending to use in the text file: ● User's Platform (Default) ● OS ● UNIX ● DOS/Windows To ensure that the line endings conform to the built-in line endings for the target system, choose User's Platform.

Table 9-37 ▪ Modify Text File - Multiple Files Customizer (cont.)

Option	Description	
Source File Encoding	Enter the encoding type of the source file that you are modifying. For a list of supported encoding types, view the Supported Encodings page for the version of Java you are using:	
	Java 11	https://docs.oracle.com/en/java/javase/11/intl/supported-encodings.html
	Java 17	https://docs.oracle.com/en/java/javase/17/intl/supported-encodings.html
	Java 21	https://docs.oracle.com/en/java/javase/21/intl/supported-encodings.html
	 ***Note*** ▪ *Note the following regarding file encoding:* • *Retention of the existing encoding type is not supported.* • *There will not be any build time validation.* • *The result of applying an encoding type will be written to the log file.*	
Destination File Encoding	Enter the encoding type of the destination file to be saved after it has been modified by the Modify Text File action. For a list of supported encoding types, view the Supported Encodings page for the version of Java you are using:	
	Java 11	https://docs.oracle.com/en/java/javase/11/intl/supported-encodings.html
	Java 17	https://docs.oracle.com/en/java/javase/17/intl/supported-encodings.html
	Java 21	https://docs.oracle.com/en/java/javase/21/intl/supported-encodings.html
	If the encoding type you enter in the Destination File Encoding field is invalid or unsupported, then system default encoding will be applied. If you leave the Destination File Encoding field blank, then UTF-8 encoding will be attempted. If the installer fails to apply UTF-8, then system default encoding will be applied.  ***Note*** ▪ *Note the following regarding file encoding:* • *Retention of the existing encoding type is not supported.* • *There will not be any build time validation.* • *The result of applying an encoding type will be written to the log file.*	

Table 9-37 ▪ Modify Text File - Multiple Files Customizer (cont.)

Option	Description
Additional Text	Use this option to instruct the installer to add additional text to text file. Choose one of the following options: • **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. • The backup file name is determined by adding `.backup` to the original file name. • Backup files are not removed by the uninstaller.
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).
Search and replace strings	If you want the installer to search for specific text strings in the text file(s) and replace them with replacement strings, select this option and then click **Configure** to open the **Search and Replace Strings** dialog box. On the **Search and Replace Strings** dialog box, click **Add** to define the **Search For** and **Replace With** texts. When this action runs, the installer locates all instances of the **Search For** text and replaces them with their corresponding **Replace With** text.
Warn if search string is not found	If you want InstallAnywhere to log a warning message in the log file whenever a text string—specified under the **Search For** field on the **Search and Replace Strings** dialog box—is not found in the target text file(s), select this option. The warning message notifies that expected text replacement is not applied to the text file(s) since the specified search string does not exist in the file(s). By default, this option is cleared.

Modify Text File - Single File Action

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 9-38 ▪ Modify Text File - Single 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 Choose a Text File dialog box and select the installed file you want the installer to modify. Available only in the Install and Post-Install sequences.
Existing File	Select this option if you want the installer to modify a text file that already exists on the target system. Enter the path of the text 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 path of the text file in the text box.
Change Line Endings To	Specify the type of line ending to use in the text file: ● User's Platform (Default) ● OS ● UNIX ● DOS/Windows To ensure that the line endings conform to the built-in line endings for the target system, choose User's Platform.
Source File Encoding	Enter the encoding type of the source file that you are modifying. For a list of supported encoding types, view the Supported Encodings page for the version of Java you are using. **Java 11** https://docs.oracle.com/en/java/javase/11/intl/supported-encodings.html **Java 17** https://docs.oracle.com/en/java/javase/17/intl/supported-encodings.html **Java 21** https://docs.oracle.com/en/java/javase/21/intl/supported-encodings.html  ***Note ▪*** *Note the following regarding file encoding:* ● *Retention of the existing encoding type is not supported.* ● *There will not be any build time validation.* ● *The result of applying an encoding type will be written to the log file.*

Table 9-38 ▪ Modify Text File - Single File Customizer (cont.)

Option	Description
Destination File Encoding	Enter the encoding type of the destination file to be saved after it has been modified by the Modify Text File action. For a list of supported encoding types, view the Supported Encodings page for the version of Java you are using: **Java 11** https://docs.oracle.com/en/java/javase/11/intl/supported-encodings.html **Java 17** https://docs.oracle.com/en/java/javase/17/intl/supported-encodings.html **Java 21** https://docs.oracle.com/en/java/javase/21/intl/supported-encodings.html If the encoding type you enter in the Destination File Encoding field is invalid or unsupported, then system default encoding will be applied. If you leave the Destination File Encoding field blank, then UTF-8 encoding will be attempted. If the installer fails to apply UTF-8, then system default encoding will be applied.  ***Note ▪*** *Note the following regarding file encoding:* • *Retention of the existing encoding type is not supported.* • *There will not be any build time validation.* • *The result of applying an encoding type will be written to the log file.*
Additional Text	Use this option to instruct the installer to add additional text to text file. Choose one of the following options: • **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. • The backup file name is determined by adding `.backup` to the original file name. • Backup files are not removed by the uninstaller.

Table 9-38 ▪ Modify Text File - Single File Customizer (cont.)

Option	Description
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).
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** texts. When this action runs, the installer locates all instances of the **Search For** text and replaces them with their corresponding **Replace With** text.
Warn if search string is not found	If you want InstallAnywhere to log a warning message in the log file whenever a text string—specified under the **Search For** field on the **Configure Search and Replace Strings** dialog box—is not found in the target text file, select this option. The warning message notifies that expected text replacement is not applied to the text file since the specified search string does not exist in that file. By default, this option is cleared.

Read/Modify XML File Action

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 9-39 ▪ 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. Available only in the Install and Post-Install sequences.
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.

Table 9-39 ▪ Read/Modify XML File Customizer (cont.)

Option	Description
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: • **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 of the exact occurrence.  ***Note ▪*** *The name of the Tag can be specified along with its occurrence using the format of tagname{occurrence#}, such as book{2}, where occurrence is a zero based sequential index of occurrence number of the tag book.*
Replace value	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: • **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.  ***Note ▪*** *The name of the Tag can be specified along with its occurrence using the format of tagname{occurrence#}, such as book{2}, where occurrence is a zero based sequential index of occurrence number of the tag book.*

Table 9-39 ▪ Read/Modify XML File Customizer (cont.)

Option	Description
Find occurrence of	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: • **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 specified in the **Variable To Store Value** field of the action.  ***Note ▪*** *The Tag specified should be an atomic/leaf node tag in the XML file.*

Rules, Tags, and Rollback Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.
- **Rollback**—Use this tab to specify rollback behavior for the selected action. For more information, see Configuring Installation Rollback Behavior.

Performing Complex XML File Modifications

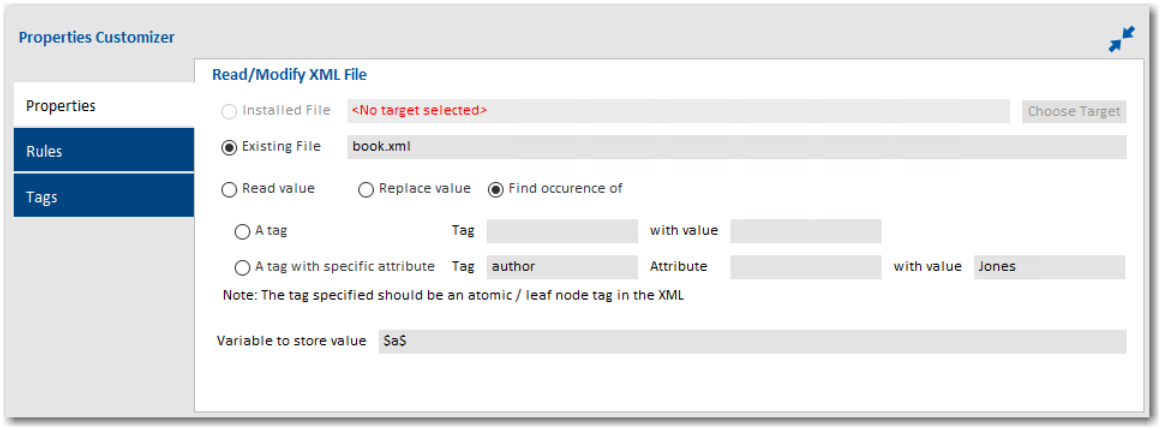
To perform complex modifications, you can use multiple Read/Modify XML File actions. For example, suppose you had the following XML file:

```
<?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 9-1: 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`.



- 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 Action

Te Show Message Dialog action creates a modal dialog that requests end-user input. The message dialog 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 enter information. When controlled by rules, the message dialog can also serve as a data verification tool.

The following options are included:

Table 9-40 ▪ 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.
Cancel and Exit on ESC or X	If you want the installer to cancel and exit the installation if the end user presses the Escape button or clicks the Close button (X) on the message dialog, select this check box. This check box is cleared by default.
Button 0, 1, 2	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 9-40 ▪ Show Message Dialog Options (cont.)

Option	Description
Button 2	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... 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. Also, specify the action to take when the button is clicked: Return to Previous Panel, Continue to Next, or Cancel and Exit.  ***Note*** ▪ *If you add a Show Message Dialog to the Pre-Install, Post-Install, Pre-Uninstall, or Post-Uninstall phase, the Return to Previous Panel option is enabled. If a Show Message Dialog is added to the Install or Uninstall sequence, Return to Previous Panel will be disabled.*
Results Variable	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: ● **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`.

Associate File Extension/URL Prefix - Windows Action

You can use the Associate File Extension/URL Prefix - Windows Action to associate extensions to installed & existing applications on Windows.

Properties Tab

The Properties tab of the Associate File Extension/URL Prefix - Windows Action customizer includes the following properties:

Table 9-41 ▪ Associate File Extension/URL Prefix - Windows Customizer | Properties Tab

Property	Description
Installed File	Select this option if you want the installer to associate an Application, the installer deploys on the target system. Click the **Choose Application** button to open the **Choose a File Dialog** box, and then select the installed file you want the installer to associate. The installed `.exe` File or Launcher must be selected.
Existing File	Select this option if you want the installer to associate an application that already exists on the target system. Enter the path of the `.exe` file in the text box.
Extension	Enter the extension you want to associate without a Dot. Extension can be any alpha-numeric value supported by Windows. New extensions or existing extensions can be given. This property is Mandatory if "Extension" option is selected.
URL Prefix	Enter the URL Prefix you want to associate with an Installed or Existing Application. Prefix can be any alpha-numeric value supported by Windows. New Prefix can be associated to a Protocol handler Application. This property is Mandatory if "URL Prefix" option is selected.
Icon	Select this option if you want the installer to associate an icon to the application the installer deploys on the target system. Click the **Choose Icon** button to open the **Choose a File Dialog** box and select the installed Icon you want for the associated application. .ICO Icon file with at least 128 x 128 size to be selected. Ideal to select 256 X 256 pixels size transparent icons for better display in Windows Large/Extra Large Icon View. This property is optional. If Icon is not selected, Default Icon of the Application will be retained.
MIME Type	Enter an **Existing Mime Type** for the **File** type. This value is optional.
Friendly Name	Enter a **Friendly** name for the **Type**. This value is Optional. The default type name will be retained if Friendly name is not specified.

Property	Description
ProgID	Enter a Prog ID that must be registered for this File extension. The file type's Prog ID is an arbitrary string. But it must be unique on the target system. One **ProgID** naming convention is to append the word file to the extension. For example, `.ext` extension can use the `ProgID  extFile`. Another naming convention is to name **ProgID** after the application used to open the file type. For Example, `SampleApp.Document`. But it must be unique on the target System. This value is optional. If no ProgID is given a unique **ProgID** will be generated and used.

The **Verbs** button can be selected to Configure Verbs for the Extension.

Verbs Configuration Dialog

This Dialog is used to Configure Verbs for the extension.

Table 9-42 ▪ Verbs Configuration Dialog

Value	Description
Command Sequence	Enter a sequence number for the Verb. If the file extension has more than one Verb associated with it, the sequence number determine the order in which the Verbs are displayed on the right-click menu.
Verb	The Verb that appears in the right-click menu.
Display Name	Enter the text you want to display for the Verb. For Example, "Open with NotePad++". This value is optional. If **Display Name** is not specified, the name of the verb will be displayed in the right-click menu.
Argument	Enter the command line arguments for the Verb. Use `%1` in the argument in place of the selected filename. For Example, when an end user right click `C:\file.exe` and `-p %1` is the argument for the verb, the command line argument becomes `-p c:\file.ext`. Enter the arguments without any quotes.

Panel Actions

The following table lists the panel actions available in InstallAnywhere.

Table 9-43 ▪ Panel Actions

Action	Description
Choose Alias, Link, Shortcut Panel Action	The Choose Alias, Link, Shortcut panel action enables end users to choose an installation location for shortcuts (Windows), aliases (OS or OS X), and links (UNIX). This panel action is included in new projects by default.
Choose Database Connection Panel Action	The Choose Database Connection panel action enables end users to specify information for connecting to a database.
Choose Features to Uninstall Panel Action	This panel enables end users to select which features they want to uninstall.

Table 9-43 ▪ Panel Actions (cont.)

Action	Description

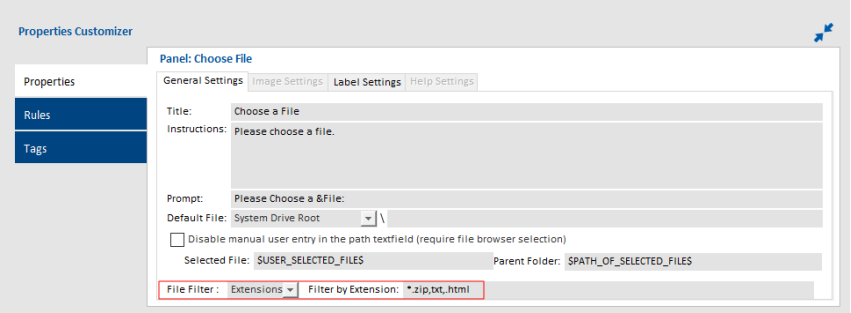
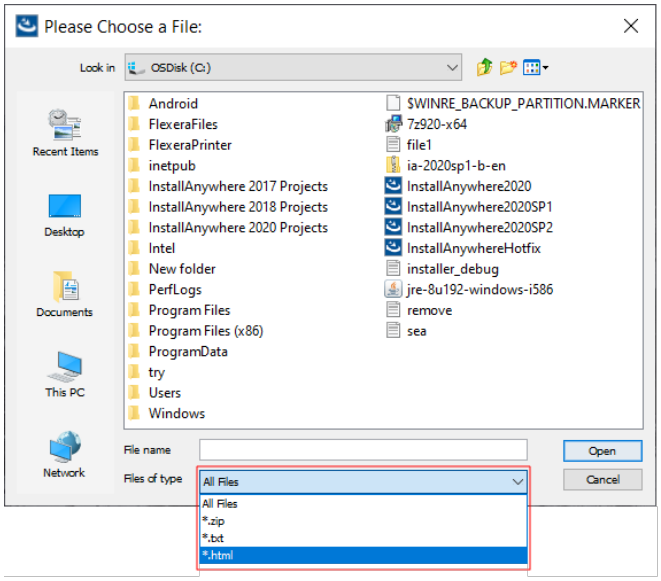
Choose File Panel Action	The Choose File panel action lets end users select a file; it uses the file name and the path to the file as the values of InstallAnywhere standard variables. You can use these variables elsewhere in your installer.
	
	Additionally, you can configure the Choose File panel to filter the files by specific type. The configurable filter options are:
	
	● **File Filter** - Select one of the following values from the drop-down option:
	  ● **All Files** - It accepts all files in **Choose File** panel and a specific file extension cannot be filtered. By default, this value is selected.
	  ● **Extensions** - It enables you to configure or customize your file extensions for **Choose File** panel.
	
	
	
	● **Filter by Extension** - Provide extensions that you can configure as file filters for **Choose File** panel. Multiple filters can be provided using comma separator. This option is enabled only when **Extensions** is selected under **File Filter.** Examples for file extensions: *.zip, .txt, .html, etc.
	
	Extensions value will be listed under **File of Type** along with **All Files** at runtime. The extension selected will filter files with same extension in the provided path.
	
	

Table 9-43 ▪ Panel Actions (cont.)

Action	Description
Choose Folder Panel Action	The Choose Folder panel action lets end users select a folder; it uses the folder as the value of an InstallAnywhere standard variable. You can use these variables elsewhere in your installer.
Choose Install Folder Panel Action	The Choose Install Folder panel action lets end users choose the primary installation folder. If you omit this panel, the Default Install Folder settings in the Platforms view on the Project page apply. This panel action is included in new projects by default.
Choose Install Sets Panel Action	The Choose Install Sets panel action enables end users to specify which install set or features they want to install.
Choose Java VM Panel Action	The Choose Java VM panel action lets end users select which Java VM to use for any installed LaunchAnywhere launchers.
Choose Uninstall Type Panel Action	The Choose Uninstall Type panel action enables end users to select whether to uninstall all or part of the application.
Custom Code Panel Action	InstallAnywhere's custom code API enables you to create custom panels where necessary.  ***Note ▪*** *For information on the support of signed JARs as dependencies, see Support for Signed JARs as Dependencies.*
Disk Space Check Panel Action	The Disk Space Check panel 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, the installer prompts the end user to free the required disk space or choose another install location.  ***Note ▪*** *This action is automatically added before files are installed. The action does not appear in the list.*
Display HTML Panel Action	The Display HTML panel action displays HTML from an archived file or a specific URL on a panel during the installation.
Display Message Panel Action	The Display Message panel action lets you display a text message to end users 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 issues with InstallAnywhere variables.

Table 9-43 ▪ Panel Actions (cont.)

Action	Description
Find File/Folder Panel Action	The Find File/Folder panel action implements a search process that searches portions of the file system for a specific file or for a file that matches a certain pattern. The end user can also choose a matching file.
Get Password Panel Action	The Get Password panel action lets end users enter a password. You can choose to validate the password against a list of specified passwords (enabling the index feature that allows different passwords to effectively unlock different features), or you can store the entered password in a variable (as when requesting a password to be used in a configuration routine). The installer automatically encrypts password values according to the security settings that are specified in the Variables view on the Project page.
Get Serial Number Panel Action	Implementing InstallAnywhere's built-in serial number verification and creation routines, the Get Serial Number panel action enables you to add serial number functionality to the installer. You 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 action enables you to create rules that can manage all aspects of the installation based on rights that are granted by the serial numbers that end users enter.
Get User Input - Advanced Panel Action	The Get User Input - Advanced panel action enables you to get input from end users 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. For more information, see Get User Input Panels.
Get User Input - Simple Panel Action	The Get User Input - Simple panel action enables action enables you to get input from end users. For more information, see Get User Input Panels.
Important Note Panel Action	The Important Note panel action enables you to display a text or HTML file without the use of the radio buttons that are on the license agreement panel. It is particularly useful for displaying Readme or errata type documents.
Install Complete Panel Action	The Install Complete panel action displays information about the installation's status to end users. It also optionally is displayed if a restart is needed on Windows-based system. This action is available only after files have been installed.
Introduction Panel Action	The Introduction panel action offers an introduction to the installation. This panel action is included in new projects by default.

Table 9-43 ▪ Panel Actions (cont.)

Action	Description
License Agreement Panel Action	The License Agreement panel action displays license information and prompts end users to agree or disagree with the terms of the license. With this action, you need to specify a text file (either HTML or plain text) that contains the license information that you want to be displayed on the run-time panel.
Minimal UI Panel Action	The Minimal UL panel action lets you display the name of the product, the process of installation and the license agreement.
Pre-Install Summary Panel Action	The Pre-Install Summary panel action summarizes information that was collected before installing files. This panel action is included in new projects by default.
Scrolling Message Panel Action	The Scrolling Message panel action lets you display a lot of text in a message panel that includes scroll bars. This may be particularly useful for instructions.
Tomcat Runtime Deployment Panel Action	With the Tomcat Deployment Option panel action, you can offer end users one or more options for deploying to Tomcat servers, and also enable them to specify server connection information.
Uninstall Complete Panel Action	The Uninstall Complete panel action displays information about the installation's status to end users. It also optionally is displayed if a restart is needed on Windows-based system.
Uninstaller Introduction Panel Action	The Uninstaller Introduction panel action offers an introduction to the uninstaller.
WebSphere Runtime Deployment Panel Action	With the WebSphere Deployment Option panel action, you can offer end users options for deploying to WebSphere servers, as well as enable them to specify server connection information.

Choose Database Connection Panel Action

The Choose Database Connection panel action enables end users to specify information for connecting to a MySQL, Microsoft SQL Server, or PostgreSQL database. The variables that are used to store the values that end users specify can be used with the Run SQL Script action to run a SQL script.



Important ▪ *SQL databases limit remote administration by users unless otherwise configured. Therefore, for connections to remote SQL database servers, the user name and the password that the end user specifies must be for a user who is running the installer from an appropriate IP address and has been granted adequate privileges that are required to run your SQL script.*

The Choose Database Connection customizer includes the following tabs:

- General Settings
- Image Settings, Label Settings, and Help Settings

General Settings

The General Settings tab of the Choose Database Connection customizer includes the following settings:

Table 9-44 ▪ General Settings Tab on the Choose Database Connection Customizer

Setting	Description
Title	Enter the text that you want to use in the title bar of the panel. The default value is: **Choose Database Connection**
Instructions	Enter the text that you want to display on the Choose Database Connection panel. The default value is: **Provide database configuration details and authentication**
Database Name	Specify the name of the database on which the installer is running the SQL script. By default, the entry in this setting is the user-defined variable $DB_NAME_VARIABLE$. You can specify a different variable if needed.
Server Host	Specify the server host to which your installer is connecting. By default, the entry in this setting is the user-defined variable $DB_SERVERHOST_VARIABLE$. You can specify a different variable if needed.
Server Port	Specify the port that should be used for the server connection. By default, the entry in this setting is the user-defined variable $DB_SERVERPORT_VARIABLE$. You can specify a different variable if needed.
User Name	Specify the user name for the account that should be used to connect to the server through server authentication. By default, the entry in this setting is the user-defined variable $DB_USERNAME_VARIABLE$. You can specify a different variable if needed.
Password	Specify the password that corresponds with the specified user name for the account that should be used to connect to the server through server authentication. By default, the entry in this setting is the user-defined variable $DB_PASSWORD_VARIABLE$. You can specify a different variable if needed.

Table 9-44 ▪ General Settings Tab on the Choose Database Connection Customizer (cont.)

Setting	Description
Include Test Connection button on run-time panel	If you want the Choose Database Connection panel to include a Test Connection button that lets end users test the connection information that they entered, select this check box.
Enable test connection when the Next button is clicked	If you want the clicking of the Next button on the Choose Database Connection panel to test the connection information that end users entered, select this check box.
Test Connection Type	Select the type of database to which your installer is connecting.

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 Install Sets Panel Action

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.



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 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 9-45 ▪ 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: • **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 Panel Action

You can use the Choose Java VM panel action to enable end users to select which Java VM to use for any installed LaunchAnywhere launchers. You can specify whether the installer offers to install a VM that is bundled with the installer. You can also permit end users to provide additional search paths or to choose a specific VM that is 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 9-46 ▪ 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 check boxes 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 in the Windows and UNIX areas in the Platforms view on the Project page.
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.

The behavior of this panel can be affected by the VM Search Settings area on the General subtab (Project page > JVM Settings view > Search Panel Settings tab), and by the search paths that are specified on the Windows and UNIX subtabs (Project page > JVM Settings view > Search Panel Settings tab).



Note ▪ *Consider the following information:*

- *The Choose Java VM panel is not shown in OS or OS X–based installers. To force a OS or OS X–based installer to use a different VM, use the Select Java VM for LaunchAnywhere setting in the OS X area in the Platforms view on the Project page.*
- *On the Windows platform, when the installer or uninstaller is launched with administrative privileges, JVMs located in unsecured folders are considered invalid and are not used. This behavior can be turned off by using one of the following methods:*
  - *Configure the `designer.jvm.securefolder.check` property to `false` in the `com.zerog.ia.Designer.properties` file.*

- *Configure the **Secure JVM Folder Check** setting to **Disable** in the **Project** > **Platforms** >**Windows** area.*
- *Run the installer or uninstaller using the `<installer/uninstaller_name> -Djvm.securefolder.check=<false>` command line, where `<installer/uninstaller_name>` refers to your installer or uninstaller executable.*

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.

Custom Code Panel Action

You can use the Custom Code panel action to create custom panels when necessary, using InstallAnywhere's custom code API. The Custom Code customizer includes controls on the following tabs:

- General Settings
- Image Settings, Label Settings, and Help Settings

General Settings

The General Settings tab of the Custom Code customizer includes the following controls:

Table 9-47 ▪ General Settings Tab of Custom Code Customizer

Control	Description
Path	Identifies the archive that contains all classes needed to implement the Custom Code panel. • **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`.
Dependencies	Custom code tasks sometimes are dependent on 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.*
Scroll Panel Option	Use the following options to enable scrolling on the Custom Code panel: • **Enable Vertical Scroll** • **Enable Horizontal Scroll**

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 Panel Action

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 9-48 ▪ 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:
	• **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:
	`http://www.mywebsite.com/HTMLFile.html`
	If you want to display a HTML page located on the target system, precede the location with `file:////`. For example:
	`file:////C:\MyHTMLFiles\HTMLFile.html`
	
	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:////`*.*

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 *file*, simply enclose the variable name in dollar signs: *$file$*.



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.

License Agreement Panel Action

The License Agreement panel action lets you display license information and prompts end users to agree or disagree with the terms of the license.

With this action, you need to specify a text file (either HTML or plain text) that contains the license information that you want to be displayed on the run-time panel.

The Properties Customizer consists of the following tabs:

- General Settings
- Label Settings

General Settings

The **General Settings** tab of the Properties Customizer Includes the following controls:

Table 9-49 ▪ Properties Customizer / General Settings Tab

Control	Description
Title	Enter a title for the License Agreement panel.
Prompt	Enter the instructions that you want to display to the end user on the License Agreement panel of the installer. By default, the prompt text is **`Installation and Use of $PRODUCT_NAME$ Requires Acceptance of the Following License Agreement`**.

Table 9-49 ▪ Properties Customizer / General Settings Tab (cont.)

Control	Description
License Agreement Text Source File	Specify the path of the license agreement source file from which the license agreement text you want to be displayed on the License Agreement panel during installation. Select one of the following options to specify the file path: • **Path**—Select this option and click **Choose File** to locate a license agreement source file. • **Existing File**—Select this option to reference a license agreement source file by specifying its location when the file already exists within the InstallAnywhere project. Use this option to reuse the existing file for this panel instead of selecting a new file from the system.  ***Note*** ▪ *A license agreement source file must be in either plain text (.txt) or HTML (.html) format.*
Open external links in the user's default browser (HTML only)	Specify how you want hyperlinks in an HTML-based license agreement on the License Agreement panel are handled during installation. If the **Open external links in the user's default browser (HTML only)** checkbox is selected, hyperlinks will open in the user's default web browser, and If this checkbox is cleared, hyperlinks will open within the installer's embedded HTML viewer, where supported.
Default setting for acceptance of the license agreement	Specify whether you want the **I accept the terms of the License Agreement** checkbox is selected by default on the License Agreement panel. This setting includes the following options (radio buttons): • **Agree**— Select this option to have the **I accept the terms of the License Agreement** checkbox selected by default on the License Agreement panel. • **Disagree**—Select this option to have the **I accept the terms of the License Agreement** checkbox cleared by default on the License Agreement panel, requiring the end user to explicitly select it before proceeding.
Force user to scroll through license agreement	Specify whether you want that the end user must scroll to the end of the license agreement text on the License Agreement panel before the **Next** button is enabled. Selecting this checkbox ensures that the end user view the entire license agreement before continuing with the installation.
Include 'Do Not Accept'	Specify whether you want to display the **I do NOT accept the terms of the License Agreement** radio button on the License Agreement panel during installation. Selecting the **Include 'Do Not Accept'** check box display this radio button. If a user selects the **I do NOT accept the terms of the License Agreement** radio button on the License Agreement panel, the installation does not proceed and remains blocked unless the user selects the **I accept the terms of the License Agreement** radio button.

Table 9-49 ▪ Properties Customizer / General Settings Tab (cont.)

Control	Description
Preview	Click the Preview button to see a sample run-time panel.

Label Settings

For information on using the options on the Label Settings, see Panel Action Settings.

Minimal UI Panel Action

The Minimal UL panel action lets you display the name of the product, the process of installation and the license agreement. The Properties Customizer consists of the following tabs:

- General Settings
- Label Settings

General Settings

On the General Settings tab of the Properties Customizer, you assign the Minimal UI panel a title and select the source of the License Agreement Text File.

Table 9-50 ▪ Properties Customizer / General Settings Tab

Control	Description
Title	Enter a title for the Minimal UI panel.
Show License Agreement Checkbox	Select this checkbox to agree to the license agreement.
License Agreement Text Source File	Specify the source of the License Agreement Text Source file by selecting one of the following options: • **File 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. • **Web URL**—Enter the URL for the page you want the Minimal UI panel to render. If you are entering an HTTP location, use the following syntax: `http://www.mywebsite.com/HTMLFile.html`
Preview	Click the Preview button to see a sample run-time panel.

Label Settings

For information on using the options on the Label Settings, see Panel Action Settings.

Pre-Install Summary Panel Action

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 9-51 ▪ 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: `Please Review the Following Before Continuing:`
Include the following information in the Pre-Installation Summary panel	Select the information that you want to display on this Pre-Install Summary panel. Options include the following: • Product name • Install folder • Alias, Link, Shortcut folder • Selected install sets • Selected product features • VM install folder/System VM • Disk space information 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.  ***Note*** ▪ *For information on checking disk space, see Checking Disk Space During Installation.*
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.

Table 9-51 ▪ Pre-Install Summary Customizer / General Settings Tab (cont.)

Control	Description
Custom Fields List	List of the custom variable fields that you have added to display on this panel. Information displayed includes: • **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.

Tomcat Runtime Deployment Panel Action

With the Tomcat Runtime Deployment panel action, you can offer end users one or more options for deploying to Tomcat servers, and also enable them to specify server connection information. The variables that are used to store the values that end users specify can be used with the Deploy WAR/EAR Archive action to deploy Web applications to the Tomcat server.

The Tomcat Runtime Deployment customizer includes the following tabs:

- General Settings
- Image Settings, Label Settings, and Help Settings

General Settings

The General Settings tab of the Tomcat Runtime Deployment customizer includes the following settings:

Table 9-52 ▪ General Settings Tab on the Tomcat Runtime Deployment Customizer

Control	Description
Title	Enter the text that you want to use in the title bar of the panel. The default value is: **Choose Tomcat Deployment Option**
Instructions	Enter the text that you want to display on the Choose Tomcat Deployment Option panel. The default value is: **Where would you like to deploy the WAR?**
Associated Tomcat Server	Select the name of the Tomcat server that you want to be associated with the Choose Tomcat Deployment Option panel.

Table 9-52 ▪ General Settings Tab on the Tomcat Runtime Deployment Customizer (cont.)

Control	Description
Deployment Option Variable	Specify the variable that indicates the deployment option that is used for the WAR file. By default, the entry in this setting is the user-defined variable $TOMCAT_DEPLOYMENT_OPTION$. You can specify a different variable if needed. The variable can be set to one of the following values: ● 1—Local Tomcat ● 2—Remote Tomcat ● 3—Save the WAR locally on the target system for later deployment
Server Path Variable	Specify the variable that indicates the path on the server where the WAR file should be deployed. By default, the entry in this setting is the user-defined variable $TOMCAT_SERVER_PATH$. You can specify a different variable if needed. This setting applies to local Tomcat servers; this setting is available if the Local Tomcat check box is selected for the associated Tomcat server host.
Host Name Variable	Specify the variable that indicates the name of the Tomcat server or its IP address. By default, the entry in this setting is the user-defined variable $TOMCAT_HOSTNAME$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.
Port Variable	Specify variable that indicates the port number on which Tomcat is running. By default, the entry in this setting is the user-defined variable $TOMCAT_PORT$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.
User Name Variable	Specify the variable that indicates the user name for the account that is specified in the `tomcat-users.xml` file (which is in the conf folder in the Tomcat directory). By default, the entry in this setting is the user-defined variable $TOMCAT_USERNAME$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.

Table 9-52 ▪ General Settings Tab on the Tomcat Runtime Deployment Customizer (cont.)

Control	Description
Password Variable	Specify the variable that indicates the password that corresponds with the specified user name for an account that is specified in the `tomcat-users.xml` file (which is in the conf folder in the Tomcat directory). By default, the entry in this setting is the user-defined variable $TOMCAT_PASSWORD$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.
SSL Enabled	Indicate whether SSL is enabled on the remote Tomcat server. By default, the entry in this setting is the user-defined variable $TOMCAT_SSL_CONNECTION_STATUS$. You can specify a different variable if needed. If this variable is set to Yes at run time, the action uses SSL for deployment to the server. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.

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.

WebSphere Runtime Deployment Panel Action

With the WebSphere Runtime Deployment panel action, you can offer end users options for deploying to WebSphere servers, as well as enable them to specify server connection information. The variables that are used to store the values that end users specify can be used with the Deploy WAR/EAR Archive action to deploy Web applications to the WebSphere server.

The WebSphere Runtime Deployment customizer includes the following tabs:

- General Settings
- Image Settings, Label Settings, and Help Settings

General Settings

The General Settings tab of the WebSphere Runtime Deployment customizer includes the following settings:

Table 9-53 ▪ General Settings Tab on the WebSphere Runtime Deployment Customizer

Control	Description
Title	Enter the text that you want to use in the title bar of the panel. The default value is: **Choose WebSphere Deployment Option**
Connection Name	Select the name of the WebSphere server that you want to be associated with the Choose WebSphere Deployment Optional panel.
Preview	Click the Preview button to see a sample run-time panel.
Deployment Option Variable	Specify the variable that indicates the deployment option that is used for the WAR file. By default, the entry in this setting is the user-defined variable $WEBSPHERE_DEPLOYMENT_OPTION$. You can specify a different variable if needed. ● 1—Local or remote WebSphere server ● 2—Save the EAR or WAR locally on the target system for later deployment
Host Name Variable	Specify the variable that indicates the name of the WebSphere server or its IP address. By default, the entry in this setting is the user-defined variable $WEBSPHERE_HOSTNAME$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
SOAP Port Variable	Specify variable that indicates the SOAP port number on which WebSphere is running. By default, the entry in this setting is the user-defined variable $WEBSPHERE_PORT$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.

Table 9-53 ▪ General Settings Tab on the WebSphere Runtime Deployment Customizer (cont.)

Control	Description
Admin Security Enabled	Specify variable that indicates whether administrative security is enabled on the WebSphere application server. If this variable is set to Yes at run time, the action uses the credentials in the User Name and Password settings to connect to the server. By default, the entry in this setting is the user-defined variable $WEBSPHERE_IS_SECURITY_ENABLED$. You can specify a different variable if needed. The available values are Yes and No. This setting applies to remote WebSphere servers.
User Name Variable	Specify the variable that indicates the user name for the account that should be used to connect to the WebSphere server if administrative security is enabled. By default, the entry in this setting is the user-defined variable $WEBSPHERE_USERNAME$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
Password Variable	Specify the variable that indicates the password that corresponds with the specified user name for an account that should be used to connect to the WebSphere server. By default, the entry in this setting is the user-defined variable $WEBSPHERE_PASSWORD$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
SSL TrustStore Path	Specify the variable that indicates the path to the SSL TrustStore file. By default, the entry in this setting is the user-defined variable $WEBSPHERE_TRUSTSTORE_PATH$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
SSL TrustStore Password	Specify the variable that indicates the password to the SSL TrustStore. By default, the entry in this setting is the user-defined variable $WEBSPHERE_TRUSTSTORE_PASSWORD$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.

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 request end-user input when the end user launches an installer through a command-line interface.

Table 9-54 ▪ Console Actions

Action	Description
Choose Database Connection Console Action	The Choose Database Connection console action enables end users to specify information for connecting to a database.
Choose Features to Uninstall Console Action	The Choose Features to Uninstall console action enables end users to select which features they want to uninstall.
Choose Install Folder Console Action	The Choose Install Folder console action lets end users choose the primary installation folder. If you omit this console, the Default Install Folder settings in the Platforms view on the Project page apply.
Choose Install Sets Console Action	The Choose Install Sets console action enables end users to specify which install set or features they want to install.
Choose Java VM Console Action	The Choose Java VM console action lets end users select which Java VM to use for any installed LaunchAnywhere launchers.
Choose Link Folder Console Action	The Choose Link Folder console action enables end users to choose an installation location for links on UNIX-based systems.
Choose Uninstall Type Console Action	The Choose Uninstall Type console action enables end users to select whether to uninstall all or part of the application.
Custom Code Console Action	InstallAnywhere's custom code API enables you to create custom consoles where necessary.  ***Note ▪*** *For information on the support of signed JARs as dependencies, see Support for Signed JARs as Dependencies.*
Display Message Console Action	The Display Message console action lets you display a text message to end users during the installation. This can be useful for conveying information about installation choices that the end user has made. This console is also particularly useful in debugging installer issues with InstallAnywhere variables.

Table 9-54 ▪ Console Actions (cont.)

Action	Description
Get Password Console Action	The Get Password console action allows end users to enter a password. You can choose to validate the password against a list of specified passwords (enabling the index feature that allows different passwords to effectively unlock different features), or you can store the entered password in a variable (as when requesting a password to be used in a configuration routine). The installer automatically encrypts password values according to the security settings that are specified in the Variables view on the Project page.
Get Serial Number Console Action	Implementing InstallAnywhere's built-in serial number verification and creation routines, the Get Serial Number console action enables you to add serial number functionality to the installer. You 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 action enables you to create rules that can manage all aspects of the installation based on rights that are granted by the serial numbers that end users enter.
Get User Input Console Action	The Get User Input console action enables you to get input from end users. For more information, see Get User Input Panels.
Install Complete Console Action	The Install Complete console action displays information about the installation's status to end users. It also optionally is displayed if a restart is needed on Windows-based system. This action is available only after files have been installed.
Install Failed Console Action	The Install Failed console action should be displayed when a console installer has generated an error.
License Agreement Console Action	The License Agreement console action displays license information and prompts end users to agree or disagree with the terms of the license. With this action, you need to specify a text file (either HTML or plain text) that contains the license information that you want to be displayed on the console.
Pre-Install Summary Console Action	The Pre-Install Summary console action summarizes information that was collected before installing files.
Ready to Install Console Action	The Ready to Install console action alerts end users that the installer is about to install files.
Show Message Console 'Dialog' Console Action	The Show Message Console 'Dialog' console action displays a message to end users.
Tomcat Runtime Deployment Console Action	With the Tomcat Deployment Option console action, you can offer end users one or more options for deploying to Tomcat servers, and also enable them to specify server connection information.

Table 9-54 ▪ Console Actions (cont.)

Action	Description
Uninstall Complete Console Action	The Uninstall Complete console action displays information about the uninstallation's status to end users. It also optionally is displayed if a restart is needed on Windows-based system.
Uninstaller Introduction Console Action	The Uninstaller Introduction console action offers an introduction to the uninstaller.
WebSphere Runtime Deployment Console Action	With the WebSphere Deployment Option console action, you can offer end users options for deploying to WebSphere servers, as well as enable them to specify server connection information.

Choose Database Connection Console Action

The Choose Database Connection console action enables end users to specify information for connecting to a MySQL, Microsoft SQL Server, IBM DB2, or PostgreSQL database. The variables that are used to store the values that end users specify can be used with the Run SQL Script action to run a SQL script.



Important ▪ *SQL databases limit remote administration by users unless otherwise configured. Therefore, for connections to remote SQL database servers, the user name and the password that the end user specifies must be for a user who is running the installer from an appropriate IP address and has been granted adequate privileges that are required to run your SQL script.*

The Choose Database Connection customizer includes the following settings:

Table 9-55 ▪ Choose Database Connection Customizer Settings

Setting	Description
Title	Enter the text that you want to use in the title bar of the console. The default value is: **Choose Database Connection**
Instructions	Enter the text that you want to display on the Choose Database Connection console. The default value is: **Provide database configuration details and authentication**
Database Name	Specify the name of the database on which the installer is running the SQL script. By default, the entry in this setting is the user-defined variable $DB_NAME_VARIABLE$. You can specify a different variable if needed.
Server Host	Specify the server host to which your installer is connecting. By default, the entry in this setting is the user-defined variable $DB_SERVERHOST_VARIABLE$. You can specify a different variable if needed.
Server Port	Specify the port that should be used for the server connection. By default, the entry in this setting is the user-defined variable $DB_SERVERPORT_VARIABLE$. You can specify a different variable if needed.
User Name	Specify the user name for the account that should be used to connect to the server through server authentication. By default, the entry in this setting is the user-defined variable $DB_USERNAME_VARIABLE$. You can specify a different variable if needed.
Password	Specify the password that corresponds with the specified user name for the account that should be used to connect to the server through server authentication. By default, the entry in this setting is the user-defined variable $DB_PASSWORD_VARIABLE$. You can specify a different variable if needed.
Enable test connection when moving to next action	If you want to test the connection information that end users entered before moving from the Choose Database Connection console action to the next action, select this check box.
Test Connection Type	Select the type of database to which your installer is connecting.

Get Password Console Action

The Get Password console action enables end users to enter a required password in order to proceed an installer installation. This action is typically used to restrict access to certain installation paths, features, or modes. The installer will prompt the user for a password, and based on the password entered, it can either proceed the installation process or prevent the same.

Properties Tab

The **Properties** tab of the Get Password action customizer includes the following settings:

Table 9-56 ▪ Get Password Customizer Settings

Setting	Description
Title	Enter the text that you want to use in the title bar of the Get Password console. The default value is: **Enter Password**
Instructions	Enter the text that you want to display on the Get Password console. The default value is: **This installation requires a password to continue.**
Prompt	Enter the label text that you want to display just before the password input field on the Get Password console. This label text on the console specifies the information the installer requests. The default value is: **Please Enter the Password**
Results	Specify the required user-defined variables in which the user-entered password and its related index value are stored, by utilizing the following options: ● **Password Entered**—Specify the user-defined variable that stores the password entered by the user on the console. ● **Index of Matching Password**—Specify the user-defined variable that stores the position (index) of the user-entered password within a list of valid passwords, defined in an external password validation file. When an installer runs and the user enters a password, InstallAnywhere checks that password against each line in the password validation file. If the entered password matches one of the entries, the index (starting from 0 for the first line) is stored in the user-defined variable that is defined here. This password validation process enables the index feature, allowing each different password to effectively unlock different features. Therefore, provides a flexible and secure way to manage multiple access keys or license levels.

Table 9-56 ▪ Get Password Customizer Settings

Setting	Description
Validate with File	Select this check box and click **Choose File** to locate a password validation file that contains a list of valid passwords. The file should be a plain text file with one password per line.
Allow password to be left blank	Select this check box if you want to allow the user to skip entering a password in the Get Password console and proceed with the installation. By default, this check box is selected.
Re-Enter Password to Confirm	Select this checkbox if you want the user to be prompted to confirm the password by re-entering the same password. The installer compares both entries to ensure they match before proceeding. By default, this check box is cleared.

Rules and Tags Tabs

For information on the other tabs on this customizer, see the following:

- **Rules**—Use this tab to add rules to the selected action. For more information, see Assigning a Rule to an Action and Rules Reference.
- **Tags**—Use this tab to add build configuration tags to the selected action. For more information, see Assigning Tags to Project Elements.

Tomcat Runtime Deployment Console Action

With the Tomcat Runtime Deployment console action, you can offer end users one or more options for deploying to Tomcat servers, and also enable them to specify server connection information. The variables that are used to store the values that end users specify can be used with the Deploy WAR/EAR Archive action to deploy Web applications to the Tomcat server.

The Tomcat Runtime Deployment customizer includes the following settings:

Table 9-57 ▪ Tomcat Runtime Deployment Customizer Settings

Setting	Description
Title	Enter the text that you want to use in the title bar of the console. The default value is: **Choose Tomcat Deployment Option**
Instructions	Enter the text that you want to display on the Choose Tomcat Deployment Option console. The default value is: **Where would you like to deploy the WAR?**
Associated Tomcat Server	Select the name of the Tomcat server that you want to be associated with the Choose Tomcat Deployment Option console.

Table 9-57 ▪ Tomcat Runtime Deployment Customizer Settings (cont.)

Setting	Description
Deployment Option Variable	Specify the variable that indicates the deployment option that is used for the WAR file. By default, the entry in this setting is the user-defined variable $TOMCAT_DEPLOYMENT_OPTION$. You can specify a different variable if needed. The variable can be set to one of the following values: ● 1—Local Tomcat ● 2—Remote Tomcat ● 3—Save the WAR locally on the target system for later deployment
Server Path Variable	Specify the variable that indicates the path on the server where the WAR file should be deployed. By default, the entry in this setting is the user-defined variable $TOMCAT_SERVER_PATH$. You can specify a different variable if needed. This setting applies to local Tomcat servers; this setting is available if the Local Tomcat check box is selected for the associated Tomcat server host.
Host Name Variable	Specify the variable that indicates the name of the Tomcat server or its IP address. By default, the entry in this setting is the user-defined variable $TOMCAT_HOSTNAME$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.
Port Variable	Specify variable that indicates the port number on which Tomcat is running. By default, the entry in this setting is the user-defined variable $TOMCAT_PORT$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.
User Name Variable	Specify the variable that indicates the user name for the account that is specified in the `tomcat-users.xml` file (which is in the conf folder in the Tomcat directory). By default, the entry in this setting is the user-defined variable $TOMCAT_USERNAME$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.

Table 9-57 ▪ Tomcat Runtime Deployment Customizer Settings (cont.)

Setting	Description
Password Variable	Specify the variable that indicates the password that corresponds with the specified user name for an account that is specified in the `tomcat-users.xml` file (which is in the conf folder in the Tomcat directory). By default, the entry in this setting is the user-defined variable $TOMCAT_PASSWORD$. You can specify a different variable if needed. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.
SSL Enabled	Indicate whether SSL is enabled on the remote Tomcat server. By default, the entry in this setting is the user-defined variable $TOMCAT_SSL_CONNECTION_STATUS$. You can specify a different variable if needed. If this variable is set to Yes at run time, the action uses SSL for deployment to the server. This setting applies to remote Tomcat servers; this setting is available if the Remote Tomcat check box is selected for the associated Tomcat server host.

WebSphere Runtime Deployment Console Action

With the WebSphere Runtime Deployment console action, you can offer end users options for deploying to WebSphere servers, as well as enable them to specify server connection information. The variables that are used to store the values that end users specify can be used with the Deploy WAR/EAR Archive action to deploy Web applications to the WebSphere server.

The WebSphere Runtime Deployment customizer includes the following settings:

Table 9-58 ▪ WebSphere Runtime Deployment Customizer

Control	Description
Title	Enter the text that you want to use in the title bar of the console. The default value is: **Choose WebSphere Deployment Option**
Connection Name	Select the name of the WebSphere server that you want to be associated with the Choose WebSphere Deployment Optional console.
Preview	Click the Preview button to see a sample run-time panel.

Table 9-58 ▪ WebSphere Runtime Deployment Customizer (cont.)

Control	Description
Deployment Option Variable	Specify the variable that indicates the deployment option that is used for the WAR file. By default, the entry in this setting is the user-defined variable $WEBSPHERE_DEPLOYMENT_OPTION$. You can specify a different variable if needed. ● 1—Local or remote WebSphere server ● 2—Save the EAR or WAR locally on the target system for later deployment
Host Name Variable	Specify the variable that indicates the name of the WebSphere server or its IP address. By default, the entry in this setting is the user-defined variable $WEBSPHERE_HOSTNAME$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
SOAP Port Variable	Specify variable that indicates the SOAP port number on which WebSphere is running. By default, the entry in this setting is the user-defined variable $WEBSPHERE_PORT$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
Admin Security Enabled	Specify variable that indicates whether administrative security is enabled on the WebSphere application server. If this variable is set to Yes at run time, the action uses the credentials in the User Name and Password settings to connect to the server. By default, the entry in this setting is the user-defined variable $WEBSPHERE_IS_SECURITY_ENABLED$. You can specify a different variable if needed. The available values are Yes and No. This setting applies to remote WebSphere servers.
User Name Variable	Specify the variable that indicates the user name for the account that should be used to connect to the WebSphere server if administrative security is enabled. By default, the entry in this setting is the user-defined variable $WEBSPHERE_USERNAME$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.

Table 9-58 ▪ WebSphere Runtime Deployment Customizer (cont.)

Control	Description
Password Variable	Specify the variable that indicates the password that corresponds with the specified user name for an account that should be used to connect to the WebSphere server. By default, the entry in this setting is the user-defined variable $WEBSPHERE_PASSWORD$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
SSL TrustStore Path	Specify the variable that indicates the path to the SSL TrustStore file. By default, the entry in this setting is the user-defined variable $WEBSPHERE_TRUSTSTORE_PATH$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.
SSL TrustStore Password	Specify the variable that indicates the password to the SSL TrustStore. By default, the entry in this setting is the user-defined variable $WEBSPHERE_TRUSTSTORE_PASSWORD$. You can specify a different variable if needed. This setting applies to remote WebSphere servers.

System i (i5/OS) Actions

InstallAnywhere 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 9-59 ▪ System i (i5/OS) Actions

Action	Description
Choose Remote System i (i5/OS) Install Folder Action	Allows the user to choose the installation folder on a remote System i (i5/OS) machine.
Get System i (i5/OS) Login Credentials Action	Requests i5/OS login credentials from the user.
System i (i5/OS) Command Action	Calls an i5/OS command with the specified parameters.
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).

Table 9-59 ▪ System i (i5/OS) Actions (cont.)

Action	Description
System i (i5/OS) Install File Action	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) Action	Enables directories to be installed in the Integrated File System (IFS) on a remote i5/OS system.
System i (i5/OS) Library Action	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 Action	Enables licensed programs to be restored on an i5/OS system.
System i (i5/OS) Object Action	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) Action	Installs an InstallAnywhere Merge Module on the target system.
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.
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.



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 Action

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 9-60 ▪ 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 Action

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 9-61 ▪ 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: `Enter Sign on Credentials (i5/OS System, User, and Password)`
Instructions	Keep the default instructions or customize them by editing the text. The default instructions are: `This installation requires an iSeries system name, user name, and password to continue.`
System Name Prompt	Enter the text to show when this action requests a system name. The default value is: `Please Enter the S&ystem Name:`
User Name Prompt	Enter the text to show when this action requests a user name. The default value is: `Please Enter the &User Name:`
Password Prompt	Enter the text to show when this action requests a password. The default value is: `Please Enter the Pass&word:`
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: • **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 Action

The System i (i5/OS) Command action calls an i5/OS command with the specified parameters. Specify the following options.

Table 9-62 ▪ 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 Action

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 9-63 ▪ System i (i5/OS) Find Component in RAIR Options

Option	Description
Comment	Enter the text the installer shows when this action runs.
Find component	Use these controls to define the comparisons that comprise the query: • **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.

Table 9-63 ▪ System i (i5/OS) Find Component in RAIR Options (cont.)

Option	Description
Set variables	Use these controls to record query results for later access: • **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 Action

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 9-64 ▪ System i (i5/OS) Install File Options

Option	Description
Original	Identify the original file by selecting one of the following options: • **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 InstallAnywhere variable: `$OS400_INSTALL_DIR$$/$`
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.

Table 9-64 ▪ System i (i5/OS) Install File Options (cont.)

Option	Description
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: • 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. • **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) Action

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 9-65 ▪ System i (i5/OS) Integrated File System (IFS) Options

Option	Description
IFS Directory Name	Specify the IFS source directory.

Table 9-65 ▪ System i (i5/OS) Integrated File System (IFS) Options (cont.)

Option	Description
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: ● *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 Action

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 9-66 ▪ 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.

Table 9-66 ▪ System i (i5/OS) Library Options (cont.)

Option	Description
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: ● *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 Action

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 9-67 ▪ 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.

Table 9-67 ▪ System i (i5/OS) Licensed Program Options (cont.)

Option	Description
Language	Specifies the built-in language version (NLV) used for the save operation. Specify one of the following options: ● *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.  ***Note ▪*** *This parameter is ignored if the Object Type is *PGM.*
Release Level	Specifies the version, release, and modification level (VRM) of the licensed program to save. ● *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. ● *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. ● 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.
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: ● *CURRENT ● *PRV ● Other—Enter a six-character release value in **VxRxMx** format (where *x* is a digit 0-9).

Table 9-67 ▪ System i (i5/OS) Licensed Program Options (cont.)

Option	Description
Re-save on Each Build	Choose one of the following options: ● **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 Action

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 9-68 ▪ System i (i5/OS) Object Options

Option	Description
Source Object Names	Enter the names of the objects to save from the source library. Both generic names and specific names can be used. ● 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	Enter the types of objects to save. ● 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.

Table 9-68 ▪ System i (i5/OS) Object Options (cont.)

Option	Description
Source Library Name	Enter the name of the library from which the objects are saved on the i5/OS build system. • 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	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.
Target Release	Specify the minimum supported release level of the operating system on which you intend to restore the objects. Choose one of the following options: • *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) Process Remote Install (Merge Modules) Action

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 9-69 ▪ 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.

Table 9-69 ▪ System i (i5/OS) Process Remote Install (Merge Modules) Options (cont.)

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. `$OS400_INSTALL_DIR$` 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	Specify how you want the merge module's installation progress to be displayed: ● To display an intermediate progress bar on a separate pop-up dialog box while this merge module is being installed, select this check box. ● To suppress the separate merge module progress bar and integrate the progress of the merge module installation into the progress bar of the parent installer, clear this check box.
InstallAnywhere Variables to be passed between this installer and the Merge Module	Click Edit Variables to open the Edit Advertised Variables dialog box.



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, select No in the Update the Product Registry setting in the General Settings view on the Project page.*

System i (i5/OS) Program Action



Note ▪ *The System i (i5/OS) Program action is available in the Install, Pre-Install, and Post-Install sequences.*

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 9-70 ▪ 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	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: ● *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) Action

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 9-71 ▪ 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: ● 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).

Table 9-71 ▪ System i (i5/OS) Program Temporary Fix (PTF) Options (cont.)

Option	Description
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: ● *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: ● *CURRENT ● *PRV ● Other—Enter a six-character release value in the format **`VxRxMx`**, where *x* is a digit (0-9).



Note ▪ *For more information on i5/OS commands and parameters, see the i5/OS Control Language Programming Guide.*

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 that InstallAnywhere users have developed. If you have written a plug-in that you think would be useful to other developers, and you would like to share it, contact the InstallAnywhere Support team.*

The following three plug-ins are installed with InstallAnywhere:

- ExecuteAsRoot Action
- ExtractToFile Action
- Properties File Reader Action

ExecuteAsRoot Action

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. You can download Expect from http://expect.nist.gov.

InstallAnywhere Input Variables

The following input variables are used with the ExecuteAsRoot plugin action:

Table 9-72 ▪ 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 9-73 ▪ 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 Action

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 9-74 ▪ 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: `com/acme/myfile.txt`
ExtractToFile_Destination	Path and name of new file to create. For example: `$USER_INSTALL_DIR$/myfile.txt`

InstallAnywhere Output Variables

None.

Properties File Reader Action

The Properties File Reader 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 Properties File Reader plugin action:

Table 9-75 ▪ InstallAnywhere Input Variables Used with the Properties File Reader Plug-In

Variable	Description
`$PROPERTIES_FILE_LOCATION$`	The location of the properties file.
`$SUBSTITUTE_VARIABLES$`	(Optional) Set to **`TRUE`** if you want to substitute InstallAnywhere variables found in values. Set to **`FALSE`** if you do not want to substitute. **`TRUE`** is the default value.
`$PROPERTIES_TO_IGNORE$`	(Optional) Set to a comma-separated list of properties in the properties file to ignore. For example: **`USER_INSTALL_DIR,MY_PROP_1`**
`$OVERRIDE_EXISTING_VARIABLES$`	(Optional) Set to **`TRUE`** if you want to override existing variables with properties from the properties file. Set to **`FALSE`** if you do not want to override existing variables.

InstallAnywhere Output Variables

None.

Common Action Settings

Many InstallAnywhere action customizers include the same settings. Information on these common settings is grouped into the following sections.

Table 9-76 ▪ 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 9-77 ▪ 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 executable files that are installed
Installed File/Existing File	Determines whether the file is being installed, or already exists on the end user's system
Override default Unix / 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
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/uninstallers may show the installation/uninstallation steps through a set of labels—words which represent the step—or by displaying specific images for the steps. Whether an installer/uninstaller displays labels or images is determined by the Type of Additions to the Installer Panels setting in the **Installer Steps** area of the **Look & Feel** view on the **Installer UI** page:

- **Images**—If the **Images** is selected, the customizer for the panel in the **Pre-Install**, **Post-Install**, **Pre-Uninstall**, and **Post-Uninstall** sequences enables the use of the **Image Settings** tab.
- **List of installer steps**—If the **List of installer steps** is selected, the customizer for the panel in the **Pre-Install**, **Post-Install**, **Pre-Uninstall**, and **Post-Uninstall** sequences enables the use of the **Label Settings** tab.

Also,if the **Enable installer help** and **Use different help text for each panel** options in the **Help** view on the **Installer UI** page are selected, the **Help Settings** tab on panel action customizers is enabled.

Image Settings

Use the Image Settings tab to specify what image your installers/uninstallers 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 9-78 ▪ Image Settings Tab

Option	Description
Use the Installer's Default Image	Select this option to use the image specified in the default Installer/Uninstaller Panel Image on the Installer Steps tab.
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)	Select this option to use an image file you provide.  ***To specify an image:*** 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**.  ***Note ▪*** *The size of the install/uninstall progress panel is 170 pixels wide by 305 pixels tall. Installer/Uninstaller dimensions may change slightly by platform to better display text and different fonts.*

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/uninstallation progresses. The installer/uninstaller build process will auto populate the list based on the panel titles.

Table 9-79 ▪ 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/uninstall.
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/uninstallation. To verify the label appearance, click Preview.



Note ▪ *Note the following in relation with step labels specification and configuration for both the installer and uninstaller:*

- *Using the **Installer Steps** area in the **Look & Feel** view on the **Installer UI** page and the **Label Settings** tabs found on each individual panel's customizer, you can assign multiple panels to the same label. Thus, if there are numerous steps, or if the installer/uninstaller has several panels for the same step, the interface can be adjusted as needed.*
- *Configuration of icons and texts font colours are common for both the uninstaller step lables and installer step lables. On the **Installer UI > Look & Feel Settings > Customer UI Designer** dialog box, under the **Installer Steps** area, using the **Installer Step Labels settings** button in the **Configure Install Labels** setting enables to you to configure the icons and texts font colours.*

Help Settings

On the Help Settings tab, you can specify customized help for this action.



Note ▪ *The Help Settings tab is enabled only if the **Enable installer help** and the **Use different help text for each panel** options are selected in the **Help** view on the **Installer UI** page.*

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.



Important ▪ *If you enable installer/uninstaller help and supply help for installer/uninstaller panels, do not include* `<META>` *tags in the HTML code of the help panels. An InstallAnywhere installer/uninstaller is unable to display help pages that contain* `<META>` *tags.*

Enabling Installer Help

InstallAnywhere lets you add help to your installer/uninstaller. You can choose to use the same text for all help panels or customized help text for each panel.



Task ***To enable installer help:***

1. In the Advanced Designer, on the **Installer UI** page, click **Help**. The **Help** view opens.
2. Select the **Enable installer help** check box.
3. In the **Help Text Format** area, select either **HTML** or **Plain text**. Selecting **HTML** allows greater formatting control of the message using HTML formatting tags. For example:

HTML	Display
`<B>MyHelp</B> <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/uninstaller panel on the **Help Settings** tab of the action customizer for the **Pre-Install**, **Post-Install**, **Pre-Uninstall**, and **Post-Uninstall** sequences 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

There are two versions of the Get User Input panel that exist as separate actions:

Table 9-80 ▪ Get User Input Panels

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 - Console	Allows developers to request input from the end user in console action.

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 9-81 ▪ 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.
Input Method	Lists the types of input methods the Get User Input - Simple panel can render. • **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	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: • 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. The specific settings available on the Configure Input Items dialog box depend on the current Input Method setting.
Preview	Shows a preview of the panel as it is currently configured.

Table 9-81 ▪ Get User Input - Simple Panel Controls (cont.)

Control	Description
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 9-82 ▪ 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 9-82 ▪ Get User Input - Advanced Panel Controls (cont.)

Control	Description
Configure Selection	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. • **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, manage the display of the **Restore Default** button, 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.

Get User Input - Console

This action 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 - console, you provide a console title, prompt, and input method such as Question, Multiple Choice, or Single Choice. However, this action only allows you to provide one type of control.

Table 9-83 ▪ - Get User Input - Console 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.
Input Method	Lists the types of input methods the Get User Input - Console can render. • **Question**-Shows one or more questions, accepts text typed into the field, and stores those strings in the results variable for this panel. • **Multiple Choice**-Shows one or more choices, accepts user input by typing choice number (multiple choice can be done by typing choice number comma separated), and stores those values in the results variable. • **Single Choice**-Shows one or more choices, accepts user choice by typing choice number, and records those choices in the results variable.
Configure	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. In general, this dialog box allows you to: • Add or remove controls of the type set in Input Method. • Define labels and default values for each control. The specific settings available on the Configure Input Items dialog box depend on the current Input Method setting.
Results Variable	Names the results variable in which the user input is stored.

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-based or OS or OS X–based 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 9-84 ▪ Built-in Rules

Rule	Description
Check File/Folder Attributes Rule	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	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	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. This rule also lets you customize the events that you want to occur when the installer is running in upgrade mode.
Check System Architecture Rule	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	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	This rule allows developers to compare the timestamp of an existing target file in order to make an overwrite decision.
Compare InstallAnywhere Variables Rule	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.
Compare InstallAnywhere Variables Numerically Rule	Use to compare two InstallAnywhere variables numerically or to compare an InstallAnywhere variable against a specific value.
Compare Versions	Use to compare two version numbers during an installation.

Table 9-84 ▪ Built-in Rules (cont.)

Rule	Description
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.
Match Regular Expression Rule	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. The **Match Regular Expression Rule** uses the **java.util.regex** library. For more information, see: https://docs.oracle.com/javase/8/docs/api/java/util/regex/package-summary.html
System i (i5/OS) Licensed Program Exists Condition Rule	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.
System i (i5/OS) Primary Language Install Condition Rule	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	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 9-85 ▪ 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.)

Table 9-85 ▪ Controls on the Check File/Folder Attributes Rule Customizer (cont.)

Control	Description
This Install Path	Inserts the install path for the currently selected file or folder action. (Available only if this rule is assigned to a file- or folder-related action in the Install sequence.)
Perform only if the file/ folder ...	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. **Already Exists | Does Not Already Exist** Select one of the following options: ● 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. **File |Folder** Select one of the following options: ● 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. **Readable | Writable | Both** Select one or both of the following options: ● 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. **In-Use | Not In-Use** Select one of the following options: ● 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 9-86 ▪ 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 9-87 ▪ Check Platform Rule Customizer

Option	Description
Do Not Perform On	Lists platforms on which the action is not performed. 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.
Perform On	Lists platforms on which the action is permitted.  ***Note*** ▪ *You can use the Check Platform rule to specify that an action, panel, or console should only be executed on Windows 8 / Windows Server 2012 platforms. To add Windows 8 / Windows 2012 Server to the Perform On list, click More Platforms to open the Add Platform dialog box, and then select Windows 8 from the list.*
-->	Moves the platforms currently selected in the Do Not Perform On list to the Perform On list.

Table 9-87 ▪ Check Platform Rule Customizer (cont.)

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 Platform	Deletes the currently selected platform from both the Do Not Perform On and the Perform On lists.

Check Running Mode Rule

When you enable maintenance mode, InstallAnywhere automatically adds Check Running Mode rules to the action groups that are created in the Pre-Install and Pre-Uninstall sequences.

In addition, you can also manually apply a Check Running Mode rule to action groups, actions, or panels in your project to specify whether that element should be executed during install mode, upgrade mode, or maintenance mode. This enables you to customize the events that occur when an end user launches various modes.



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 9-88 ▪ 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 sequences: • 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 are executed.

Table 9-88 ▪ Check Running Mode Rule Customizer (cont.)

Option	Description
Add Features	Select this option to associate the selected element with the Add Features phase of maintenance mode. This option is available in the following sequences: ● Pre-Install ● Install ● Post-Install 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 are executed.
Repair Installation	Select this option to associate the selected element with the Repair Installation phase of maintenance mode. This option is available in the following sequences: ● Pre-Install ● Install ● Post-Install 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 are executed.
Uninstallation	Select this option to associate the selected element with the Uninstallation phase of maintenance mode. This option is available in the following sequences: ● Pre-Uninstall ● Uninstall ● Post-Uninstall 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 are executed.
Remove Features	Select this option to associate the selected element with the Remove Features phase of maintenance mode. This option is available in the following sequences: ● Pre-Uninstall ● Uninstall ● Post-Uninstall 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 are executed.

Table 9-88 ▪ Check Running Mode Rule Customizer (cont.)

Option	Description
Upgrade	Select this option to associate the selected element with upgrade mode. This option is available in the following sequences: • Pre-Install • Install • Post-Install For example, when an end user launches an upgrade to uninstall an earlier version of the product and then install the new version, only those actions and panels with a Check Running Mode rule set to Upgrade are 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.*



Note ▪ *If a panel in the Pre-Install sequence does not have a Check Running Mode rule assigned either to itself or any of its parents, this panel is run during Install, Maintenance Mode/Add Features, and Maintenance Mode/Repair Installation.*

If a panel in the Pre-Uninstall sequence 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.

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 9-89 ▪ 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`**).

Table 9-89 ▪ Compare InstallAnywhere Variables Numerically Customizer (cont.)

Option	Description
[Operator]	From the list, select one of the following operators: • greater than or equal to (>=) • equal to (==) • less than (<) • less than or equal to (<=) • greater than (>) • greater than or equal to (>=)

Compare Versions

The **Compare Versions** rule enables you to specifically compare two version numbers during an installation. This rule is displayed on the **Choose a Rule/Expression** dialog box that opens when you click **Add Rule** on the **Organization** or **Sequence** page.

When you add a **Compare Versions** rule, you are prompted to enter two operands and an operator in the Compare Versions **Properties Customizer**. Both operands may be expressed as either an InstallAnywhere variable being resolved (such as `$VARIABLE1$`) or as a literal version number string (such as `1.0.0.0`).

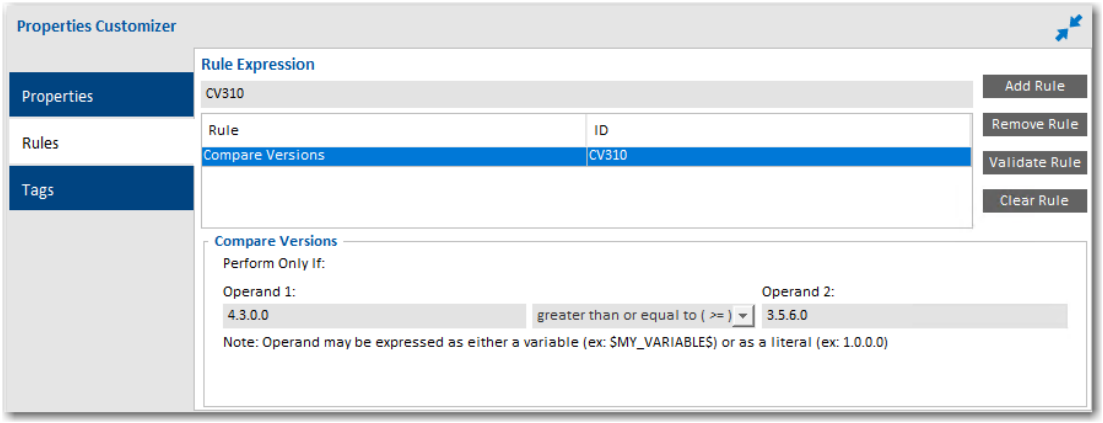


Figure 9-2: Compare Versions Rule Properties Customizer

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 9-90 ▪ Evaluate Custom Rule Customizer

Option	Description
Path	Click the Choose JAR or ZIP button and browse for the .jar or .zip file.
Class	Enter the fully qualified custom rule class name (such as `com.acme.MyCustomCodeRule`).  ***Note ▪*** *This class must extend* `com.zerog.ia.api.pub.CustomCodeRule` *and must implement a* `public boolean evaluateRule()` *method.*
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.

Variables

InstallAnywhere has support for the following types of variables.

Table 9-91 ▪ InstallAnywhere Variables Reference

Type	Description
Standard InstallAnywhere Variables	Standard InstallAnywhere variables are built-in variables that you can use to represent alterable information.
LAX Properties	LAX properties are variables that you can use to configure a LaunchAnywhere executable file.
Magic Folders	Magic folders are variables that define common locations on target systems.

Standard InstallAnywhere Variables

InstallAnywhere includes a number of default variables that store information that is essential to the installation process. The following variables are standard InstallAnywhere variables:

Table 9-92 ▪ 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**.	Read-write

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$CHOSEN_FEATURE_LIST$	The *$CHOSEN_FEATURE_LIST$* variable is a comma-separated list of all the end user–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. 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. 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.  ***Note ▪*** *Use this variable, along with $CHOSEN_FEATURE_n$ and $CHOSEN_FEATURE_NUM$, to track the features that your end users select during install and uninstall.* *A similar set of variables exists ($CHOSEN_INSTALL_FEATURE_LIST$, $CHOSEN_INSTALL_FEATURE_LIST_LONG$, $CHOSEN_INSTALL_FEATURE_NUM$, $CHOSEN_INSTALL_FEATURE_n$, and $FEATURE_UNINSTALL_LIST$). However, you can use the CHOSEN_FEATURE set of variables at install and uninstall by specifying features' short names, which are not localizable and thus serve as consistent IDs, regardless of the locale in which the installer or uninstaller is running.*	Read-write
$CHOSEN_FEATURE_LIST_LONG$	A new read only variable $CHOSEN_INSTALL_FEATURE_LIST_LONG$ is added under the InstallAnywhere Variables list which contains the long names of the installed Features. This variable can be used to get the long name of the feature from the feature short name.	

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$CHOSEN_FEATURE_n$	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. For example, if *$CHOSEN_FEATURE_NUM$* equals **2**, two variables of this form are created: *$CHOSEN_FEATURE_1$* and *$CHOSEN_FEATURE_2$*.  ***Note ▪*** *Use this variable, along with $CHOSEN_FEATURE_LIST$ and $CHOSEN_FEATURE_NUM$, to track the features that your end users select during install and uninstall.* *A similar set of variables exists ($CHOSEN_INSTALL_FEATURE_LIST$, $CHOSEN_INSTALL_FEATURE_LIST_LONG$, $CHOSEN_INSTALL_FEATURE_NUM$, $CHOSEN_INSTALL_FEATURE_n$, and $FEATURE_UNINSTALL_LIST$). However, you can use the CHOSEN_FEATURE set of variables at install and uninstall by specifying features' short names, which are not localizable and thus serve as consistent IDs, regardless of the locale in which the installer or uninstaller is running.*	Read-write
$CHOSEN_FEATURE_NUM$	This variable holds the total number of features (as a string) that the end user chooses to install. 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 *$CHOSEN_FEATURE_3$*.  ***Note ▪*** *Use this variable, along with $CHOSEN_FEATURE_n$ and $CHOSEN_FEATURE_LIST$, to track the features that your end users select during install and uninstall.* *A similar set of variables exists ($CHOSEN_INSTALL_FEATURE_LIST$, $CHOSEN_INSTALL_FEATURE_LIST_LONG$, $CHOSEN_INSTALL_FEATURE_NUM$, $CHOSEN_INSTALL_FEATURE_n$, and $FEATURE_UNINSTALL_LIST$). However, you can use the CHOSEN_FEATURE set of variables at install and uninstall by specifying features' short names, which are not localizable and thus serve as consistent IDs, regardless of the locale in which the installer or uninstaller is running.*	Read-write

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$CHOSEN_INSTALL_BUNDLE_LIST$	This variable is a comma-separated list of all install features (short name) chosen for the installation.  ***Note ▪*** *This variable is deprecated. Instead, use* *$CHOSEN_INSTALL_FEATURE_LIST$.*	Read-only
$CHOSEN_INSTALL_BUNDLE_n$	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. For example, if *$CHOSEN_INSTALL_BUNDLE_NUM$* equals 2, then two variables of this form are created: *$CHOSEN_INSTALL_BUNDLE_1$* and *$CHOSEN_INSTALL_BUNDLE_2$*.  ***Note ▪*** *This variable is deprecated. Instead, use* *$CHOSEN_INSTALL_FEATURE_n$.*	Read-only
$CHOSEN_INSTALL_BUNDLE_NUM$	If Choose Product Features is enabled, this variable holds the total number of components (as a string) that the end user chooses to install.  ***Note ▪*** *This variable is deprecated. Instead use* *$CHOSEN_INSTALL_FEATURE_NUM$.*	Read-only

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$CHOSEN_INSTALL_FEATURE_LIST$	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. (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.) 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.	Read-write
$CHOSEN_INSTALL_FEATURE_LIST_LONG$	This variable is a comma-separated list of all the 'selected' features (long names).	Read-only
$CHOSEN_INSTALL_FEATURE_n$	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 *$CHOSEN_INSTALL_FEATURE_2$*.	Read-write
$CHOSEN_INSTALL_FEATURE_NUM$	If Choose Product Features is enabled, this variable holds the total number of features (as a string) that the end user chooses to install.	Read-write
$CHOSEN_INSTALL_SET$	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 that are sent to the LaunchAnywhere executable.	Used by LaunchAnywhere
$COMMA$	This resolves to a comma (,).	Read-only

Table 9-92 ▪ 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.	Read-write
$componentname_MATCHED_KEY_FILE$	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.	Read-write
$DEPENDENCY_FAILURES$	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.	Read-write
$DEPENDENCY_REPORT$	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.	Read-write
$DEPENDENCY_STATUS$	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.	Read-write
$DEPENDENCY_SUCCESSES$	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.	Read-write
$DEVELOPER_DISK_SPACE_ADDITIONAL$	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.	Read-write
$DOLLAR$	This resolves to $.	Read-only

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$EMPTY_STRING$	This variable represents a value of null. This makes *$EMPTY_STRING$* useful to determine whether any variables have been initialized. Variables that have not yet been initialized will have this as their value.	Read-only
$EXTRACTOR_DIR$	The full path to the directory from where the self-extractor executable file was launched. *$EXTRACTOR_DIR$* is available for the following target systems: • AIX • HP-UX • Linux • Solaris • UNIX • Windows See also the *$INSTALLER_LAUNCH_DIR$* variable, which is available for OS or OS X–based installers and pure Java–based installers.	Read-only
$EXTRACTOR_EXECUTABLE$	The full path to the self-extractor executable file in the location where it was launched.	Read-only
$FEATURE_UNINSTALL_LIST$	The *$FEATURE_UNINSTALL_LIST$* a comma-delimited list of features (display name, not short name) that the end user chose to uninstall.	Read-write
$FREE_DISK_SPACE_BYTES$ **$FREE_DISK_SPACE_KILOBYTES$** **$FREE_DISK_SPACE_MEGABYTES$** **$FREE_DISK_SPACE_GIGABYTES$**	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 sequence.	Read-only
$IA_BROWSE_FOLDERS$	Controls the use of Swing or built-in resources for rendering the Browse for Folder dialog box. For more information, see $IA_BROWSE_FOLDERS$ Variable.	Read-write
$IA_CLASSPATH$	The classpath as specified in the InstallAnywhere Advanced Designer environment.	You can set this in the Advanced Designer

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$IA_CONDITIONAL_LOG$	Setting this variable to a specific value generates a log file on a specified condition during installation and uninstallation. This variable can be set to the following values: • SUCCESS • WARNING • ERROR • INCOMPLETE • SKIPPED • CANCELLED	Read-write
$IA_CUSTOM_RESPONSE_FILE_PATH$	Setting this variable to a required or custom directory generates a response file in the same specified directory during installation. To set this variable, use either the **Set InstallAnywhere Variable – Single Variable** or **Set InstallAnywhere Variable – Multiple Variables** action within the **Pre-Install**, **Install**, or **Post-Install** views on the **Sequence** page. To effectively use the *$IA_CUSTOM_RESPONSE_FILE_PATH$* variable, the **Always Generate Response File** setting in the **Project Information** area on the **General Settings** view must be set to **Yes**.	Read-write
$IA_GLOBAL_REG_LOCATION$	Holds the path for global InstallAnywhere registry location and prints via the Show Message dialog box. This variable can be used to determine where the global registry is present on the target machine.	Read-only
$IA_INSTALL_LOG$	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: • **TRUE**—The uninstaller is running in maintenance mode. • **FALSE or empty**—The uninstaller is running in standard mode.	Read-write

Table 9-92 ▪ 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: • **ADD**—Add Features mode. • **REMOVE**—Remove Features mode. • **REPAIR**—Repair Features mode. • **UNINSTALL**—Uninstall mode.	Read-write
$IA_RESPONSEFILE_PATH$	Stores the complete location of a response file being used by the installer.	Read-write
$IA_ROLLBACK$	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 custom code encounters an exception of type FatalInstallException. Otherwise, it is set to FALSE or is empty.	Read-write
$INSTALL_LOG_DESTINATION$	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.	Read-write
$INSTALL_LOG_NAME$	This variable sets the name of the installation log. If not set, the installation log name is based on the product name.	Read-write
$INSTALL_SUCCESS$	This variable alerts the end user if the installation was successfully completed or if it failed. Available values for this variable are: • SUCCESS • WARNING • NONFATAL_ERROR • FATAL_ERROR	Read-only
$INSTALLER_LAUNCH_DIR$	The full path to the directory from where the self-extractor executable file was launched. This variable is available for pure Java–based installers and OS or OS X–based installers. See also the *$EXTRACTOR_DIR$* variable, which is available for other platforms.	Read-only

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$INSTALLER_LOCALE$	The locale as a string (see java.util.Locale.toString()) that the end user selected at the beginning of the installation.	Read-only
$INSTALLER_TITLE$	This is the title of the installer; it is displayed in the title bar.	Read-write
$INSTALLER_UI$	This resolves to the UI mode for the installer.  ***Note ▪*** *The `$INSTALLER_UI$` variable is case-sensitive when used in a rule, but is not case-sensitive when used in a response file:* • **Rule**—*If you use 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` evaluates correctly, but `$INSTALLER_UI$=silent` does not evaluate correctly.* • **Response file**—*If you use the `$INSTALLER_UI$` variable in a response file and pass the response file to the installer, `$INSTALLER_UI$=silent` evaluates correctly.*	Read-only, but you can 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
$lax.nl.env.ENVIRONMENT_VARIABLE_NAME$	 ***Note ▪*** *Windows and UNIX only.* Access any system environment variable (for example, access PATH via *`$lax.nl.env.PATH$`*) by specifying the variable name as an all uppercase string. These variables are resolved at application runtime, when LaunchAnywhere executes. Developers can also access system environment variables via LaunchAnywhere properties.	Read-only

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$lax.nl.env.exact_case.ENVIRONMENT_VARIABLE_NAME$	 ***Note ▪*** *Windows and UNIX only.* Access any system environment variable (for example, access Path via *$lax.nl.env.exact_case.Path$*) 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.	Read-only
$NEVER_UNINSTALLS_VM$	Tells the uninstaller to never uninstall a Java Virtual Machine. This is useful when multiple projects and uninstallers share one virtual machine. Set to TRUE to never uninstall a JVM.	Read-write
$NULL$	This is equivalent to $EMPTY_STRING$.	Read-only
$PRODUCT_ID$	Resolves to the value of the Product Code setting (Project page > General Settings view).	Read-only
$PRODUCT_NAME$	This is the product name.	Read-write
$PRODUCT_SHORT_NAME$	Setting this variable to a desired or custom short name of the product displays the same name in the Install, Uninstall, and Maintenance Mode panels during installation, uninstallation, and maintenance mode, respectively. To set this variable, use either the **Set InstallAnywhere Variable – Single Variable** or **Set InstallAnywhere Variable – Multiple Variables** action within only the **Pre-Install** view on the **Sequence** page.  ***Note ▪*** *If the $PRODUCT_SHORT_NAME$ variable is not defined, InstallAnywhere displays the product's default name in the Install, Uninstall, and Maintenance Mode panels during installation, uninstallation, and maintenance mode, respectively.*	Read-write
$PRODUCT_VERSION_NUMBER$	Resolves to the value of the Product Version (Project page > General Settings view).	Read-only

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$PROMPT_USER_CHOSEN_OPTION$	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.	Read-write
$REQUIRED_DISK_SPACE_BYTES$ **$REQUIRED_DISK_SPACE_KILOBYTES$** **$REQUIRED_DISK_SPACE_MEGABYTES$** **$REQUIRED_DISK_SPACE_GIGABYTES$**	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 sequence.	Read-only
$RESTART_NEEDED$	This variable tells the installer or uninstaller whether the Windows-based target system needs to be restarted to complete the installation or uninstallation. A restart may be recommended or required in the following conditions: • The installer or uninstaller is attempting to modify, rename, or delete a file that is locked. • The installer or uninstaller is attempting to copy or move a file to a location that is locked. • The installer or uninstaller is attempting to clean up associated temporary files or folders but one or more of those temporary items are locked. Available values are: • **NO**—A restart is not needed. • **YES_RECOMMENDED**—This value indicates that the installer or uninstaller is encountering locked files or folders. • **YES_REQUIRED**—This value indicates that the installer or uninstaller is encountering locked files or folders through InstallShield MultiPlatform API Win32Service code, and the code indicates that a restart is required.	Read-Write
$SHORTCUT_NAME$	This variable resolves to "Shortcut" on Windows-based systems, "Alias" on OS or OS X–based systems, and "Link" on all other systems.	Read-only

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$SKIP_UNINSTALL$	Tells the uninstaller whether 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.*	Read-write
$UNINSTALL_SUCCESS$	The same as *$INSTALL_SUCCESS$*, but for the uninstaller. Available values are: • SUCCESS • SKIPPED • INCOMPLETE	Read-only
$UNINSTALLER_TITLE$	This is the title of the uninstaller displayed in the title bar.	Read-write
$IA_INSTANCE_MANAGEMENT_OPTION$	This variable is set by the choice an end user makes on the Manage Instances dialog box (or equivalent console panel), or during an upgrade. Available values are: • **NEW**—Indicates that the end user chose the Install a New Instance option. • **MODIFY**—Indicates that the end user chose the Modify an Existing Instance option. • **NOT_DEFINED (or empty)**—Variable is not defined. This variable is also used in upgrade mode. In upgrade mode, the value of this variable is: • **UPGRADE**—Indicates that an instance is being upgraded.  ***Note ▪*** *This is a read-only value that is you can use as a standard InstallAnywhere variable. This variable is not serialized into the `installvariables.properties` file or the response files.*	Read-only

Table 9-92 ▪ Standard InstallAnywhere Variables (cont.)

Variable	Description	Status
$IA_INSTANCE_MODIFY_PATH$	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). Available values are: ● **[Path to instance being modified]**—When the user has chosen an instance to modify ● **NOT_DEFINED (or empty)**—Variable is not defined.  ***Note ▪*** *This is a read-only value that is you can use as a standard InstallAnywhere variable. This variable is not serialized into the `installvariables.properties` file or the response files.*	Read-only
$IA_INSTALL_INSTANCE_NUM$	This variable reflects the number of the instance of the product that is being installed. Available values are: ● **[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.  ***Note ▪*** *This is a read-only value that is you can use as a standard InstallAnywhere variable. This variable is not serialized into the `installvariables.properties` file or the response files.*	Read-only
$IA_UPGRADE_BASE_VERSION$	This variable indicates the version number of the base version that is being updated.	Read-only
$IA_UPGRADE_BASE_LOCATION$	This variable indicates the version number of the base product that is being updated.	Read-only
$IA_8DOT3_FILENAMECREATION_STATE$	This variable identifies whether the 8.3 File creation is enabled or not for all volumes. Available values are: ● `TRUE` - The 8.3 File creation is enabled. ● `FALSE` or empty - The 8.3 File creation is disabled.	

$IA_BROWSE_FOLDERS$ Variable

The *$IA_BROWSE_FOLDERS$* variable controls whether InstallAnywhere-generated installers use Swing or built-in 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 9-93 ▪ $IA_BROWSE_FOLDERS$ Variable Values

Value	Description
Native	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 that are running in languages other than the target system's locale display text in that system's locale rather than in the installer-selected language. For example, if an end user who is installing an application on a target operating system using the English locale chooses German as the selected language in the installer, some English strings appear on the Browse for Folder dialog box.
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 built-in experience for end users. It may have a different layout than the standard Browse for Folder dialog box that is rendered by the end user's operating system.



Note ▪ *If the $IA_BROWSE_FOLDERS$ variable is set to any value other than Native or Swing, the installer uses native resources.*

Comparison Between Native and Swing Settings

The following images illustrate the differences between the Browse for Folder dialog box that is rendered with native resources versus one that is rendered with Swing resources on Windows-based and Linux-based systems.

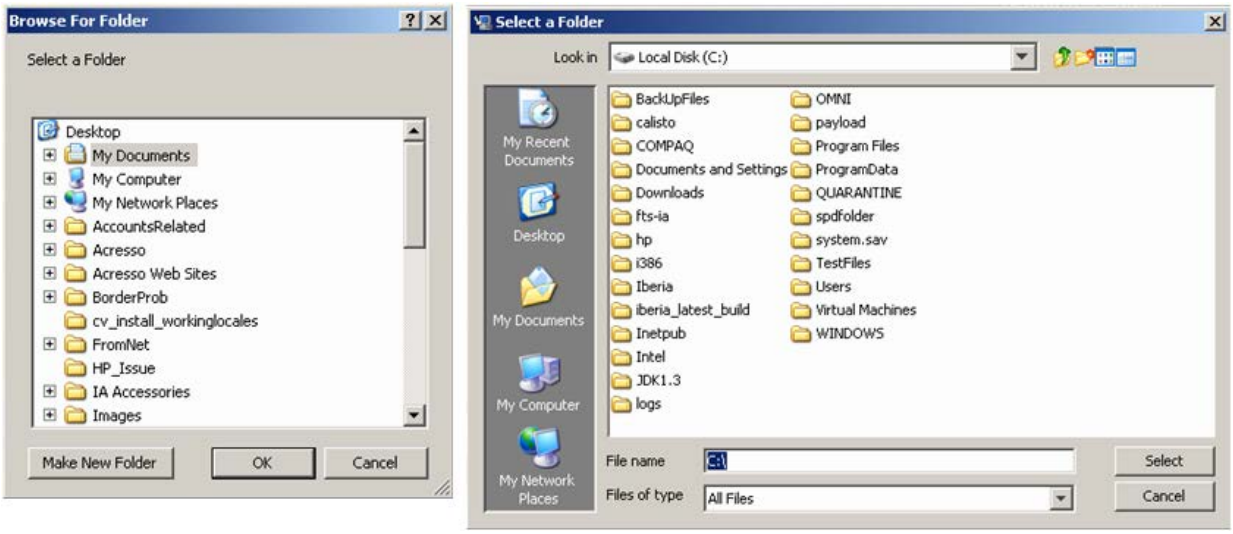


Figure 9-3: Native (Left) vs. Swing (Right) on Windows

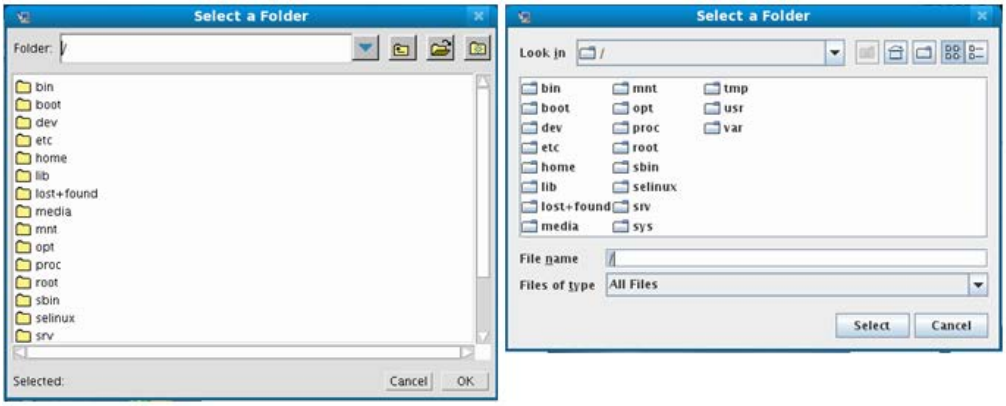


Figure 9-4: Native (Left) vs. Swing (Right) on Linux

Changing the Look and Feel for OS or OS X–Based Systems with Certain Versions of JRE

The Change Media run-time panel has a Browse button that should enable end users to browse to the next installation media that they want to use. Apple OS or OS X–based systems that use Oracle Java 7 with an update earlier than update 45 do not support built-in resources for the button on that panel, so the browse button does not work as expected. If you want to use Swing resources instead of built-in resources for that panel and avoid problems with the browse button, you can set the ia..filechooser.substituteSwingInsteadOfNativeForOlderJRE7 variable to true for installers that target OS or OS X–based systems that use Oracle Java 7 with an update earlier than update 45. The default value for this property is false.

Task ***To specify that you want to use the Swing Version of the Change Media panel:***

1. In the Advanced Designer, on the **Project** page, click **JVM Settings**. The **JVM Settings** view opens.
2. Click the **Installer Settings** tab.

3. In the **Optional Installer Arguments** area, in the **Additional Arguments** box, enter the following:

   ```
   -Dia..filechooser.substituteSwingInsteadOfNativeForOlderJRE7=true
   ```

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 sequence.

Table 9-94 ▪ LAX Properties

Property	Definition
lax.application.name	The name of the application that the launcher executes.
lax.class.path	The classpath for the application. By default, set to *$IA_CLASSPATH$* (the classpath specified in the InstallAnywhere Designer environment.) 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.  ***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.*
lax.command.line.args	A list of arguments that are 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 **java myApp arg1 arg2**, set this property to the following: **"arg1 arg2"** Be sure to place quotes around any arguments that have spaces. When it is necessary to pass in an argument that is known only at install time (for instance, the installation directory), use an InstallAnywhere variable.
lax.dir	The path to directory holding LaunchAnywhere's built-in launcher. If you are specifying a path on Windows-based systems, use escaped backslashes (such as `C:\\Program Files\\OfficeSuite.exe`).

Table 9-94 ▪ LAX Properties (cont.)

Property	Definition
lax.java.compiler	The JIT compiler that is being used for execution of this application.  ***Note*** *▪ Runtime only. The* `lax.java.compiler` *property cannot be set via the LaunchAnywhere Properties or Uninstaller Properties dialog boxes.*
lax.main.method	The name of the application's starting method that this LaunchAnywhere Executable invokes.
lax.main.class	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.
lax.nl.current.vm	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`.
lax.nl.env.*variable_name* **lax.nl.env.exact_case.***variable_name*	Use to access any system environment variable. • **Case-insensitive**—If you want to specify the property name as a case-insensitive string, use `lax.nl.env.`*variable_name*. For example, to access the system environment variable *PATH*, you could specify **lax.nl.env.PATH** or **lax.nl.env.path**. • **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 **lax.nl.env.exact_case.variable_name**. For example, to access system environment variable *Path*, specify **lax.nl.env.exact_case.Path**.  ***Note*** *▪ These properties are resolved at application run time, when LaunchAnywhere executes. You can also get access to system environment variables via InstallAnywhere variables.* *These properties are available on Windows-based systems and UNIX-based systems.*
lax.nl.env.path	The system *PATH* for the computer this application is running on.  ***Note*** *▪ Runtime only. The* `lax.nl.env.path` *property cannot be set via the LaunchAnywhere Properties or Uninstaller Properties dialog boxes.*

Table 9-94 ▪ LAX Properties (cont.)

Property	Definition
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 that are 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 **-Dmyvariable=value**.
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 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. Available values are: • **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 `-ms` or `Xms`. The default is 16777216 (16 MB).
lax.nl.java.option.java.heap. size.max	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 `-mx` or `Xmx`. The default is 50331648 (48 MB).
lax.nl.java.option.verbose	Defines the level of content in output messages. Available values are: • **gc**—Output garbage collection messages. • **normal**—Output all normal verbose messages. • **all**—Output all normal and garbage collection messages. • **none**—Do not output any verbose messages.

Table 9-94 ▪ LAX Properties (cont.)

Property	Definition
lax.nl.java.option.verify.mode	Sets when Java will verify classes for security and errors. Available values are: ● remote ● all ● none
lax.nl.message.vm.not.loaded	Defines the message to show the end user in a dialog box if no VM can be found.
lax.nl.valid.vm.list	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 that InstallAnywhere supports. Some common criteria include: ● 11.0.* ● 17.0+ ● 21.0* The value of this property overrides the value that is listed in `lax.nl.current.vm` if the VM that is listed in that property is not of a valid type. The order of the valid VM list specifies the precedence in which VMs that are found on the system should be chosen if a valid VM is not listed in `lax.nl.current.vm`.  ***Note*** ▪ *For details on valid VM selection criteria, see About Java VM Selection Criteria.*
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	The location of your application's stderr 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 setting in the Advanced Designer (Project page > General Settings view > Log Settings area): Send stderr to.

Table 9-94 ▪ 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 setting in the Advanced Designer (Project page > General Settings view > Log Settings area): Send stdout to.
lax.user.dir	The working directory for your application. The default value for this property is a 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 that are available to the Java application. It can take any value. You must redirect stdout and stderr to see the results of this output.
lax.highdpi.scaling.override.system	This property uses System scaling. By default the value is set to 'false'. When set to true the Override High DPI Scaling Override setting in Windows is set to 'System'.

Magic Folders

Each magic folder represents a specific location, such as the end user–selected installation directory, the desktop, or the location for library files. At run time, the installer determines the operating system on which it is running, and it 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 9-95 ▪ 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 9-95 ▪ Magic Folders (cont.)

Folder Name	InstallAnywhere Variable	Destination
Programs Folder	*$PROGRAMS_DIR$*	The default application directory on the destination system. (Program Files on Windows. Applications folder on OS or OS X. Home directory of the end user who is running the installer 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. You can specify a default value for this variable on a per-platform basis in the Platforms view on the Project page in the Advanced Designer.
Desktop	*$DESKTOP$*	This variable represents the desktop on the target machine. This folder only resolves on Windows, Linux, and systems.
Temp Directory	*$TEMP_DIR$*	This variable represents the temp directory on the target machine. When an end user is running the pure Java installer on Windows, *$TEMP_DIR$* resolves to the end 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 $USER_HOME$.
Programs Folder (64-bit)	*$PROGRAMS_DIR_64$*	The default 64-bit application directory on a 64-bit system.

Table 9-95 ▪ Magic Folders (cont.)

Folder Name	InstallAnywhere Variable	Destination
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 9-95 ▪ Magic Folders (cont.)

Folder Name	InstallAnywhere Variable	Destination
USER_MAGIC_FOLDER_*n*	*$USER_MAGIC_FOLDER_n$*	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. For UNIX-based systems, if the leading / is omitted in the path before the Magic Folder location, the location resolves to `<installer location>/ USER_MAGIC_FOLDER_n.` UNIX-based operating systems assume that any path not preceded by a / is below the current directory.
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	*$FONTS$*	The Fonts directory on Windows-based systems; for example, **$WIN_WINDOWS$\Fonts**. For more information, see Installing Fonts.
User Applications	*$X_USER_APPLICATIONS$*	The User Applications directory of the end user running the installer (OS or OS X only).
The Dock	*X_DOCK*	The OS or OS X Dock (for Shortcuts only). Other types of files cannot be installed to the Dock.
/usr/local/bin	*$UNIX_USR_LOCAL_BIN$*	The `/usr/local/bin` directory (UNIX-based systems only).

Table 9-95 ▪ Magic Folders (cont.)

Folder Name	InstallAnywhere Variable	Destination
/opt	*$UNIX_OPT$*	The `/opt` directory (UNIX computers only).
/usr/bin	*$UNIX_USR_BIN$*	The `/usr/bin` directory (UNIX-based systems only).

Localization Reference

InstallAnywhere allows developers to build installers for up to 31 different languages.

Table 9-96 ▪ Localization Reference

Section	Description
Language Codes	Lists languages and language codes supported by InstallAnywhere.
Common Localizable Elements	Identifies many elements that most frequently require localization.

Language Codes

InstallAnywhere supports the following languages.

Table 9-97 ▪ Language Codes

Language	Language Code
Arabic	ar
Basque	eu
Catalan	ca
Chinese (Simplified)	zh_CN
Chinese (Traditional)	zh_TW
Croatian	hr_HR
Czech	cs
Danish	da
Dutch	nl
English	en

Table 9-97 ▪ Language Codes (cont.)

Language	Language Code
Finnish	fi
French	fr
French (Canada)	fr_CA
German	de
Greek	el
Hebrew	iw
Hungarian	hu
Indonesian	in
Italian	it
Japanese	ja
Korean	ko
Norwegian	no
Polish	pl
Portuguese	pt
Portuguese (Brazil)	pt_BR
Russian	ru
Slovak	sk
Slovenian	sl
Spanish	es
Swedish	sv
Thai	th
Turkish	tr



Note ▪ *Some codes shown in the Language Code column, above, 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 9-98 ▪ Language Resource Properties

Property	Definition
`Installer.#.ProductName`	Name of product displayed on installer title bar.
`Installer.#.splashScreenGUITitle`	Title of the installer splash screen. The splash screen title bar only appears when the installer supports more than one locale; otherwise only the splash screen image is shown.
`Installer.#.splashScreenGUIImagePath`	Path to the splash screen image file.
`Installer.#.splashScreenGUIImageName`	File name of the splash screen image file.
`Installer.#.splashScreenGUIConfirm`	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.
`Installer.#.splashScreenGUIInstructions`	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.

Table 9-98 ▪ Language Resource Properties (cont.)

Property	Definition
`Installer.#.splashScreenConsoleTitle`	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.
`Installer.#.splashScreenConsolePrompt`	The text for the prompt on the Choose Locale console. (Default: CHOOSE LOCALE BY NUMBER) The Choose Locale console only appears when the console installer supports more than one locale.
`Installer.#.RulesFailedMessage`	Message displayed if specified rules keep the installer from running.
`Installer.#.ShortcutDestinationPathOS`	Path to where aliases are created during installation on 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`	Name of localized license agreement to be displayed as the installer is preparing itself.  ***Note*** ▪ *This is a filename only—not a fully qualified absolute path name.*

Table 9-98 ▪ Language Resource Properties (cont.)

Property	Definition
`LicenseAgr.#.Path`	Path name to localized license agreement to be displayed as the installer is preparing itself.  ***Note ▪*** *This is only a path name and does not include the filename itself.*
`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.#.Title`	Title of OS or 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`	Name of billboard image file to be displayed as the installer is preparing itself.  ***Note ▪*** *This is a filename only—not a fully qualified absolute path name.*
`Billboard.#.ImagePath`	Path name to billboard image to be displayed as the installer is preparing itself.  ***Note ▪*** *This is only a path name and does not include the file name itself.*
`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.

Table 9-98 ▪ Language Resource Properties (cont.)

Property	Definition
`human.readable.language.name`	The name of the language represented by the data in this resource file (for example, English, Espanol, and so on).
`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 9-99 ▪ 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 Log Settings area in the General Settings view on the Project page, and from the Output Debug Information action—see* *Reviewing Debug Information*.

Product Registry

The product registry is essentially a product configuration database that keeps track of features and components of products and accomplishes tasks such as associating file name extensions with applications. Proper use of product codes and versions are essential to the operation of the registry in the future—for example, the use of the Find Component in InstallAnywhere Registry action.

InstallAnywhere finds the locations of components in the registry by checking the product code. To make reliable use of the Find Component in InstallAnywhere Registry action, you must enter the product code and version correctly.

The product registry stores the information that is configured in the General Settings view on the Project page. The location of the registry depends on the platform to which you are installing.

Table 9-100 ▪ 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.
OS or 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 is not required. Keeping a common vendor ID enables you to identify all of your organization's products that are 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 in General Settings view on the Project page.*

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 9-101 ▪ 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 in **Send stderr to** and **Send stdout to** settings (Project page> General Settings view > Log Settings area). It includes: • 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 view), 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
```



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 Tab.*

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 and uninstaller. The record of these responses can be used to control subsequent installer and uninstaller executions. Typically, response files use the default name `installer.properties` and provide settings to support an installer and uninstaller running in silent mode. Response files are essentially just text files that can provide settings for an installer and uninstaller.

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 Yes in the **Always Generate Response File** setting (Project page > General Settings view > Project Information area).

  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.

  ***Note** ▪ Only the installer's response files will be created using the **Always Generate Response File** option.*

- **Using the -r command-line switch**—You can also generate a response file when running an installation and uninstallation 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"
  uninstall.exe -r "/Users/MyName/myresponse.properties"
  ```

  InstallAnywhere further enables you to make a response file for the installer under the following circumstances:

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

  ***Note** ▪ For the uninstaller's response file creation, both a path and a file name should be specified for the response file while using the `-r` command-line switch.*



***Important** ▪ A response file must be saved with the appropriate encoding.*

For Windows-based target systems, the response file must be saved in one of the following encodings:

- *UTF-8 without a BOM*
- *UTF-16 little endian*

For Linux-based and OS or OS X–based target systems, the response file must be UTF-8 without a BOM.

If an unsupported encoding is used, the installer is unable to read the file properly.

The silent installs and uninstalls on only support the -i silent parameters and response files, not passing InstallAnywhere variables, such as $USER_INSTALL_DIR$ using the -DUSER_INSTALL_DIR=/Applications/TestFolder syntax.

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.

Customize Directory for Response File

You can customize the directory in which the response file is generated during installation by setting the *$IA_CUSTOM_RESPONSE_FILE_PATH$* standard InstallAnywhere variable. To set this variable, use the **Set InstallAnywhere Variable – Single Variable** or **Set InstallAnywhere Variable – Multiple Variables** action within the **Pre-Install**, **Install**, or **Post-Install** views on the **Sequence** page.

Choose Install Sets Variables

The Choose Install Set panel/console variables are recorded in response files. Here is an example:

```
#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 view on the Sequence page 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, pas the `-p` argument with the path and name of the build properties file to `build.exe`:

`build.exe Project_File_Path -p PathToBuildProperties.xml`

For example:

```
build.exe C:\MySetups\MyProduct.iap_xml -p C:\Path\BuildPropeties.xml
```

If you do not provide an absolute path to your build properties file, the builder looks for it in the same directory as your project.



Important ▪ *The `BuildProperties.xml` file supports the specification of multiple build configurations. The build configurations that are specified in this file override the build configurations that are defined in the project.*

InstallAnywhere includes a build properties file template named `BuildProperties.xml`:

IA_HOME`/resource/build/BuildProperties.xml`

This template file provides a sample of all possible build settings; you can use it as a template to meet your build requirements.

Ant Properties

InstallAnywhere supports the use of Ant properties 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.*

Supported Properties in the BuildProperties.xml File

Many of the settings that you can configure 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.

Project Page > General Settings View Settings

The following table correlates settings in the General Settings view on the Project page with settings in the `BuildProperties.xml` file.

Table 9-102 ▪ Project Page > General Settings View Settings

Option	BuildProperties.xml File Setting
Send stderr to	`InstallerStdErrRedirect="<path_to_file>"`
Send stdout to	`InstallerStdOutRedirect="<path_to_file>"`

Project Page > Platforms View Settings

The following table correlates settings in the Platforms view on the Project page with settings in the `BuildProperties.xml` file.

Table 9-103 ▪ Project Page > Platform View Settings

Platform	Setting	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>"`
Windows, pure 64-bit	Default Windows UI Mode > Installer UI Mode	`Windows64DefaultUI="<Silent/GUI/Console>"`

Project Page > JVM Settings View Settings

The following table correlates settings in the JVM Settings view on the Project page with settings in the `BuildProperties.xml` file.

Table 9-104 ▪ Project Page > JVM Settings View Settings

Option	BuildProperties.xml File Setting
Valid VM list	`InstallerValidVMList="<11.0+, 17.0*>"`
Minimum Heap Size	`InstallerInitialHeapSize="<16777216>"`
Maximum Heap Size	`InstallerMaxHeapSize="<50331648>"`

Installer UI Page > Look & Feel View Settings

The following table correlates an setting in the List of Labels for Installer Steps setting in the Look & Feel view on the Installer UI page with a setting in the `BuildProperties.xml` file:

Table 9-105 ▪ Installer UI Page > Look & Feel View Settings

Option	BuildProperties.xml File Setting
Auto populate labels when saving	`AutoPopulateLabels="<true/false>"`

Organization Page > Components View Settings

The following table correlates an setting in the Components view on the Organization page of the Advanced Designer with a setting in the `BuildProperties.xml` file:

Table 9-106 ▪ Organization Page > Components View Settings

Option	BuildProperties.xml File Setting
Auto-clean when building	`AutoCleanComponents="<true/false>"`

Build Page > Build Installers View Settings

Build Configurations Tab

The following table correlates the settings on the Build Configurations tab in the Build Installers view on the Build page of the Advanced Designer with settings in a `BuildProperties.xml` file:

Table 9-107 ▪ Build Page > Build Installers View > Build Configurations Tab Settings

Option	BuildProperties.xml File Setting

Select Build Configuration	Create a `<configuration>` element for each build configuration that you want to build, using the following syntax:
	`<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/osx`
	`  /java/unix/windows"`
	`  buildWithVM="true/false"`
	`  buildWithNoVM="true/false"`
	`  outputDir="myOutputDir"`
	`  bundledVM="path_to_file"`
	`  />`
	`</configuration>`
	Regarding each `<target>` element, note the following:
	• Each `<target>` element appends one new target to those already in the build configuration.
	• There can be multiple `<target>` elements.
	• The `outputDir` and `bundledVM` attributes are optional.
	• Each `<target>` element must include a `buildWithVM` attribute, a `buildWithNoVM` attribute, or both.
	
	Important ▪ *The following subelements of the* `<configuration>` *element can also be specified outside of a* `<configuration>` *element. However, those elements defined within a* `<configuration>` *element override those defined elsewhere.*

Build Output Location	`BuildOutputLocation="<path_to_directory>"`
[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.*

Build Targets Subtab

The following table correlates the settings on the Build Targets subtab of the Build Configurations tab (Build page > Build Installers view) of the Advanced Designer with settings in a `BuildProperties.xml` file:

Table 9-108 ▪ Build Page > Build Installers View > Build Configurations Tab > Build Targets Subtab Settings

Option	BuildProperties.xml File Setting

OS X	`BuildOSX="<true/false>"`
	`WantAuthenticationOSX="<true/false>"`
	`WantAuthenticationOSXShowGUI="<true/false>"`

Windows	`BuildWindowsWithVM="<true/false>"`
	`BuildWindowsWithoutVM="<true/false>"`
	`WindowsVMpackLocation="<path_to_file>"`
	`WindowsConsoleLauncher="<true/false>"`

Windows_Pure_64_Bit	`BuildWindows64WithVM="<true/false>"`
	`BuildWindows64WithoutVM="<true/false>"`
	`Windows64VMpackLocation="<path_to_file>"`

AIX	`BuildAIXWithVM="<true/false>"`
	`BuildAIXWithoutVM="<true/false>"`
	`AIXVMpackLocation="<path_to_file>"`

HP-UX	`BuildHPUXWithVM="<true/false>"-`
	`BuildHPUXWithoutVM="<true/false>"`
	`HPUXVMpackLocation="<path_to_file>"`

Linux	`BuildLinuxWithVM="<true/false>"`
	`BuildLinuxWithoutVM="<true/false>"`
	`LinuxVMpackLocation="<path_to_file>"`

Solaris	`BuildSolarisWithVM="<true/false>"`
	`BuildSolarisWithoutVM="<true/false>"-`
	`SolarisVMpackLocation="<path_to_file>"`

Unix (All)	`BuildUnixAll="<true/false>"`

UNIX_with_VM	`BuildNamedUnixWithVM="<true/false>"`
	`BuildNamedUnixWithoutVM="<true/false>"`
	`NamedUnixVMpackLocation="<path_to_file>"`
	`NamedUnixTitle="<name_of_unix>"`

Other Java-Enabled Platforms	`BuildPureJava="<true/false>"`
[Not available in the Advanced Designer]	`OverrideAllPlatformSettings="<true/false>"`

Distribution Subtab

The following table correlates the settings on the Distribution subtab of the Build Configurations tab (Build page > Build Installers view) of the Advanced Designer with settings in a `BuildProperties.xml` file:

Table 9-109 ▪ Build Page > Build Installers View > Build Configurations Tab > Distribution Subtab Settings

Option	BuildProperties.xml File Setting
Build Web Installers	`BuildWebInstaller="<true/false>"`
Optimize [Web] Installer Size by Platform and Tags	`OptimizeWebInstaller="<true/false>"`
Bundle JRE as Directory	`IncludeJreAsDirWeb="<true/false>"`

Web page displays in	This option is set within a `<webInstaller>` element inside of a build configuration element, `<configuration>`:
	`<configuration name="<name_of_build_configuration>">`
	`  <webInstaller>`
	`  <language><en/ja></language>`
	`  </webInstaller>`
	`  …`
	`</configuration>`
	
	Note ▪ *See Select Build Configuration for more information.*

Build CD-ROM installers	`BuildCDROMInstaller="<true/false>"`
Optimize [CD-ROM] Installer Size by Platform and Tags	`OptimizeCDROMInstaller="<true/false>"`
Build Merge Module Template	`BuildMergeModule="<true/false>"`
Optimize Merge Module/ Template Size by Platform and Tags	`OptimizeMergeModule="<true/false>"`
Read Only	`BuildReadOnlyMergeModule="<true/false>"`

buildproperties.properties File

You can use a `buildproperties.properties` file to build installers for a project. To do this, pass the `-p` argument with the path and name of the `buildproperties.properties` file to `build.exe`:

```
build.exe C:\MySetups\MyProduct.iap_xml -p C:\Path\buildproperties.properties
```

The settings that are specified in the .properties file override the build settings in the project.

InstallAnywhere includes a sample `buildproperties.properties` file:

IA_HOME/resource/build/buildproperties.properties

Supported Properties in the buildproperties.properties File

Many of the settings that you can configure 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.

Project Page > General Settings View Settings

The following table correlates settings in the General Settings view on the Project page with settings in the buildproperties.properties file.

Table 9-110 ▪ Project Page > General Settings View Settings

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 Page > Platforms View Settings

The following table correlates settings in the Platforms view on the Project page with settings in the buildproperties.properties file.

Table 9-111 ▪ Project Page > Platforms View Settings

Platform	Option	buildproperties.properties File Setting
UNIX	Default UNIX Installer UI Mode > Installer UI Mode	default.ui.mode.unix=GUI/Silent/Console
Windows	Default Windows UI Mode > Installer UI Mode	default.ui.mode.windows=GUI/Silent/Console
Windows, pure 64-bit	Default Windows UI Mode > Installer UI Mode	default.ui.mode.windows64=GUI/Silent/Console

Project Page > JVM Settings View Settings

The following table correlates settings in the JVM Settings view on the Project page with settings in the buildproperties.properties file.

Table 9-112 ▪ Project Page > JVM Settings View Settings

Option	buildproperties.properties File Setting
Valid VM list	com.zerog.ia.installer.options.valid.vm.list=11.0.*, 17.0+

Table 9-112 ▪ Project Page > JVM Settings View Settings (cont.)

Option	buildproperties.properties File Setting
Minimum Heap Size	`com.zerog.ia.installer.options.heap.size.initial=16777216`
Maximum Heap Size	`com.zerog.ia.installer.options.heap.size.max=50331648`

Installer UI Page > Look & Feel View Settings

The following table correlates a setting in the Look & Feel view on the Installer UI page with a setting in the `buildproperties.properties` file:

Table 9-113 ▪ Installer UI Page > Look & Feel View Settings

Option	buildproperties.properties File Setting
Auto populate labels when saving	`com.zerog.ia.build.options.auto-populate.labels=true/false`

Organization Page > Components View Settings

The following table correlates a setting in the Components view on the Organization page with a setting in the `buildproperties.properties` file:

Table 9-114 ▪ Organization Page > Components View Settings

Option	buildproperties.properties File Setting
Auto-clean when building	`com.zerog.ia.build.options.auto-claen.components=true/false`

Build Page > Build Installers View Settings

Build Configurations Tab

The following table correlates the settings on the Build Configurations tab in the Build Installers view on the Build page of the Advanced Designer with settings in a `buildproperties.properties` file:

Table 9-115 ▪ Build Page > Build Installers View > Build Configurations Tab Settings

Option	buildproperties.properties File Setting

Select Build Configuration	`number.of.configs=2`
	`config.1.name=c2`
	`config.2.name=c1`
	
	In the `buildproperties.properties` file, the build target, distribution, and locale settings include a number to identify its associated build configuration, such as:
	
	`config.1.com.zerog.ia.build.platform.windows.novm=false`
	
	For more information, see Build Targets Subtab, Distribution Subtab, and Locales Subtab.

Table 9-115 ▪ Build Page > Build Installers View > Build Configurations Tab Settings (cont.)

Option	buildproperties.properties File Setting
Build Output Location	`com.zerog.ia.build.options.output.location=path_to_directory`
[Working Directory]	`com.zerog.ia.build.options.workdir.location=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.*

Build Targets Subtab

The following table correlates the settings on the Build Targets subtab of the Build Configurations tab (Build page > Build Installers view) of the Advanced Designer with settings in a `buildproperties.properties` file:

Table 9-116 ▪ Build Page > Build Installers View > Build Configurations Tab > Build Targets Subtab Settings

Option	buildproperties.properties File Setting
OS X	`config.1.com.zerog.ia.build.platform.osx.novm=true/false`

Windows	`config.1.com.zerog.ia.build.platform.windows.novm=true/false`
	`config.1.com.zerog.ia.build.platform.windows.vm=true/false`
	`config.1.com.zerog.ia.build.platform.windows.use.console.laucher=true/false`
	`config.1.com.zerog.ia.build.vmpack.windows.path=path`

Windows_Pure_64_Bit	`config.1.com.zerog.ia.build.platform.windows64.novm=true/false`
	`config.1.com.zerog.ia.build.platform.windows64.vm=true/false`
	`config.1.com.zerog.ia.build.vmpack.windows64.path=path`

AIX	`config.1.com.zerog.ia.build.platform.aix.novm=true/false`
	`config.1.com.zerog.ia.build.platform.aix.vm=true/false`
	`config.1.com.zerog.ia.build.vmpack.aix.path=path`

HP-UX	`config.1.com.zerog.ia.build.platform.hpux.novm=true/false`
	`config.1.com.zerog.ia.build.platform.hpux.vm=true/false`
	`config.1.com.zerog.ia.build.vmpack.hpux.path=path`

Linux	`config.1.com.zerog.ia.build.platform.linux.novm=true/false`
	`config.1.com.zerog.ia.build.platform.linux.vm=true/false`
	`config.1.com.zerog.ia.build.vmpack.linux.path=path`

Solaris	`config.1.com.zerog.ia.build.platform.solaris.novm=true/false`
	`config.1.com.zerog.ia.build.platform.solaris.vm=true/false`
	`config.1.com.zerog.ia.build.vmpack.solaris.path=path`

Table 9-116 ▪ Build Page > Build Installers View > Build Configurations Tab > Build Targets Subtab Settings

Option	buildproperties.properties File Setting

UNIX_with_VM	`config.1.com.zerog.ia.build.platform.named_unix.vm=true/false`
	`config.1.com.zerog.ia.build.platform.named_unix.novm=true/false`
	`config.1.com.zerog.ia.build.platform.unix.novm=true/false`
	`config.1.com.zerog.ia.build.vmpack.unix.path=path`

Other Java-Enabled Platforms	`config.1.com.zerog.ia.build.platform.java.novm=true/false`
[Not available in the Advanced Designer]	`com.zerog.ia.build.options.ignoreAllPlatformSettings=true/false`

Distribution Subtab

The following table correlates the settings on the Distribution subtab of the Build Configurations tab (Build page > Build Installers view) of the Advanced Designer with settings in a `buildproperties.properties` file:

Table 9-117 ▪ Build Page > Build Installers View > Build Configurations Tab > Distribution Subtab Settings

Option	buildproperties.properties File Setting
Build Web Installers	`config.1.com.zerog.ia.build.options.output.web=true/false`
Optimize [Web] Installer Size by Platform and Tags	`config.1.com.zerog.ia.build.options.optimization.platform.web=true/false`
Bundle JRE as Directory	`config.1.com.zerog.ia.build.options.includejreasdir.web=true/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/false`
Optimize [CD-ROM] Installer Size by Platform and Tags	`config.1.com.zerog.ia.build.options.optimization.platform.cdrom=true/false`
Build Merge Module Template	`config.1.com.zerog.ia.build.options.output.merge=true/false`
Optimize Merge Module/ Template Size by Platform and Tags	`config.1.com.zerog.ia.build.options.optimization.platform.merge=true/false`
Read Only	`config.1.com.zerog.ia.build.options.output.merge.read.only=true/false`

Locales Subtab

The following table correlates the settings on the Locales subtab of the Build Configurations tab (Build page > Build Installers view) of the Advanced Designer with settings in a `buildproperties.properties` file:

Table 9-118 ▪ 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`

Command-Line Reference

InstallAnywhere's build program, installers and uninstallers, and launchers accept command-line arguments that dictate, to some extent, how those programs run.

Table 9-119 ▪ InstallAnywhere Command-Line Reference

Section	Description
Build Command-Line Arguments	Describes the build and platform arguments for the command-line build.
Installer and Uninstaller Command-Line Arguments	Describes the command-line arguments for the installers and uninstallers that InstallAnywhere builds.
Maintenance Mode Command-Line Arguments	Describes the command-line arguments for running maintenance mode.
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.

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.

Information about `build.exe` command-line arguments is organized into the following sections:

- Build-Related Arguments
- Platform-Related Arguments
- Distribution-Related Arguments
- License-Related Arguments



Note ▪ *For Windows-based development systems, InstallAnywhere provides two versions of the command-line build tool:* `build.exe` *and* `build-as-invoker.exe`*. Using* `build-as-invoker.exe` *is recommended for users who do not have administrative privileges on their Windows-based build system.*

Build-Related Arguments

The command-line build supports the following build-related arguments.

Table 9-120 ▪ Build-Related Command-Line Arguments

Option	Description
+	Add platform to build. (See Platform Arguments, below.) For example, to build for Windows, no VM, use: `build.exe C:\my_setups\MyProduct.iap_xml +W` Using this option activates all build targets in your InstallAnywhere project for the specified platform/VM combination.  ***Note*** ▪ *You can use the + and - options for the entire project (without specifying the -all option), as well as for platform targets and distribution options for individual configurations.*
-	Remove platform from build. (See Platform Arguments, below.) For example, to suppress the build of OS or OS X, use: `build.exe C:\my_setups\MyProduct.iap_xml -X` Using this option deactivates all build targets in your InstallAnywhere project for the specified platform/VM combination.  ***Note*** ▪ *You can use the + and - options for the entire project (without specifying the -all option), as well as for platform targets and distribution options for individual configurations.*
-all	To build all of a project's build configurations, even those that do not have the **Add to project build** check box selected, use the `-all` option. Using the `build.exe` command without this `-all` option builds only the build configurations that have been added to the project (those that have the **Add to project build** check box selected).
-applianceMode	Build virtual appliances along with your installers as part of the build. `build <IA_Project_File> -applianceMode [<Appliance_Configuration_Name>] [-all]`

Table 9-120 ▪ Build-Related Command-Line Arguments (cont.)

Option	Description
-d	Use the specified working directory (relative or full path) for the build. This is the directory that `build.exe` uses to store all temporary files: the build log, locales directory, and others. `build.exe MyProduct.iap_xml -d ..\mydir` The default value for the working directory is the InstallAnywhere project file directory. Relative paths are relative to the location of the project file.  ***Tip ▪*** *The same result can be achieved by passing a build properties file with the working directory set by a BuildWorkdirLocation setting.*
-encrypt	Generate an encrypted hexadecimal value that can be directly copied and pasted into a response file as an encrypted variable value. `build -encrypt <product-code> <text-to-encrypt>` The resulting encrypted hexadecimal value can then be used later. For example, you could update the parent response file with an encrypted value generated from a password without needing to first run the entire installation to only generate this value.
-?	Display help information on the `build.exe` command options.
-i	Use the 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: `build.exe MyProduct.iap_xml -i 172.17.1.110/PN113/PASS123`  ***Note ▪*** *Address, user ID, and password values are separated by a slash character (/).*
-p	As an alternative to specifying arguments to the build executable file on the command line, you can store the settings in a build properties file, and use the `-p` switch to point to the desired build-properties file. For example: `build.exe SampleApp.iap_xml -p BuildProperties.xml`
-v	Print the InstallAnywhere product version. `build.exe -v`

Table 9-120 ▪ Build-Related Command-Line Arguments (cont.)

Option	Description
productVersion=*[version]*	Use to set the version of an InstallAnywhere project during build. For example, to set the version `1.0.0.0`, use: `build.exe C:\my_setups\MyProduct.iap_xml productVersion=1.0.0.0`  ***Note ▪*** *Note the following while using this command-line argument:* • *Version should be mentioned without square brackets.* • *This command-line argument should be used only after the project's specification.*
-btv *[path_of_file]*	Use to specify a build time variables properties file.
-unregister	Use to delete licensing information and to unregister InstallAnywhere.



Note ▪ *You can use several different methods to set the project version at build time. For more information, see* *Setting Project Version at Build Time*.

Platform-Related Arguments

The following arguments—in conjunction with the plus (+) and minus (-) parameters—add and remove platform-related options:

Table 9-121 ▪ Platform-Related 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
l, L	Linux without VM option
lv, LV	Linux with VM option
s, S	Solaris without VM option
sv, SV	Solaris with VM option

Table 9-121 ▪ Platform-Related Arguments (cont.)

Argument	Description
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
wc, WC	Windows without VM option and with console launcher option
wg, WG	Windows without VM option and with graphical launcher option
wvc, WVC	Windows with VM option and with console launcher option
wvg, WVG	Windows with VM option and with graphical launcher option
w64, W64	Pure 64-bit Windows without VM option
w64v, W64V	Pure 64-bit Windows with VM option
w64c, W64C	Pure 64-bit Windows without VM option and with console launcher option
w64g, W64G	Pure 64-bit Windows without VM option and with graphical launcher option
w64vc, W64VC	Pure 64-bit Windows with VM option and with console launcher option
w64vg, W64VG	Pure 64-bit Windows with VM option and with graphical launcher option
x, X	Apple OS or OS X without VM option
x7, X7	Apple OS or OS X without VM and search for Oracle JRE 7 option
xv, XV	Apple OS or OS X with VM option

Distribution-Related Arguments

The following arguments—in conjunction with the plus (+) and minus (-) parameters—add and remove distribution-related options:

Table 9-122 ▪ Distribution-Related Arguments

Argument	Description
cd	Build CD-ROM installers

Table 9-122 ▪ Distribution-Related Arguments (cont.)

Argument	Description
merge	Build Merge modules
opt	Optimize by platform
web	Build Web installers

License-Related Arguments

The command-line build supports the following license-related arguments.

Table 9-123 ▪ License-Related Command-Line Arguments

Option	Description
-generateHostID	Obtain the host ID for the machine. The host ID is used for configuring the license for InstallAnywhere.  ***Note ▪*** *If the machine is not connected to a network, the* `-generateHostID` *argument fails to obtain the host ID. For more information and an alternative method for obtaining the Host ID, see Using build.exe to Build Installers from the Command Line.*
-ls	Connect the command-line build to the specified license server. This option is available for users who purchase concurrent licenses of InstallAnywhere. Provide the location (IP address or host name) and port (optional) for the license server. For example, if the license server was operating on port 8888 of localhost, you might use: `build.exe -ls 127.0.0.1:8888`  ***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.*
-registerNodeLocked ***[path_of_license_file]***	Use to specify a node-locked license file.

Installer and Uninstaller Command-Line Arguments

InstallAnywhere installers and uninstallers can be run using the following command-line arguments.

Table 9-124 ▪ Command-Line Arguments for InstallAnywhere Installers and Uninstallers

Argument	Description

-i	Sets the interface mode (`silent/console/gui`) for installer and uninstaller as shown below:
	`c:\myinstall.exe -i silent`
	`c:\myuninstall.exe -i silent`
	
	Note ▪ *Using `-i` command-line switch without an argument (interface mode) or with an invalid argument (interface mode) for launching the Installer, results to display a usage message including the appropriate available options on 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 ▪*** *Consider the following while using the command-line argument:* • *This path can be absolute or relative. (Relative paths are relative to the location of the installer).* • *Uninstaller does not support the command-line argument for setting location of a response file.*
-r	Creates a response file for the installer and uninstaller. (See Generating Response Files) At the command line, enter `<installer_name/uninstaller_name> -r "path_and_file_name"` For example, to create response file for the installer: `c:\myinstall.exe -r c:\temp\myinstaller.properties` 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 installer's response file, the file will be named `installer.properties` or `[installername].properties` and it will be created in the same directory as the installer.  ***Note ▪*** *Response files can be used to provide input for both silent installers and uninstallers.*

Table 9-124 ▪ Command-Line Arguments for InstallAnywhere Installers and Uninstallers (cont.)

Argument	Description
-D	Sets or modifies variables. `c:\myinstall.exe -Dmyvar=myvalue`  ***Note ▪*** *Uninstaller does not support the command-line argument for setting or modifying variables.*

-l	Uses the specified language code (and optional country code) to set the locale for the installer. (See Language Codes.)
	`c:\myinstall.exe -l en`
	`sh ./install.bin -l pt_BR`
	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.
	The optional country code is a two-character (commonly uppercase) code defined by the ISO-3166 standard.
	
	Note ▪ *Consider the following while using the command-line argument:*
	● *Locale options are only respected if the installer includes localizations for the locale you specify.*
	● *Uninstaller does not support the command-line argument for setting locale and always run with locale set for installer.*

-jvmxms	Sets the JVM heap size initial value.
	`<installer_name> -jvmxms <size>`
	The default size for these values is measured in bytes. Append the letter **`k`** or **`K`** to the value to indicate kilobytes, **`m`** or **`M`** to indicate megabytes, and **`g`** or **`G`** to indicate gigabytes. For example, to set the initial JVM heap size to 25 megabytes, enter the following:
	`install.exe -jvmxms 25m`
	
	Note ▪ *Uninstaller does not support the command-line argument for setting initial value of JVM heap size.*

Table 9-124 ▪ Command-Line Arguments for InstallAnywhere Installers and Uninstallers (cont.)

Argument	Description

-jvmxmx	Sets the JVM heap size maximum value.
	
	`<installer_name> -jvmxmx <size>`
	
	The default size for these values is measured in bytes. Append the letter **k** or **K** to the value to indicate kilobytes, **m** or **M** to indicate megabytes, and **g** or **G** to indicate gigabytes. For example, to set the maximum JVM heap size to 50 megabytes, enter the following:
	
	`install.exe -jvmxms 50m`
	
	
	
	Note ▪ *Uninstaller does not support the command-line argument for setting maximum value of JVM heap size.*

-? **-help**	Shows help for the InstallAnywhere installer.  ***Note ▪*** *Consider the following while using the command-line argument:* - *On Windows,* `-help` *works only from the console launcher. Make sure to set the LaunchAnywhere to Console in the Windows area in the Platforms view on the Project page. (For an installed LaunchAnywhere to provide this information, you need to make sure it is explicitly set to Console Launcher on the action.)* - *Uninstaller does not support the command-line argument for showing uninstaller help.*
-tempdir	Sets the temp directory path. At the command line, enter `<installer_name/uninstaller_name> -tempdir "Path"` You can set a required temp directory path by utilizing the `-tempdir` argument with the installer or uninstaller name as shown above. For instance, to set the temp directory path to `C:\ExampleTemp` for the installer, where the InstallAnywhere launchers are required to be extracted, enter the following at the command line: `install.exe -tempdir "C:\ExampleTemp"`  ***Note ▪*** *If the specified directory path is not accessible or is not a valid path, then the default temp directory path will be utilized by the InstallAnywhere installer/uninstaller.*

Table 9-124 ▪ Command-Line Arguments for InstallAnywhere Installers and Uninstallers (cont.)

Argument	Description
-Djvm.securefolder.check	To enable or disable the validation of the Java Virtual Machine (JVM) secure folder during installation or uninstallation. At the command line, enter `<installer/uninstaller_name> -Djvm.securefolder.check=<option>` where `<option>`: ● **true**—Enables Java Virtual Machine (JVM) secure folder validation, allowing the InstallAnywhere launcher to verify whether the JVM is located in a secure or administrative folder. By default, this validation is enabled on the Windows platform when running the installer or uninstaller with administrative privileges. ● **false**—Disables Java Virtual Machine (JVM) secure folder validation, allowing the installer or uninstaller to proceed with installation or uninstallation, respectively, even if the Java Virtual Machine (JVM) is located in an unsecured folder. When this validation is enabled, the launcher allows the installer or uninstaller to run only if the JVM is located in a secure or administrative folder. JVMs found in unsecured folders are blocked, and the launcher terminates execution with a security-related error message before the installer or uninstaller starts. For example, to disable Java Virtual Machine (JVM) secure folder validation during installation, you can enter the following at the command line: `install.exe -Djvm.securefolder.check=false`

Maintenance Mode Command-Line Arguments

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 9-125 ▪ 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`

Table 9-125 ▪ Maintenance Mode Command-Line Options (cont.)

Option	Description
`-repair`	Use to repair an installed product. `Change MyProduct Installation.exe -repair`

If no options are specified, 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.*

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*

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.nl.current.vm`.

Syntax

`LaunchAnywhereName` `LAX_VM "`*`full_path_to_java_executable`*`"`

Example

```
OfficeSuite LAX_VM "C:\Program Files\Java\jdk21.0.2_13\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

InstallAnywhere includes an Ant task to build installers from Ant. The InstallAnywhere Ant task (`iaant.jar`) is located in your InstallAnywhere application folder:

`IA_HOME``\resource\build\iaant.jar`



Tip ▪ *To integrate the InstallAnywhere Ant task in an Ant project, 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 later.*

The following Ant task parameters are available:

- Build Parameters
- Platform Options
- Build Options
- Installer Options

Build Parameters

Use the following parameters to control the build process for the InstallAnywhere Ant task.

Table 9-126 ▪ Build Parameters

Attribute	Description
IAProjectFile	The location of the InstallAnywhere project that you want to build. This is a required parameter.
IALocation	The location where InstallAnywhere is installed. This is an optional parameter.  ***Note*** ▪ *If IALocation is not specified, the task searches for a copy of InstallAnywhere to run against. If InstallAnywhere is not installed in one of the default locations, 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`
	For example:
	`<buildinstaller IAProjectFile="Project_File" IALocation="IA_HOME" i5OSLogin="hostname_or_ip/userName/password">`
	After the project successfully builds, an asterisk-masked `i5OSLogin` parameter (password portion) prints to the console.
	This is an optional parameter.

Table 9-126 ▪ Build Parameters (cont.)

Attribute	Description
propertiesfile	The location of a `BuildProperties.xml` file. If this parameter is used, all other attributes are ignored. This is an optional parameter.
failOnError	Stop the build process if the command exits with a return code other than 0. Defaults to false. This is an optional parameter.

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. 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 9-127 ▪ Platform Options

Parameter	Description
BuildLinuxWithVM	True/False
BuildLinuxWithoutVM	True/False
LinuxVMPackLocation	Path
BuildHPUXWithVM	True/False
BuildHPUXWithoutVM	True/False
HPUXVMPackLocation	Path
BuildAIXWithVM	True/False
BuildAIXWithoutVM	True/False
AIXVMPackLocation	Path
BuildSolarisWithVM	True/False
BuildSolarisWithoutVM	True/False

Table 9-127 ▪ Platform Options (cont.)

Parameter	Description
SolarisVMPackLocation	Path
BuildNamedUNIXWithVM	True/False
BuildNamedUNIXWithoutVM	True/False
NamedUNIXVMPackLocation	Path
NamedUNIXTitle	String
BuildWindowsWithVM	True/False
BuildWindowsWithoutVM	True/False
WindowsVMPackLocation	Path
BuildWindows64WithVM	True/False
BuildWindows64WithoutVM	True/False
Windows64VMPackLocation	Path
BuildUNIXAll	True/False  ***Note ▪*** *This parameter corresponds with Unix (All)/Generic Unix build target in the Build Installers view on the Build page of the Advanced Designer.*
BuildOSX	True/False
WantAuthenticationOSX	True/False
WantAuthenticationOSXShowGUI	True/False
BuildPureJava	True/False

Table 9-127 ▪ Platform Options (cont.)

Parameter	Description
OverrideAllPlatformSettings	True/False  ***Note ▪*** *When you set OverrideAllPlatformSettings 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 OS X target (BuildOSX="true"), the Ant task builds installers for Windows, Linux, and OS or OS X. However, if you also set OverrideAllPlatformSettings to True, Ant builds an installer for OS or OS X only.*

Build Options

Build options specify build distribution settings.

Table 9-128 ▪ Build Options

Parameter	Description
BuildCDROMInstaller	True/False
BuildWebInstaller	True/False
BuildMergeModule	True/False
BuildReadOnlyMergeModule	True/False
BuildWorkdirLocation	Path  ***Note ▪*** `BuildWorkdirLocation` *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* `BuildWorkdirLocation` *and* `BuildOutputLocation` *options, the* `BuildOutputLocation` *takes precedence over the working directory for build output.*
BuildOutputLocation	Path
OptimizeCDROMInstaller	True/False
OptimizeWebInstaller	True/False

Table 9-128 ▪ Build Options (cont.)

Parameter	Description
OptimizeMergeModule	True/False
AutoPopulateLabels	True/False
AutoCleanComponents	True/False

Installer Options

Installer options affect the behavior of the installers that the project builds.

Table 9-129 ▪ Installer Options

Parameter	Description
InstallerStdErrRedirect	Path
InstallerStdOutRedirect	Path
InstallerValidVMList	String
InstallerInitialHeapSize	Int
InstallerMaxHeapSize	Int
UNIXDefaultUI	Silent/Console/GUI
WindowsDefaultUI	Silent/Console/GUI
Windows64DefaultUI	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/.*

Build Configurations

Build configurations support InstallAnywhere projects that include multiple build targets for the same platform but use different VMs. Each target in a build configuration identifies a build target to add to those that already exist in the InstallAnywhere project file. Ant always builds the targets that you define in the InstallAnywhere Ant task's Configuration section.

Table 9-130 ▪ Configuration Elements and Attributes

Tag	Description
configuration	Contains targets that append additional build targets to those already defined in the InstallAnywhere project file. **Attributes** The following are attributes of the `configuration` element: • **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	Contains targets that append additional build targets to those already defined in the InstallAnywhere project file. **Nested Element(s)** The following are nested element(s) of the `locales` element: • **localeSuffix**—Set to `en/ja/de/fr/ar`... any of 31 available run-time locales.

Table 9-130 ▪ Configuration Elements and Attributes (cont.)

Tag	Description
target	Contains the settings to define a single build target. Targets must include a platform setting. **Attributes of Target Element** The following are attributes of the target element: ● **platform**—Specifies the target platform. Available values are `windows`, `Windows_Pure_64_Bit`, `linux`, `osx`, `solaris`, `aix`, `hpux`, `unix`, `unixwithvm`, and `java`. ● **buildWithNoVM**—Indicates whether this target includes a bundled VM. Use `true` to build an installer without a bundled VM; otherwise, use `false`. ● **buildWithVM**—Indicates whether 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.

Custom Code APIs

InstallAnywhere includes several APIs that enable you to extend its functionality and automate most IDE and GUI tasks.

Table 9-131 ▪ 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.  ***Note ▪*** *The General API javadocs are found by opening the* `\javadoc\com\zerog\ia\api\pub\package-summary.html` *page in the InstallAnywhere installation directory.*

Table 9-131 ▪ InstallAnywhere APIs (cont.)

API	Description
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.  ***Note ▪*** *The Product Registry API javadocs are found by opening the* `\javadoc\com\zerog\ia\api\pub\registry\package-summary.html` *page in the InstallAnywhere installation directory.*
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.  ***Note ▪*** *The Services API javadocs are found by opening the* `\javadoc\index.html` *page in the InstallAnywhere installation directory.*
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. For more information on using the Project Automation APIs, see the following: ● Project Automation API Code Samples ● Project Automation APIs for Maintenance Mode and Instance Management ● Project Automation javadocs, which are found by opening the `\project-auto\javadocs\index.html` page in the InstallAnywhere installation directory.

Table 9-131 ▪ InstallAnywhere APIs (cont.)

API	Description
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 ▪*** *The Test Automation API javadocs can be found by opening the* `\gui-test-auto\javadocs\index.html` *page in the InstallAnywhere installation directory. The readme.txt file can be found in the* `\gui-test-auto` *directory of the InstallAnywhere installation directory.*

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 Code Samples

Note ▪ *If you are using a node-locked license of InstallAnywhere, the licensing libraries must be loaded while using the project automation API. This is typically done by configuring an IDE setting or by passing the following parameters through the command line when running the Java 11:*

```
--add-opens java.base/jdk.internal.loader=ALL-UNNAMED -
Djava.library.path=IA_HOME\resource\fnp\libraries
```

If the library path includes spaces, ensure that you enclose the library path within quotes:

```
-Djava.library.path="IA_HOME\resource\fnp\libraries"
```

InstallAnywhere includes project automation API code samples that you can use to edit your project files programmatically in Java. These code samples are installed in the following directory:

InstallAnywhere Installation Directory`\project-auto\sample\src\com\ia\projauto\samples`

A `readme.txt` file is provided in the `project-auto\sample` directory that contains the following information:

- **How to Use**—Explains how to run these code samples using classpath settings.
- **Sample Code Configuration**—Explains how to run these code samples by making use of Ant scripts that compile and run the API code samples by using an `automationbuild.properties` file.

The following project API code samples are available:

Table 9-132 ▪ Project API Code Samples

File Name	Description
BuildConfig_Test.java	Configure build target settings (Build page > Build Installers view > Build Configurations tab > Build Targets subtab).
CreateNewBasicProject_Test.java	Create a new project.
CreateNewProjectFromTemplate_Test.java	Create a new project from an existing template.
InstallerUIConfig_Test.java	Configure settings on the Installer UI page of the project.
InstallTask_Test.java	Configure the Install sequence.
InstanceManagement_Test.java	Configure settings in the Instance Management area (Project page > Advanced view).
JavaSettings_Test.java	Configure JVM settings that are available on the Project page.
LogSettings_Test.java	Configure log settings (Project page > General Settings view).
OSPlatform_Test.java	Configure OS or OS X settings (Project page > Platforms view).
MaintenanceMode_Test.java	Configure settings in the Maintenance Mode area (Project page > Advanced view).
MultiTierVirtualAppliance_Test.java	Configure settings for multi-tier virtual appliances.
PostInstallAction_Test.java	Configure the Post-Install sequence.
PreInstallAction_Test.java	Configure the Pre-Install sequence.
ProjectAutomation_Test.java	Open, configure, and save an InstallAnywhere project.
ProjectDescription_Test.java	Configure project settings that are available on the Project page.
ReadReplaceXMLandSevenZipandTar_Test.java	Configure the following actions for any sequence: ● Expand Archive (7-Zip) ● Expand Archive (TAR) ● Read/Modify XML File
Rollback_Test.java	Configure settings in the Rollback Settings area (Project page > Advanced view).

Table 9-132 ▪ Project API Code Samples (cont.)

File Name	Description
Rules_Test.java	Configure settings in the Installer Rules view on the Project page.
Tags_Test.java	Configure tag settings (Build page > Build Installers view > Build Configurations tab > Tags subtab) for your project.
UninstallPhase_Test.java	Configure the Uninstall sequence.
Upgrade_Test.java	Configure upgrade settings (Project page > Upgrades view).
VirtualAppliance_Test.java	Configure settings for a single-tier virtual appliance.
WindowsPlatform_Test.java	Configure Windows settings (Project page > Platforms view).

Project Automation APIs for Maintenance Mode and Instance Management



Note ▪ *If you are using a node-locked license of InstallAnywhere, the licensing libraries must be loaded while using the project automation API. This is typically done by configuring an IDE setting or by passing the following parameters through the command line when running the Java 11:*

```
--add-opens java.base/jdk.internal.loader=ALL-UNNAMED -
Djava.library.path=IA_HOME\resource\fnp\libraries
```

If the library path includes spaces, ensure that you enclose the library path within quotes:

```
-Djava.library.path="IA_HOME\resource\fnp\libraries"
```

InstallAnywhere provides API support to perform project automation for configuring maintenance mode and instance management behavior.

Project Automation APIs for Maintenance Mode Behavior

Maintenance mode methods begin with the `getMaintModeConfigs()` accessor.

Enabling/Disabling Maintenance Mode (setMaintModeSupportEnabled)

To enable or disable maintenance mode, use the following method:

```
project.getMaintModeConfigs().setMaintModeSupportEnabled(<Boolean value>);
```

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 9-133 ▪ 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>);`

In these methods, the `<Boolean value>` will be either **`true`** or **`false`**.

Project Automation APIs for Instance Management Behavior

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>);
```

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 9-134 ▪ 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 limit for the number of instances 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)

You 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 9-135 ▪ 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 9-136 ▪ Values for the setVersionPartIdentifiesInstance Method

Value	Option
20	Major
21	Minor
22	Revision
23	Sub-Revision

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 functionality, use the following method:

`project.getInstanceDefinition().setEnableOvertopCheck(<Boolean Value>);`

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 9-137 ▪ Values for the setOvertopBehaviour Method

Value	Option
30	Allow user to continue with overwrite
31	Do not allow user to overwrite

Associating Check Running Mode Rules to Individual Actions

To make sure that individual actions in any of the views in the Sequences view (Install, Pre-Install, Post-Install, Uninstall, Pre-Uninstall, Post-Uninstall) work with maintenance mode, you can associate the Check Running Mode rule using the following methods:

```
CheckRunningModeRule crmAdd=new CheckRunningModeRule();
crmAdd.setRunningMode((short)value);
```

In this method, the *value* identifies the selected mode:

Table 9-138 ▪ 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.*

Index

Symbols

.ear file 195
    expanding 166
.jar 166
.jvm file 271
.tar 166
.war
    expanding 166
.war file 195
.zip 166

Numerics

64-bit Windows 239
    challenges of supporting 239
    supporting 64-bit Windows-based systems without WOW64 242
    WOW64 emulation on 241
7-Zip 166

A

action execution sequence 102
Action Group action 615
    Comment Action Group option 180, 615
action groups 102
    and maintenance mode 205
actions 100
    action execution sequence 102
    action groups 102
    adding to Install sequence 172
    assigning a rule to 179
    assigning a rule to a group of actions 180
    common customizer properties 690
    common properties 690
    Console 101, 664
    customizers 102
    execution sequence 103
    file and folder operations before uninstallation during an upgrade 341
    General 101, 615
    Install 101
    overview 100
    Panel 101, 644
    Properties tab 102
    Rollback tab 103
    Rules tab 102
    System i (i5/OS) 673
    Tags tab 103
    types of 101
    upgrade-specific 341
Add Comment 615
Add Features 446
Add Jump Label 615
Add New/Manage Credential dialog box 542
Add/Edit Installer dialog box 539
Advanced Designer 396, 710
    adding files to a project 163
    adding folders to a project 163
    adding LaunchAnywhere executable 152
    adding Post-Install actions 153
    adding Pre-Install actions 150
    Build Installers view 497
    building the installer 155
    Install view 151, 491
    Installer UI page 460
    new project 149
    Organization page 476
    organizing features and components 162

Post-Install view 153, 494
Post-Uninstall view 497
Pre-Install view 491
Pre-Uninstall view 495
Project view 397
testing the installer 156
Uninstall view 495
Advanced JRE Handling for Version Upgrades option 35, 130
advanced runtime UI themes 460
Advertise Variables 228
advertising
merge module installer variables 228
Amazon EC2 347
adding installer 361
building virtual appliances 362
creating virtual appliances 359
credentials 360
deploying virtual appliance using Amazon Web Services (AWS) Management Console 386
EC2 console 388
EC2 instance store backed 360
elastic block store backed 360
manually deploying a virtual appliance 386
obtaining an AWS account 360
Root Device Type 360
S3 console 387
selecting deployment region 361
specifying Appliance URL 360
virtual appliance build output 383
Amazon Web Services (AWS) Management Console 386
EC2 console 386
S3 console 386
Ant 615, 766
build integration 233
task examples 233
task reference 766
Apache Tomcat support 191
deployment options 192
enabling end users to specify server details 195
API code samples 775
APIs 773
documentation 773
project automation 777
Appliance Configuration tab
Hardware subtab 371
Appliance Configurations
adding to project build 365
copying 364
creating 363
creating new 363
removing 365
renaming 364
Appliance URL 356, 360, 368
Appliance UUID 356, 359
ApplianceBuildResults.txt 383, 384
appliances (see virtual appliances)
archive files 166
expanding 166
Associate File Extensions 690
authentication 318
authentication and OS X 304
authentication wrapper 304
automation 774
API code samples 775

B

background images 116
installer mode 115
baselinevm.properties 352
specifying protocol 355
bidi orientation 105
Billboards 474
billboards 116
Browse for Folder dialog box 724
Build Appliances 508
Appliance Configuration tab 379, 509
Build Log tab 523
VM Configuration tab 516
Build Configurations
adding to project build 219
assigning Tags to project elements 221
associating Tags with 222
building 229, 381
building all 230
building via command line 230
copying 218, 364
creating 216
creating new 217
creating new Tags 220
editing 216
overview 216
removing 218
renaming 217
searching for Tags 222, 572
specifying locale settings 223
using Tags to customize 219
Build Installers view 497
Build Log subtab 507
building installers using build configurations 229
creating CD-ROM/DVD installers 225
creating installer for customized flavor of UNIX 224
creating merge modules 227
creating Web installers 225
defining targets 223
setting build distribution options 225
Build Log 507

build time variables 250
build tool 109
build.exe 109, 230
  command-line arguments 756
  command-line build 230
Build.lax
  setting JVM heap size 524
buildproperties.properties file 751
  compared to settings in Advanced Designer 752
BuildProperties.xml file 746
  compared to settings in Advanced Designer 747

C

CD-ROM installers
  creating 225
Change Product_Name Installation.exe 210
Check File/Folder Attributes rule 185, 701
Check If File/Folder Exists rule 702
Check Platform rule 187, 703
  examples 187
Check Running Mode rule 188, 704
  about 207
  and maintenance mode 207
Check System Architecture Rule 700
Check User-Chosen Language Rule 700
Choose a VM Wizard 526
Choose Alias, Link, Shortcut 644
Choose Database Connection console 666
Choose Database Connection panel 648
Choose Features to Uninstall 644, 664
Choose File 645
Choose Folder 646
Choose Install Folder 646, 664
Choose Install Sets 650, 664
Choose Java VM 651, 664
Choose Java VM panel 129
Choose Link Folder 664
Choose Remote System i (i5/OS) Install Folder 674
Choose Tomcat Deployment Option console 669
Choose Tomcat Deployment Option panel 659
Choose Uninstall Type 646, 664
Choose WebSphere Deployment Option console 671
Choose WebSphere Deployment Option panel 661
Classpath 690
Clean Components 164
ClearCase 244
color settings 105
command 279, 762
command line 279, 762
command-line build 109, 230, 330
  arguments 756
  installer and uninstaller arguments 762
  maintenance mode options 765
  platform arguments 759
  reference 756
command-line launcher
  arguments 766
Comment 615
Comment Action Groups 180, 615
Compare File Modification Timestamp Rule 700
Compare InstallAnywhere Variable Numerically rule 190
Compare InstallAnywhere Variables Numerically rule 706
Compare InstallAnywhere Variables Rule 700
Compare InstallAnywhere Variables Rule Numerically Rule 700
Compare Versions rule 707
components 113, 114, 162, 164, 477
  adding 162
  adding files to 164
  assigning components to features 165
  best practices 162
  integrating with target 164
  organizing 162
  removing empty components 164
console 279
Console actions 101, 664
console installers 116, 118
consoles 664, 762
Copy File 580, 615
Copy Folder 580, 615
Create Alias, Link, Shortcut 582
Create Folder 580, 583
Create InstallAnywhere VM Template Wizard 528
  Amazon OS Information panel 531
  Creating the Baseline VM panel 532
  Finish panel 533
  InstallAnywhere VM Template Information panel 529
  launching 528
  Review EULA for Creating the VM Template panel 531
  Review Summary Information panel 532
  VMware OS Information panel 529
  VMware vCenter/vSphere 5 Information panel 532
Create InstallAnywhere VM Template wizard 350
Create LaunchAnywhere for Java Application 584
Create LaunchAnywhere for Java Application action 126
Create LaunchAnywhere for Java Apps 580
Create Uninstaller 580, 587
createiso.properties 352
CreateVMTemplate command
  -acceptAllEULA option 354
  -amazon option 353
  -console option 353
  -createVM options 354
  examples 354
  -gui option 353
  hypervisor options 353
  -locale option 353

mode options 353
-modifyISO option 353
other options 353
-useCloudImage option 354
creating a JRE VM pack 552
credential store 366
Custom Code 294, 295, 615, 653, 664
custom code
importing InstallShield MultiPlatform APIs into InstallAnywhere 297
Custom Code API 111, 140
and variables 111
overview 110
plug-ins 112
custom code API 110
Custom Code Panel
scrolling 653
customizers 102
Properties tab 102
Rollback tab 103
Rules tab 102
Tags tab 103
Customizing a Check If File/Folder Exists rule 186
CVS 244

D

DB2 database support 200
Debug 295, 322, 325, 615
debugging 320, 323
LaunchAnywhere-launched executable files 323
OS X-based installers 325
UNIX/Linux or pure Java installers 324
user display message panel 322
Windows-based installers 323
Delete File 580, 616
Delete Folder 580, 620
dependencies
support for signed JARs 296
Deploy WAR/EAR Archive 589
deployment regions 361
Developer ID Application certificate 306
digital certificates 299
Disk Space Check 646
Display HTML 654
Display Message 646, 664
distribution options
setting 225
Distribution subtab 504
Do not limit instances 448
Dock
OS X 328
DVD installers 121
creating 225
dynamic text 132

E

EAR files 195
EBS Volume size (GiB) 361
EC2 (elastic compute cloud) 387, 388
EC2 instance store backed 360
Edit Dockerfile dialog box 554
Edit Installer Properties Table dialog box 358, 362, 554
Elastic block store backed 360
Device 361
EBS Volume size (GiB) 361
Enable Instance Management 448
Enable Instance Overwrite Warning 450
Enable Maintenance Mode Support 446
Enable Rollback 214
Enable Source Paths 168
Enable Update Notifications 580, 595
Encryption 250
Evaluate Custom Rule rule 189, 707
Execute Ant Script 615, 621
Execute Command 615, 623
Execute Custom Code 624
Execute Script/Batch File 615, 625
Execute Target File 615, 617
Execute Target File action 244
Execute Uninstaller 617
exit codes 330
Expand Archive 580, 599
External Resource Bundle 293, 294
external resource bundles 133
file format 134
naming conventions 134
support for Merge Modules in Install and Uninstall phases 134

F

Features 477
features 113, 114
adding 165
assigning files to 165
assigning to components 165
organizing 162
uninstalling 140
files 740
adding to a project 163
assigning to components 164
assigning to features 165
file formats 740
reference 740
updating location of 169
Find Component in Registry 164, 615, 626

Find File/Folder 647
FlexNet Code Aware integration 55
folders
  adding to a project 163
  SpeedFolders 139
font settings 105
fonts 166
  installing 166

G

Gatekeeper and OS X-based installers 304
General Actions 615
General actions 101
General Properties view 398
Generic JDBC Connection 201
Get Password 647
Get Serial Number 647, 665
Get System i (i5/OS) Login Credentials 675
Get User Input 665
Get User Input - Advanced 647
Get User Input - Simple 647
Get User Input panels 103, 694
  Bidi orientation 105
  color settings 105
  component types 104
  defaults 107
  font settings 105
  input methods 104
  results variables 105
  text reading order 105
  VAR_BOOLEAN_X variable 108
Get Windows Registry Entry 615, 617
graphical user interface (GUI) installers 116
  background images 116
  billboards 116
  localization 116
  look and feel 116
  panel additions 116
  splash screen 116
gui 279
GUI automation fixture 59

H

heap size 281
  setting intial and maximum values 524
help
  Help Library conventions 82
  using 82
Help view 475
helper tool 304
  for authentication on OS X 304
  generating and code signing 307
hosts 112
  application server 112
  database server 112
  operating system 112

I

ia.mac.filechooser.substituteSwingInsteadOfNativeForOlderJRE7 724
IABankingApplication sample project 392
IABookstore sample project 392
IBM WebSphere support
  deployment options 194
  enabling end users to specify server details 196
Identify Instance Using 449
Image Settings 691
Important Note 647
Install actions 101, 580
  adding 172
Install Archive 580, 602
Install Complete 647, 665
Install Failed 665
Install File 580, 602
Install from Manifest 580
Install HP-UX Depot 580, 581
Install Linux RPM 580, 581
Install Log File
  content 742
  format 741
  naming conventions 742
Install Merge Module 580, 604
Install Sets 113, 114, 476
Install Solaris Package 580
Install SpeedFolder 580
Install SpeedFolder action 607
Install view 170, 491
  adding LaunchAnywhere executable files 172
  installing fonts 166
install.exe 279, 762
InstallAnywhere
  opening existing project 88
  product registry 740
  projects 137
  variables 143, 144
InstallAnywhere Merge Module Import Assistant dialog box 557
InstallAnywhere Variable
  checking value of 250
InstallAnywhere.lax
  setting JVM heap size 524
installer modes 115
  console 116
  graphic user interface (GUI) 116
  silent 116

Installer Panel Additions 691
Installer UI page 460
Installer Valid VM List 125
installers
  assigning a rule to 179
  console 118
  creating basic 157
  defining rules 178
  graphical user interface (GUI) 116
  minimizing size of 319
  optimizing for Unix 319
  testing 237
InstallShield MultiPlatform
  calling ISMP APIs in InstallAnywhere 297
instance management 208
  identifying new instance by version 209
  settings 448
  upgrading instances 342
Introduction action 647

J

JARs
  support for signed JARs 296
Java virtual machines. See JVMs.
javadocs 773
JRE handing 35, 130
Jump to Target 615, 617
JVM heap size
  setting at runtime 281
  setting initial and maximum values 524
  setting via command line 281
JVM search instructions file 271
JVM Settings view 434
JVM Spec File 271
JVMs 122
  about Java VM selection criteria 130
  Create LaunchAnywhere for Java Application action 126
  criteria used in JVM search 125
  how a launcher selects a VM 125
  launcher's VM selection behavior at run-time 126
  No Specific VM option 128
  selecting 122
  selection by LaunchAnywhere launchers 123
  using first VM found matching search settings 128
  VM selected by the installer 127
  VM selected on the Choose Java VM panel 127
  VM Used by the Installer option 127
  when VM packs are installed 129
  when VM searches occur 124
  where VM searches occur 124

L

Label Settings 691
language code 279, 762
language codes 734
Launch Default Browser 615, 617
LaunchAnywhere 132, 144, 243, 319, 323, 710, 726
  command-line arguments 766
  LAX_VM command-line parameter 132
  overview 132
  properties 726
LaunchAnywhere executable
  adding 152
LaunchAnywhere executable files 323
LaunchAnywhere executables
  adding 172
launchers
  creating 243
launchers for Java Applications
  creating 243
LAX
  LaunchAnywhere executable 243, 323, 726
  properties 726
License Agreement 665
License Agreement (Panel action) 648
line 279, 762
List of Installer Steps 691
Local Repository 373
locales 116, 288, 289, 294, 325, 427
  language codes 734
  specifying in Build Configurations 223
Locales view 427
localization 116, 132, 293, 294
  best practices 133
  common localizable elements 736
  dynamic and static text 132
  external resource bundles 133
  language codes 734
  localizable elements 736
  overview 132
  reference 734

M

Mac App Store 304
Magic Folders 134, 143, 328, 730
  and variables 143
  overview 134
  variables 730
Maintenance Mode
  action groups 205
  Add Features 446
  adding a Check Running Mode rule 188
  command-line build options 765

configuring 204
Enable Maintenance Mode support 446
enabling 204
end user experience 211
Instance Management options 208, 448
launcher 210
Maintenance Mode launcher compared to Uninstaller 210
Manage Tags 451
options 204
overview 203
Remove Features 446
Repair Installation 446
settings 446
Uninstall Product 446
Manifest File Format 742
Manifest files 742
format 743
Match Regular Expression Rule 701
merge modules
advertising merge module installer variables 228
benefits of using 135
creating 227
External Resource Bundles
support in Install and Uninstall phases 134
integrating into projects 136
methods to add to existing installer 136
overview 135
showing progress of 229
uninstalling 140
upgrading 342
using third party merge modules 135
Modify Text File - In Archive 615, 628
Modify Text File - Multiple Files 615, 631
Modify Text File - Single File 615, 634
Move File 580, 581
Move Folder 580, 582
MySQL database support 200

N

National Vulnerability Database (NVD) 42
Native vs. Swing resources 724

O

Optimize registry entry 478
Oracle database support 201
Organization page 476
OS X 328
authentication for 304
code signing installers for 304
debugging 325
interaction between Gatekeeper and installers on 304
obtaining a code-signing certificate for 306
requiring elevated privileges on 304
troubleshooting code signing and authentication issues 314
verifying code signing 311
OutOfMemory error 524
Output Debug Information 615, 618
Output Text to Console 615, 618
OVA (OVF Archive) 357
OVF (Open Virtualization Format) 356, 367
OVF 1.1 format 367
OVF Appliance Type 356

P

Panel actions 101, 644
settings 691
Perforce 244
Perform XSL Transform 615, 618
Perform XSL Transform - In Archive 615, 619
platform 329
Platforms view 412
Plug-In Actions 687
plug–ins 112
Post-Install view 494
Post-Install vs. Post-Uninstall 142
Post-Uninstall view 497
Pre-Install actions 150
Pre-Install sequence 170
customizing 170
Pre-Install Summary 665
Pre-Install Summary Panel 658
Pre-Install view 491
Pre-Install vs. Pre-Uninstall 142
Prepare 568
Prepare the Helper Tool dialog box 568
Pre-Uninstall view 495
product registry 740
productVersion 283
project 137
creating new 86
opening 86
opening existing 88
switching to a different project 88
project automation API 774
code samples 775
for Maintenance Mode and Instance Management 777
project file 137
Project page 397
ProjectLocalizationInfo.txt 132
projects 137
adding components 162
adding features 165
adding files to 163

adding folders to 163
creating 158
defining Install sequence 170
defining rules 178
enabling installation rollback 214
opening existing 159
organizing features and components 162

Q

Query InstallShield Universal Software Information 619

R

Read/Modify XML File 619, 636
Ready to Install 665
Refresh Windows Environment 619
Register Windows Service 580, 619
registry 740
Remove Features 447
removing
empty components 164
Repair Installation 447
resources
updating location of 169
response file 279
response files 119, 120, 743, 762
Restart Windows 615, 619
Restrict to a Single Instance 449
Restrict to Number of Instances 449
rollback options 214, 451
Enable Rollback 214
Rollback Settings 214
Rollback Settings 451
rules 138, 432
applying multiple 181
assigning to a group of actions 180
assigning to an action 179
assigning to installer 179
Check File/Folder Attributes 185, 701
Check If File/Folder Exists 702
Check Platform 187, 703
Check Running Mode 188, 207, 704
Compare InstallAnywhere Variable Numerically 190
Compare InstallAnywhere Variables Numerically 706
Compare Versions 707
Customizing a Check If File/Folder Exists 186
customizing rules 185
defining 178
Evaluate Custom Rule 189, 707
how rules are evaluated during an installation 190
list of 699
Rules view 432
Run FlexNet Code Aware Analysis 55
Run SQL Script 610

S

S3 (simple storage service) 387
SCM 244
Scroll Panel Option 653
Scrolling Message 648
self-extractors 121
behavior on non-Windows platforms 121
behavior on Windows 121
service packs
obtaining for InstallAnywhere 96
Set InstallAnywhere Variable - Multiple Variables 615, 619
Set InstallAnywhere Variable - Single Variable 615, 619
Set System Environment Variable 580, 582
Set Windows Registry - Multiple Entries 580, 620
Set Windows Registry - Single Entry 580, 620
setting at run time 281
setup.exe 279, 762
Show Message Console 'Dialog' 665
Show Message Dialog 615, 640
Show progress dialog 483
signing 299
methods for OS X-based installers 305
OS X-based installers 304, 309
OS X-based installers, requirements for 306
troubleshooting OS X-based code-signed installers 314
verification of, for OS X-based installers 311
Windows-based installers 299
silent 279, 762
silent installer mode 116
silent installers 119
response files 120
source control management software 244
Source Paths 138, 168
adding and removing 168
enabling 168
predefined 138
substitution 138
SpeedFolders 139, 580
splash screen 116
SQL support
connecting to SQL databases 198
creating a database 202
obtaining DB2 drivers for 200
obtaining generic JDBC drivers for 201
obtaining MySQL drivers for 200
obtaining Oracle drivers for 201
running a SQL script at run time 202
setting up a database connection 198
supported databases and versions 197
Start, Stop, Pause Windows Service 620
static text 132

stderr 243, 320, 323, 690, 726
stdout 243, 320, 323, 690, 726
Strict VM Selection 131
Substitute Recursively 445
Swing vs. Native resources 724
System i (i5/OS) Actions 673
System i (i5/OS) Command 677
System i (i5/OS) Find Component in RAIR 677
System i (i5/OS) Install File 678
System i (i5/OS) Integrated File System (IFS) 679
System i (i5/OS) Library 680
System i (i5/OS) Licensed Program 681
System i (i5/OS) Licensed Program Exists Condition Rule 701
System i (i5/OS) Object 683
System i (i5/OS) Primary Language Install Condition Rule 701
System i (i5/OS) Process Remote Install (Merge Modules) 684
System i (i5/OS) Program 685
System i (i5/OS) Program Temporary Fix (PTF) 686
System i (i5/OS) Program Temporary Fix (PTF) Condition Rule 701

T

Tag Search Results dialog box 572
Tags
  assigning to project elements 221
  associating with Build Configurations 222
  creating 220
  searching for 222, 572
  using to customize Build Configurations 219
targets
  defining 223
team development 168
Templates 295
templates 139
  overview 139
testing 156, 237
text
  dynamic and static 132
theme
  advanced runtime UI themes 460
Tomcat support 191
  deployment options 192
  enabling end users to specify server details 195
Trigger Rollback 613
troubleshooting 319
  installers 319
tutorials 149

U

Uninstall Categories 176
Uninstall Category 614
Uninstall Complete 648, 666
Uninstall Files 614
Uninstall Folders 614
Uninstall InstallShield Universal Software 620
Uninstall LaunchAnywhere 614
Uninstall Merge Modules 614
Uninstall Product 447
Uninstall Registry Entries 614
Uninstall Shortcuts/Links/Aliases 614
Uninstall view 495
Uninstaller 140, 580, 691
  about 210
  custom code 140
  feature uninstall 140
  features 140
  InstallAnywhere variables 140
  merge modules 140
  multiple merge modules 140
  multiple products 140
  overview 139
Uninstaller Introduction 648, 666
UNIX
  creating installer for customized flavor of 224
  debugging 324
Unix
  optimizing installer size 319
updates
  obtaining for InstallAnywhere 96
upgrades 337
  and the uninstaller 342
  configuring 338
  detecting earlier version 339
  enabling support for 338
  file locking 342
  getting started creating 338
  instance management 342
  merge modules 342
  requirements for 337
  shared components 342
  standard variables that refer to the base version 342
  view 452
Upgrades view 452

V

VAR_BOOLEAN_X variable 108
variables 111, 143, 247, 322, 709
  advertising merge module installer variables 228
  and Magic Folders 143
  comparing using Compare InstallAnywhere Variables

Numerically rule 190
evaluate at assignment 145
evaluating 144
IA_BROWSE_FOLDERS 724
LAX properties 726
Magic Folders 730
methods of setting 144
notation 143
resolving at build time 250
setting 144
standard InstallAnywhere 710
Substitute Recursively option 445
Variables view 443
version
setting at build time 283
Version Field Level That Identifies New Instance 450
virtual appliance
deploying 359
virtual appliances 345
about 345
adding an installer 357
adding installers 374
advantages of 346
Ant task-based build 382
Appliance Configurations 363
Appliance URL 356, 368
building 358, 381
command-line build 382
components of 345
creating 345
custom operating system support 375
customizing product properties 379
deploying 384
dynamic discovery of OS packages 375
EULAs 356
IABankingApplication sample 392
IABookstore sample 392
installing OS from local repository 373
installing OS packages 372
manually deploying 385
multi-tier 380
overview 348
running custom scripts 377
specifying hardware requirements 371
supported platforms and hypervisors 347
using a credential store 366
using proprietary operating systems 373
versioning 380
VM configurations 370
VM templates 349
workflow 348
VM Configuration tab
Hardware subtab 518
Installers subtab 357, 520
OS Packages subtab 372, 519
Product Properties subtab 517
Repository Settings subtab 373, 522
Script Info subtab 521
VM packs
downloading 238
installing 239
overview 237
VM templates 349
creating 350
creating for VMware vCenter/vSphere 351
creating using command line 352
default credentials 351
downloading 349
launching Create InstallAnywhere VM Template wizard 350
obtaining 349
VM Type 131
VM Version 131
VMware vSphere 5 347
adding an installer 357
building a virtual appliance 358
deploying a virtual appliance 359
manually deploying 385
OVA 357
OVF 356
OVF Appliance Type 356
vCenter compatible vs. vSphere compatible 356, 367
virtual appliance build output 358, 383
virtual appliance output options 367

W

WAR files 191, 195
Web installers 121
creating 225
WebSphere support
deployment options 194
enabling end users to specify server details 196
Win32 installer 323
Windows Console 243
Windows NT 323
troubleshooting 323
Windows WOW64 Emulator Settings 448
InstallAnywhere Win32 Services 448
WOW64 redirection 448
WOW64
emulator settings 448