

This partial chapter is an excerpt from the *InstallAnywhere 2009 Training Manual*. For information about purchasing this training manual and to learn about other Acrecco software training materials, please visit www.acresso.com.

Building Releases

Once you have defined the properties, file links, behavior, and other related information of your installation project, you can build releases for testing and eventual duplication and distribution. This chapter describes how to build releases from the graphical InstallAnywhere environment.

For additional information on automated tools for building your InstallAnywhere projects, refer to [Chapter 12, “Integrating InstallAnywhere with Automated Build Environments”](#).

Build Targets

In the **Build** task, you define the target operating systems for your installer as well as defining the form for distributing the installer. In **Build > Build Targets**, you specify the desired target operating systems. You also indicate whether to provide a VM (Java Virtual Machine) with the installer to provide greater ease of use for the end user. For any build target that includes a VM, you should select the VM to bundle in the **VM to Bundle with Installer** list.

Starting with InstallAnywhere 2008 (Enterprise Edition), you can define *dynamic build targets*: using the **Build Targets** tab (pictured in the following figure), you can create and delete build targets, and create more than one build target per platform. For example, a single project can contain a build target for Windows with no VM, Windows with an IBM VM, and a Sun VM.

To create a build target, click **Add** at the bottom of the list of existing targets; to delete a build target, click the **X** icon to the left of the target.

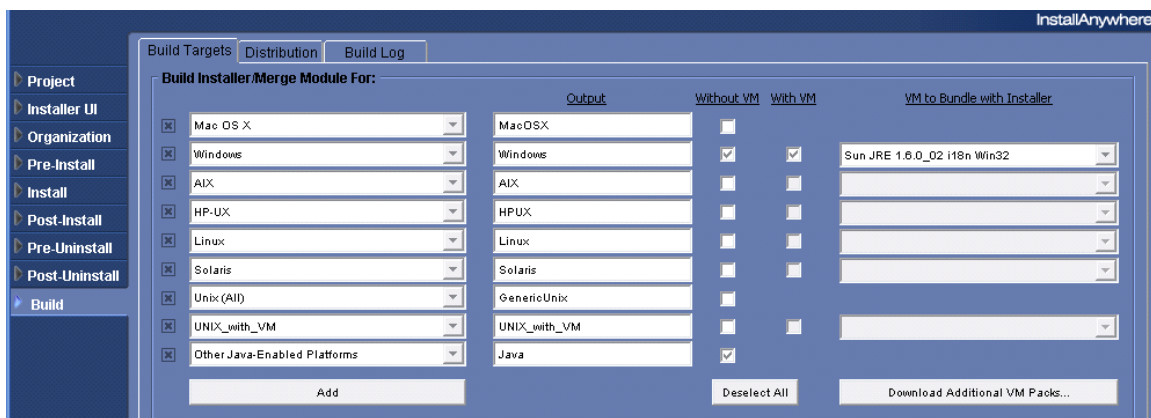


Figure 4-1: Build Targets Tab

When you click **Build Project**, all of the targets with a selected check box in the **Without VM** or **With VM** column will be built. While the build is taking place, a progress indicator similar to the following is displayed.

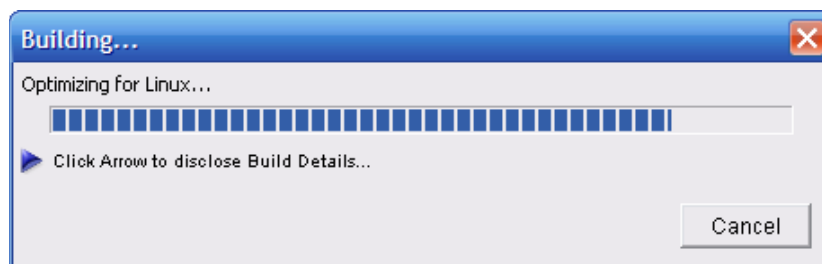


Figure 4-2: Build-progress Indicator

While InstallAnywhere provides options for many flavors of Unix, it also enables the creation a generic Unix (Unix (All)) and of other, custom flavors. To create an installer for a flavor of Unix that is not in the list of platforms, select an existing target (or create a new target) of type UNIX_with_VM, and if desired enter the custom target's name in the **Output** field.

VM Packs

VMs that you bundle with a target (for the sake of target systems that may not have an appropriate VM installed) are implemented as *VM packs*. InstallAnywhere ships with some VM packs for you to bundle, and the Aceso Software web site provides additional VM packs for you to download. Clicking the **Download Additional VM Packs** button in the **Build Targets** tab brings up the VM Packs section of the Aceso Software web site.

VM packs are stored as `.vm` files, which are `.zip` or `.jar` archives that contain the Java VM and a `vm.properties` file. (The `vm.properties` file contains display and platform information about the VM pack.) These VM packs must be stored as resources in the directory `InstallAnywhere/resources/installer_vms`. The selected bundled VM will be saved on a project-by-project basis. You can also add new VM-pack locations using **Edit > Preferences > Resources > VM Pack Resource Paths**.

In addition to the default VM packs available to InstallAnywhere, you can create your own. For information on creating a VM pack, see the InstallAnywhere help topic “Creating VM Packs.”

In the **Bundled Virtual Machine** area of the **Project > Java** task, you can specify whether a VM you bundle with your release image should be installed on a target system (as opposed to being temporarily installed for the sake of the running installer).

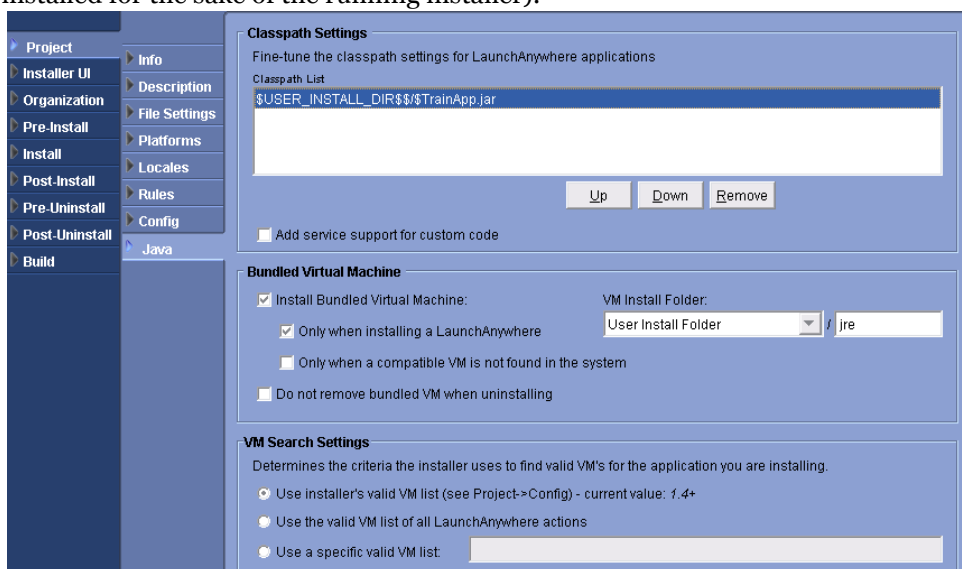


Figure 4-3: Bundled Virtual Machine Options

VM Selection

For releases with which you have not bundled a VM, you can specify which Java versions can be used with the installer, using the **Valid VM List** setting in the **Project > Config** task.

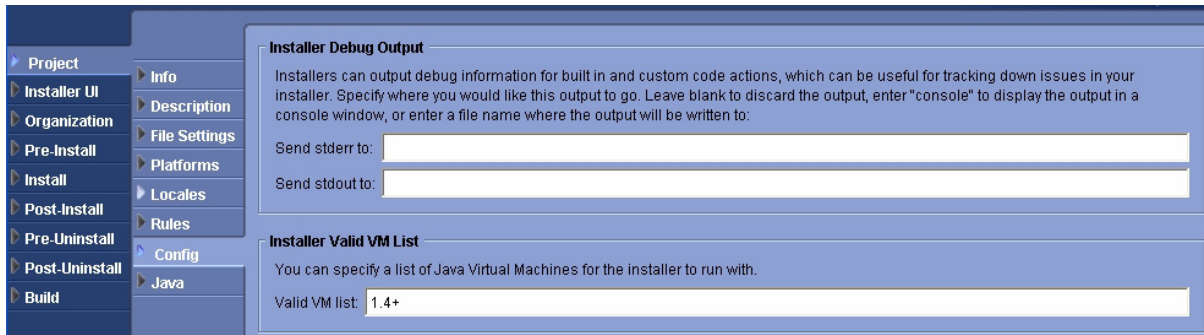


Figure 4-4: Specifying a Valid VM List

You can specify strict VM selection parameters in your launcher—for your installed application—by setting the LaunchAnywhere property `lax.nl.valid.vm.list`, or you can set the property for your installer in the **Project > Config** tab of the Advanced Designer.

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

- ALL (any VM)
- JDK (any JDK)
- JRE (any JRE)

Alternately, you can use a strict VM expression such as “JRE_1.5.1_03” or “JDK_1.4.2_02”, joining JDK or JRE, with an underscore character, to a version number.

“JRE_1.4.2_02” enables the installer or application to run only against the JRE 1.4.2_02. A value of “JDK_1.5.0_06” enables the installer or application to run only against a JDK 1.5.0_06.

You can specify minimum or wildcard versions of a specific VM, using the + or * operators:

- JRE_1.4+ selects any JRE-type JVM of version 1.4.0_0 or greater.
- JDK_1.4.2* selects any JDK-type JVM of the 1.4.2 series.

If more than one of these expressions is present, they will in effect be combined with an OR operator. In other words, a VM is valid if it matches any of the given expressions.

The optional JDK or JRE specifies which type of VM is valid. If specified, it must be followed by an underscore character. You can specify only one or the other.

The version number can have varying degrees of precision; however, it is recommended that you have at least the major and minor version numbers specified.

The + or * operator at the end of a version are used to specify a version range. When using these operators, it is assumed that any unspecified version part is zero (specifying “1.6” is interpreted as 1.6.0_0). The + operator means “at least this version”, and the * operator means “of this version”. If you do not specify an operator, only versions that exactly match the specified version are valid (“1.4” does not validate for 1.4.2_02 JVMs).

Beginning with InstallAnywhere 2009, you can indicate a particular vendor whose VMs you want to detect, where common vendor names are “IBM”, “SUN”, “HP”, and “APPLE”. If you specify a vendor, it must be separated from other specifiers with an underscore character. For example, to require an IBM JRE with version 1.5.0, the specifier would be “IBM_JRE_1.5.0*”; and to detect any IBM version 1.5.0 VM, use “IBM_ALL_1.5.0*”.

Distribution

In the **Distribution** subtask, you define the form of the release image to be distributed. You can build and optimize an installer on a single or multiple CD-ROM discs, an installer for use over the Web, or an installer to be launched from an HTML file.

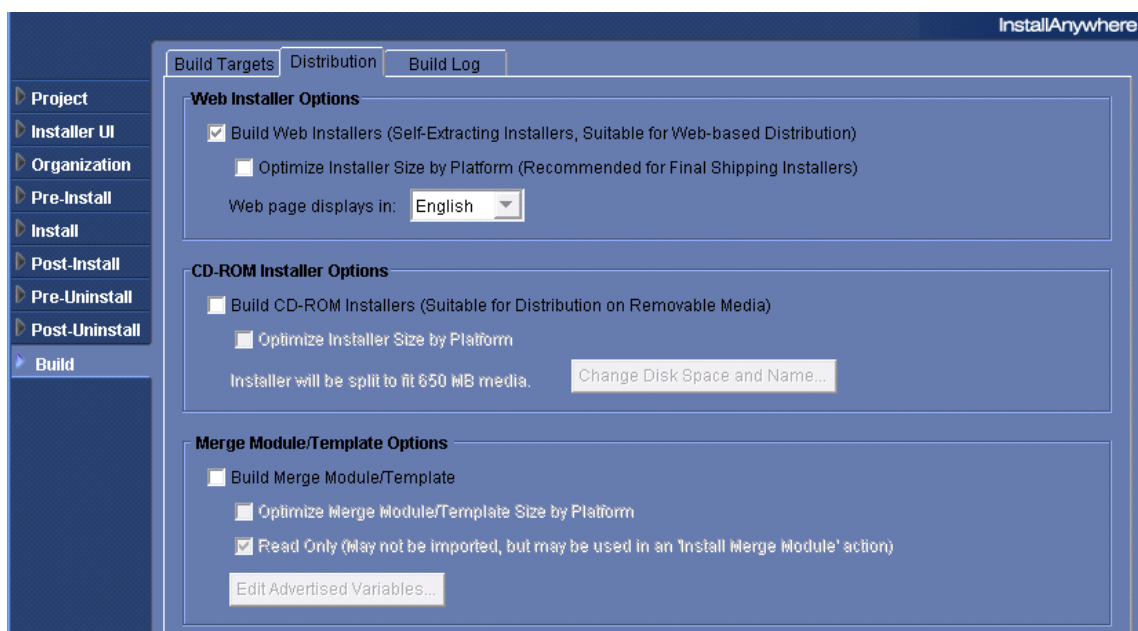


Figure 4-5: Distribution Tab

The **Distribution** subtask also enables you to build and optimize Merge Modules and Templates.

Web Installers

The web installer is a single executable file that contains all of the necessary installation logic. Building the web installer also generates an HTML page and embedded Java applet to make downloading the installer over the web easy. Select **Optimize Installer Size by Platform** to minimize the size of the final installers by excluding platform-specific resources (this is determined by evaluating Check Platform Rules). You can also select in which language to build the target web page.

The web page presented to the user appears similar to the following.

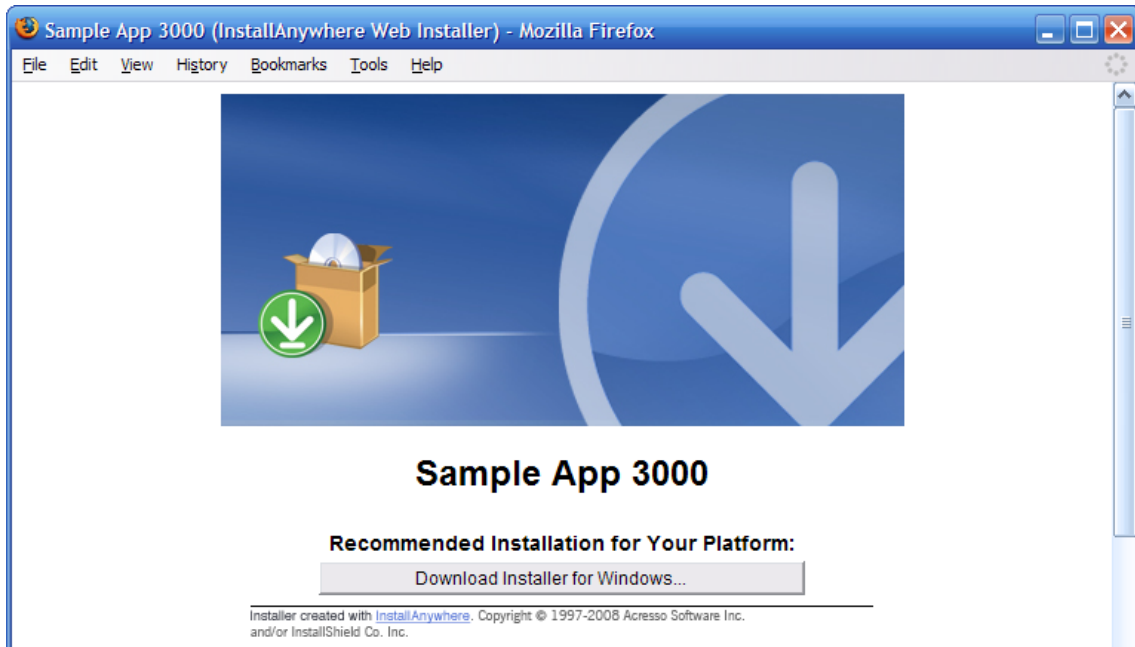


Figure 4-6: Sample Web Installer

You can test the web installer by clicking **Try Web Installer** at the bottom of the **Build** task after a successful build.

CD-ROM/DVD Installers

CD-ROM/DVD installers consist of multiple files meant to be burned onto one or more CD-ROM disks, DVD-ROM disks, or other removable media. They can also be placed on network volumes to provide easier access to large installers. The output of this build process can be directly burned onto disk.

InstallAnywhere CD/DVDs Installers have the ability to span multiple CDs/DVDs. By default, the installer will automatically segment the installer into a new disk if the size of the installer exceeds the media size (default: 650 MB). To control when InstallAnywhere will span to new disks, configure your disk names and size by clicking the “Change Disk Space and Name” button. This enables you to set the size for each disk, as well as set its name. The name will be displayed during the install process when the installer asks for the next disk in a set.

Burning CD-ROM Installers

The directory structure for CD-ROM installers is:

```
Platform1/Disk1/InstData/  
  |-...  
  |-MediaId.properties  
  |-Resource1.zip  
Platform1/Disk2/InstData/  
  |-MediaId.properties  
  |-Resource2.zip  
Platform1/Diskn/InstData/  
  |-MediaId.properties  
  |-ResourceN.zip  
...
```

Disk 1 typically contains an installer binary, often inside a VM or No VM directory. For example, when the Without VM and With VM options for Windows are both checked on the Build Targets task, InstallAnywhere will place 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 the following:

```
Windows/Disk1/InstData/  
  |-No VM  
    |-install.exe  
  |-VM  
    |-install.exe  
  |-MediaId.properties  
  |-Resource1.zip
```

This build output might appear in Windows Explorer as follows:

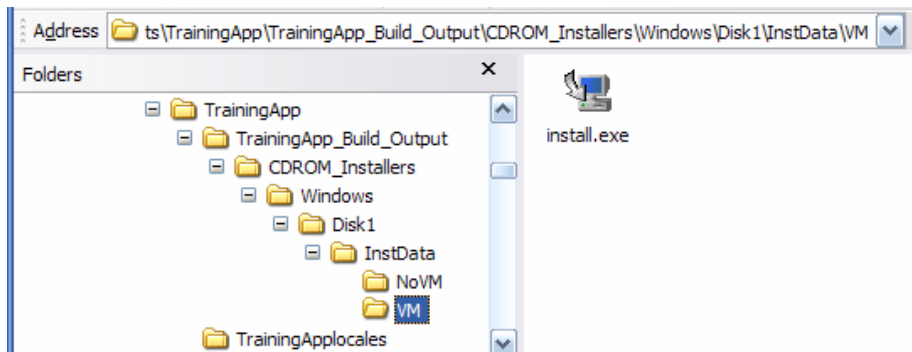


Figure 4-7: Build Output in Windows Explorer

However, on other platforms, such as Mac OS X, Unix (All), and Other Java-Enabled Platforms, the install binary is contained in the `Disk1/InstData` directory. On Mac OS X, for example, the directory structure for Disk 1 is:

```
MacOSX/Disk1/InstData/  
|-install.app  
|-MediaId.properties  
|-Resource1.zip
```

When burning CDs or DVDs, ensure that the folders `Disk1`, `Disk2`, etc., are burned as-is to the disk. Burning only the contents of these folders will cause installers to work incorrectly. The directory structure for the disk-burning application should be:

```
ISO CD NAME  
|-Disk1  
|-Disk2
```

Merge Modules and Templates

You can also build *merge modules* in this task. Merge modules enable you to create installers that can easily be integrated into other InstallAnywhere installers. More information on Merge Modules and Templates is available in [Chapter 11, “Advanced Organizational Concepts”](#).

Build Log

The **Build Log** window displays an XML log of the build once an installer project is successfully built. Click **Refresh Log** to display the current log, and click **Clear Log** to remove the log.

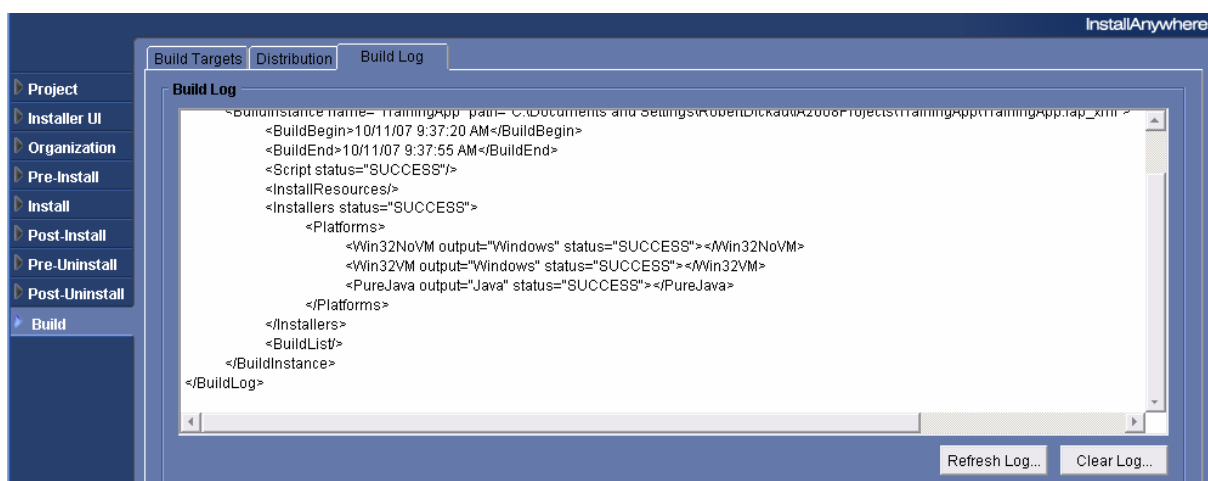


Figure 4-8: Build Log

