As of Plone 4.0.9 and Plone 4.1 there are new Windows installers being provided to the community. These installers behave differently than previous versions. This document explains the changes and the rationale behind them.

Table of Contents

Overview

Previously the Plone Windows installer ran buildout after allowing the user to input several variables, including an installation directory. However, due to the way that Python, Windows and buildout interact, long or complex installation directories did not always work, leading to fatal errors during the buildout installation process. This failure would be very difficult for an evaluator to recover the installation, thereby creating a bad “first impression” for a new user.

The new Plone Windows installer trades off some of this flexibility for vastly improved reliability. Buildout is run when building the installer, not “live” during the installation process. This means that buildout can never fail during installation. However, this means that the Windows installer will only install Plone into C:\Plone41 (for 4.1) and C:\Plone42 (for 4.2).

For users who require more flexibility in installation paths, we have provided the ability to copy-and-paste an installation directory to a new location, as well as a new facility to create custom Windows installers for Plone (which ships with the Plone Windows installer). Now, you can modify your buildouts and regenerate new Windows installers with different configurations (e.g., using multiple ZEO clients, multiple databases or different products), which you can use in production or distribute to customers.

Details

Plone’s Windows services now use a recipe called enfold.recipe.winservice which is a fork of the z3c.winservice recipe for use in Zope 2. The new Plone Windows installer has three key differences from previous versions:

- Default credentials for Plone
    - Previously you could create these from inside the installer.

    - Now it is hardcoded as login: admin and password: admin. This makes the Windows installer consistent with the Unified Installer and the Mac OS X installer. This can be changed by editing buildout.cfg, see the [shared] section which contains the line: user = login:password
- Windows Service name
    - Previously this was auto-generated during installation
    - Now it is defined in buildout.cfg see the [service] and [service-zeo] sections name = Unique Service Name
- Installation Location
    - Previously the user could select/create new folder to install Plone
    - Now it installs into C:\Plone41 or C:\Plone40 (depending on version)

How to Change Installation Location

The new Plone Windows installer does not allow you to change the installation location when you run the installer. However, it is still possible to install Plone into a different location after the initial installation, albeit via a more manual process. For example, suppose you have installed Plone into C:\Plone41 and after evaluation you want to copy it to a more permanent location, lets say Z:\Plone41-Intranet\.

The good news is that your installation directory, C:\Plone41, is 100% self-contained. There are no external dependencies defined anywhere else. You can simply copy/paste this directory, modify your buildout.cfg file (described below), and rebuild your custom environment. For a production-quality deployment of Plone, you will nearly always need to do this because the default settings for Plone Windows installation are very basic, and intended more for evaluation than for production deployment scenarios.

Here's a simple example for moving Plone from its default installation directory (C:\Plone41) to a new directory, Z:\Plone41-Intranet

- Change to Z:\ drive
- mkdir Plone41-Intranet
- cd Plone41-Intranet
- cp C:\Plone41* .
- switch to Z:\Plone41-Intranet
- edit the buildout.cfg file
    - goto [shared] section
    - change **user** variable to login:password they want to use
    - change the **http-address** to a unique port, say 9090
    - change the **zeo-address** to a unique port, say 9999
      [shared]
      user = administrator:s3kr1t

http-address = 9090
zeo-address = 9999
debug-mode = off
verbose-security = off

- goto [service] section
- change **name** variable to “Plone 4.1 Intranet”
  [service]
  recipe = enfold.recipe.winservice:service
  name = Plone 4.1 Intranet
  runzope = run-instance

- goto [service-zeo] section
- change **name** variable to “Plone 4.1 Intranet Database”
  [service-zeo]
  recipe = enfold.recipe.winservice:service
  name = Plone 4.1 Intranet Database
  runzope = run-zeo

- re-build the configuration files
  - **bin\buildout.exe**
- now install your new service
  - **bin\instance.exe install**
  - **bin\zeo_service.exe install**
- you can now start your database server service
  - **bin\zeo_service.exe start**
- It is always best to start your client/instance in foreground mode. If this fails you a substantial configuration problem or source code mismatch in your system.
  - **bin\instance.exe fg**

  Your terminal will be connected to the server. If it says “Zope Ready to Serve Requests” you are good to. Control-C or Break out of it and start the service from services panel or command line
  - **bin\instance.exe start**

NOTE:
I believe there is a problem with registry and python. Since we *do not* register the python system-wide you will need to have your current working directory as Python. **The above bin\instance.exe, bin\zeo_service.exe are incorrect.** So the command will be as follows:

Z:\Plone41-Intranet> bin\buildout
Z:\Plone41-Intranet> cd python
Z:\Plone41-Intranet\Python> python ..\bin\service.py --startup auto install
Z:\Plone41-Intranet\Python> python ..\bin\service-zeo.py --startup auto install

Building a Custom Windows Installer

The biggest feature of the new installer is the ability to create a custom Windows installers, which you can use or distribute. Previously, building a custom Windows installer required substantial knowledge of how many different sub-systems work together. Now you simply edit the installer.cfg buildout recipe. Re-run buildout with -c installer.cfg and you will get a new Windows installer executable in the current working directory.

More details later. See the “Windows Packaging Details” link below for more information.

References

- Windows Packaging Details - http://package.enfoldsystems.com/docs/windows.html
- Buildout website - http://www.buildout.org/

NOTES FOR REVISION OF building custom windows installer

Required SVN, Python 2.6, Innosetup
http://www.jrsoftware.org/isdl.php
Download isetup-5.4.2.exe
Put it in your path
start cmd (as administrator)
test: ISCC and ensure you see it

Grab SDK:
Windows SDK (for Win32 and .NET Framework)
download mt.exe
put it in your path ala
C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bin
start cmd and ensure its in your path: mt

mkdir isntaller
cd installer
svn co
https://svn.enfoldsystems.com/public/plone-packages/plone41-windows-installer/installer.cfg
c:\installer>c:\Python26\python.exe bootstrap.py -c installer.cfg

c:\installer>bin\buildout.exe -c installer.cfg
c:\installer>bin\installer.exe

You will get a buncha errors around mt tools and pyd files; ignore them.

mt.exe : general error c101008c: Failed to read the manifest from the resource o
f file "c:\INSTAL~1\build\plone41\417902~1.2\EXTERN~1.6-S\DLLs_testcapi.pyd". T
he specified image file did not contain a resource section.

ERROR:build:""C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bin\mt.EXE" "-
nologo" "-inputresource:c:\INSTAL~1\build\plone41\417902~1.2\EXTERN~1.6-S\DLLs_
tkinter.pyd;#2" "-out:c:\INSTAL~1\build\plone41\417902~1.2\EXTERN~1.6-S\DLLs_tk
inter.pyd.manifest"" failed: 31
ERROR:build:Last 100 lines of output:

NOTE:
- If you are building an installer and it fails half way through the build. You will probably have to delete the installation directory and all of the files; and re-run. the build process is not re-entrant (patches accepted).