

Changes to ZenPacks in Zenoss 2.2

Sprint 2 Note

The information below is subject to change before the Zenoss 2.2 release. Creation of new ZenPacks and migration of Basic ZenPacks as described below is believed to be working. Migration of Complex ZenPacks is not yet implemented.

Introduction

With Zenoss 2.2 ZenPacks are now packaged in a new format called python eggs. Eggs are a standard python method for packaging and distributing code. This document contains technical information intended primarily for developers of ZenPacks.

What this means to users of ZenPacks

There will be very little change noticeable to ZenPack users. Zenoss 2.2 will support the use of both old-style zenpacks as well as the new egg format zenpacks. Please note that the zenpack command should still be used for installation and removal of zenpacks, not the easy_install command that is frequently used with non-zenpack python eggs.

What this means to developers of ZenPacks

Though Zenoss 2.2 will still support installation and use of non-egg ZenPacks, you will no longer be able to create non-egg ZenPacks through the gui. Also, support for non-egg ZenPacks will likely be dropped from future versions of Zenoss. You should begin distributing your zenpacks in egg format soon.

We are working hard to make the transition to the new format as painless as possible. ZenPacks which were created entirely through the Zenoss gui should be very simple to migrate. The process for this is described below in the section "Converting Basic ZenPacks to Eggs." Those that provide custom ZenPack subclasses, datasource classes or other python classes are more involved. The process for these is described below in the section "Converting Complex ZenPacks to Eggs."

Converting Basic ZenPacks to Eggs

A new command called eggifyzenpack will do most of the work involved in converting a basic, old-style ZenPack to the new egg format. This script will create a new zenpack directory in \$ZENHOME/ZenPackDev for your zenpack, copy the contents of your zenpack to a subdirectory, then delete your zenpack from \$ZENHOME/Products.

You must decide on a package name for your ZenPack. Python code will reference your ZenPack as ZenPacks.<PACKAGE>.<ZENPACKID>. The intent of the package namespace is to group ZenPacks provided by the same developer. You should use the same package name for all of your ZenPacks. There are a couple restrictions to be aware of:

1. The package name Zenoss (and all capitalization variations of) are reserved for use by Zenoss.
2. All package names and ZenPack names must start with a letter and contain only letters, digits and underscores. Note especially that dashes are not valid.

The steps for running the script are:

1. Make a backup of your ZenPack by copying the .zip file or directory somewhere out of the way.
2. Make sure your current ZenPack is installed.
3. Determine a package name you wish to use.
3. Run the following command:

```
> eggifyzenpack --package <YourPackageName> <YourZenPackId>
```

As with previous versions of Zenoss, the Export ZenPack menu item will create a distributable file in \$ZENHOME/exports. New-style ZenPacks will end with .egg rather than .zip however.

Converting Complex ZenPacks to Eggs

Not yet completed

ZenPack Egg Structure

When the user creates a new ZenPack through the Zenoss gui a corresponding directory is created in \$ZENHOME/ZenPackDev with the ZenPack's name. That directory's structure looks like this:

setup.py : This file describes the parameters necessary in creating the egg. ZenPack developers should usually edit this information through the ZenPack edit page within Zenoss rather than directly in setup.py. See the page for setuptools under the References section below for further information on setup.py files. Comments within the Zenoss created setup.py provide some description of its contents also.

ZENPACKID.egg-info : This directory contains files describing the python egg. They are updated any time a ZenPack is edited and saved. ZenPack developers should normally not edit any of these files manually.

ZenPacks : Within python ZenPacks are now referenced through a hierarchical package namespace. The top level is ZenPacks, followed by the package name provided when a ZenPack is first created, followed by the name of the ZenPack itself. This directory and it's subdirectories follow that same structure.

ZenPacks/PACKAGE : This directory uses the package name given when the ZenPack was first created. If the ZenPack was migrated with the eggifyzenpack command then this is the --package option that was used with that command.

ZenPacks/PACKAGE/ZENPACKID : This directory has the same name as the ZenPack. It's structure and use are the same as that of non-egg ZenPacks.

Known Issues

* The zenpack command will allow removal of a ZenPack that is a dependency of other installed ZenPacks. This causes a situation where the remaining ZenPack whose dependency is missing will be broken. You can fix this situation by reinstalling the dependency, editing dependencies so no other ZenPacks depend on this one, then removing again. See issue below regarding need to export zenpack in order for dependency changes to take effect.

* ZenPack dependencies are not written to the egg until an export is performed. Therefore the dependencies indicated in the gui may not reflect the actual

dependencies in effect. Until this is fixed you should re-export a ZenPack every time you make changes to its list of dependencies.

Feedback and Problems

Discussion concerning development of the new ZenPack framework takes place on the zenoss-dev forums and mailing list:

<http://community.zenoss.com/forums/viewforum.php?f=3>

References

An introduction to python eggs:

<http://peak.telecommunity.com/DevCenter/PythonEggs>

Description of the structure of python eggs, entry points and other technical details of building eggs:

<http://peak.telecommunity.com/DevCenter/setuptools>