



ZenPack Developers' Guide

Version 1.0

DRAFT

Work in progress - for review

Please provide comments to

jane.curry@skills-1st.co.uk

This work is copyright © Zenoss Inc.

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit

<http://creativecommons.org/licenses/by-sa/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.



Synopsis

ZenPacks are the extension mechanism provided by Zenoss to build new functionality and also to easily port customisation from one Zenoss server to another. Some documentation is provided in the old Zenoss 3 Developer's Guide but updates to this are long overdue. This paper is intended to enhance, extend and update that documentation, including several sample ZenPacks.

The process of creating, modifying and exporting ZenPacks is discussed, along with debugging hints. The sample ZenPacks explore:

- Zenoss architecture
- ZenPack architecture
- Creating new object classes and relationships
- The zenpacklib tool
- Creating new collector modeler plugins to populate the new classes with data
- Converting old, non-zenpacklib ZenPacks to use zenpacklib
- Creating code for web pages for new types of objects, both JavaScript and TAL
- Creating new performance datasources and data templates
- Converting COMMAND-based ZenPacks to use the PythonCollector
- Incorporating new event classes, triggers and notifications in ZenPacks
- Creating and extending menus
- Extending functionality using routers and facades
- Logging and debugging
- The process of ZenPack creation, GitHub and ZenPack submission to Zenoss



Another objective of the paper is to provide examples of good practice. These tips are highlighted throughout the document with a green tick symbol.

The final main objective is to offer deeper insights into the architecture and functionality provided in the standard Zenoss code, especially with reference to how the ZenPack developer uses and extends this code. These sections should probably be skipped initially by someone first starting out with ZenPacks. There are a number of complete sections and they are prefaced with *. Smaller, in-depth points throughout the paper are highlighted with a yellow asterisk.



At the time of writing (Spring 2016) Zenoss 4.2.5 is the main Zenoss production platform. Many Zenoss 3 implementations are still in use and Zenoss 5.x is emerging amongst early adopters. The paper will major on 4.2.5 practices but will also cover differences for Zenoss 3 and Zenoss 5.

It is assumed that the reader is familiar with basic SNMP and Linux concepts and with standard Zenoss configuration techniques.

This paper was written based largely on Zenoss 4.2.5 with zenup fix 457, on a CentOS 6.3 platform. The hostname of the Zenoss 4 server is *zen42.class.example.org*. Some examples are shown specifically for Zenoss 5 (5.0.7), where the Zenoss 5 server is *zen50.class.example.org*.

Notations

Throughout this paper, text to be typed, file names and menu options to be selected, are highlighted by *italics*; important points to take note of are shown in **bold**.



Points of particular note are highlighted by an icon.



Points of good practice are highlighted with a tick icon.



Subtle or more advanced points are denoted with a star icon.

ZenPack samples

All the ZenPack samples can be found on GitHub at <https://github.com/ZenossDevGuide>

Table of Contents

1 Zenoss Architecture.....	13
1.1 Zenoss concepts.....	13
1.1.1 Devices, components, object classes and device classes.....	13
1.1.2 Zenoss monitoring functionality.....	15
1.1.3 Standard conventions for Zenoss code and ZenPacks.....	17
1.2 Zenoss Daemons.....	17
1.3 Zenoss 5 docker architecture.....	20
1.4 Extending Zenoss out-of-the-box functionality.....	20
2 What are ZenPacks?.....	21
2.1 Sources for ZenPacks.....	21
2.1.1 Free ZenPacks developed by Zenoss.....	21
2.1.2 Community developed ZenPacks.....	21
2.1.3 Chargeable Zenoss ZenPacks.....	22
2.1.4 Write your own ZenPack!.....	22
2.2 ZenPack basics.....	22
2.3 Existing ZenPack documentation.....	23
2.3.1 High-level documentation.....	23
2.3.2 zenpacklib documentation.....	24
2.3.3 Standard Zenoss documentation.....	25
2.3.4 Community ZenPack documentation.....	25
3 The mechanics of building a ZenPack.....	25
3.1 ZenPack development environment.....	25
3.1.1 Zenoss 4 and earlier.....	25
3.1.2 Zenoss 5.....	26
3.1.2.1 zenoss user.....	27
3.1.2.2 Common directory between containers and the base host - /z.....	28
3.1.2.3 Configuring the service for a development minimum.....	29
3.1.2.4 Useful references for managing a Zenoss 5 environment.....	30
3.2 ZenPack creation.....	31
3.2.1 What's in a name?.....	31
3.2.2 ZenPack directory hierarchy.....	32
3.2.3 ZenPack creation for Zenoss 4 and earlier.....	36
3.2.4 Zenoss 5 ZenPack creation.....	37
3.2.5 ZenPack creation using zenpacklib.....	37
3.3 Exporting ZenPacks.....	38
3.4 Installing ZenPacks.....	38
3.4.1 Installing ZenPacks on Zenoss 4.....	40
3.4.2 Installing ZenPacks on Zenoss 5.....	41
3.5 Removing ZenPacks.....	41
4 Simple ZenPacks.....	42
4.1 Adding performance templates to a simple ZenPack.....	42

4.1.1 Adding SNMP performance templates to a ZenPack.....	43
4.1.2 Adding zencommand performance templates to a ZenPack.....	44
4.2 Adding SNMP MIBs and event classes to a simple ZenPack.....	46
4.3 Adding device classes to a simple ZenPack.....	49
4.4 * Adding services and processes to simple ZenPacks.....	50
4.4.1 Adding IP services to a ZenPack.....	50
4.4.2 Adding Windows Services to a ZenPack.....	53
4.4.3 Adding Processes to a ZenPack.....	53
5 Understanding core Zenoss objects.....	55
5.1 Device.py.....	57
5.1.1 Object attributes.....	58
5.1.2 Object relationships.....	60
5.1.3 Object methods.....	62
5.2 DeviceComponent.py.....	64
5.3 * Example object class hierarchy for Fan DeviceComponent.....	67
5.4 * Example component class relationships for IpInterface.....	75
5.5 zendmd and the ZMI as tools to understand objects.....	77
5.5.1 The Zope Management Interface (ZMI).....	77
5.5.2 zendmd.....	80
6 Developing complex ZenPacks.....	84
6.1 Planning considerations.....	84
6.1.1 Names and naming convention.....	84
6.1.2 ZenPack prerequisites and other considerations.....	85
6.2 zenpacklib.....	86
6.3 Developing Python code.....	86
6.3.1 pyflakes.....	87
6.4 Developing GUI code.....	88
6.5 Useful tricks for ZenPack developers.....	88
7 Anatomy of a ZenPack.....	89
7.1 Basic principles.....	89
7.1.1 Configuration data, modeler plugins and the zenmodeler daemon.....	89
7.1.2 Performance data and monitoring templates.....	93
7.2 New objects in ZenPacks.....	94
7.3 GUI code.....	96
7.3.1 Page Template files and skins directories in older Zenoss.....	96
7.3.2 JavaScript code to define GUI elements.....	97
7.3.3 configure.zcml, infos and interfaces.....	97
7.4 Other elements of a ZenPack.....	101
8 zenpacklib UserGroup sample ZenPack.....	102
8.1 Requirements specification.....	103
8.1.1 bash commands to access user and group information.....	103
8.2 ZenPack specification.....	104
8.3 Installing zenpacklib.....	105
8.3.1 PyYAML.....	105

8.3.2	Installing zenpacklib.....	106
8.4	Creating the ZenPack.....	107
8.5	zenpack.yaml.....	109
8.5.1	zProperties.....	110
8.5.2	Zenoss device classes.....	110
8.5.3	Object classes.....	111
8.5.4	Relationships.....	115
8.6	Deploying and testing the ZenPack.....	118
8.7	Modeler plugin.....	121
8.7.1	Design details.....	121
8.7.2	UserGroupMap modeler plugin code.....	122
8.7.2.1	Creating the directory hierarchy.....	122
8.7.2.2	Imports from other Python modules.....	122
8.7.2.3	Base class for the UserGroupMap modeler plugin.....	123
8.7.2.4	Using zProperties in the modeler plugin.....	124
8.7.2.5	CommandPlugin command.....	125
8.7.2.6	The process method of the modeler plugin.....	126
8.7.3	Testing the modeler.....	134
8.7.4	Where do things go wrong with modelers?.....	135
8.8	* Renderers.....	138
8.9	Templates and zenpacklib.....	139
8.9.1	Creating a User component template with the GUI.....	140
8.9.2	Exporting templates with zenpacklib.....	142
8.10	* Creating object methods with zenpacklib.....	146
8.10.1	Writing methods for objects.....	146
9	SNMP LogMatch sample ZenPack.....	149
9.1	Requirements specification.....	149
9.2	ZenPack specification.....	155
9.3	Creating the ZenPack.....	156
9.4	Creating device and component object classes.....	157
9.4.1	Checking the device attributes and relationship.....	159
9.5	Creating the component modeler.....	160
9.5.1	* SNMP modeler code in core Zenoss.....	162
9.5.2	The LogMatchMap modeler plugin for component data.....	164
9.5.3	Testing the modeler.....	169
9.5.4	The LogMatchDeviceMap modeler for the device.....	172
9.5.5	Where do things go wrong with SNMP modelers?.....	173
9.6	GUI display code.....	174
9.6.1	JavaScript for new components.....	174
9.6.2	info.py.....	178
9.6.3	interfaces.py.....	180
9.6.4	configure.zcml.....	181
9.6.5	Where do things go wrong with GUI display code?.....	183
9.6.6	* Architecture of the ComponentPanel.....	184

9.7 Adding component performance templates.....	185
9.8 Adding other ZenPack elements through the GUI.....	187
9.9 Finalising the ZenPack.....	188
9.10 Extending the ZenPack to modify the device Overview.....	188
9.10.1 custom-overview-device.js.....	189
9.10.2 browser/configure.zcml.....	190
9.10.3 info.py.....	190
9.10.4 interfaces.py.....	191
9.10.5 Top-level configure.zcml.....	191
9.10.6 Testing the new changes.....	191
9.11 Modifying the ZenPack to remove LogMatchDevice.....	192
9.11.1 Monkey patching standard objects in __init__.py.....	193
9.11.2 LogMatch.py.....	195
9.11.3 browser/configure.zcml.....	196
9.11.4 LogMatchMap modeler plugin.....	196
9.11.5 Remove / install ZenPack.....	197
10 Rewriting the LogMatch ZenPack with zenpacklib.....	199
10.1 Creating ZenPacks with zenpacklib.....	199
10.2 zenpacklib capabilities.....	200
10.3 Converting the logmatch ZenPack for zenpacklib.....	200
10.3.1 zenpacklib benefits - items no longer required.....	200
10.3.2 zenpack.yaml.....	201
10.3.3 zenpack.yaml elements in modeler plugins.....	204
10.3.4 Completing the ZenPack.....	204
10.3.5 JavaScript to modify the device Overview panel.....	205
11 COMMAND DirFile sample ZenPack.....	206
11.1 Requirements specification.....	207
11.2 ZenPack specification.....	207
11.3 Creating the ZenPack.....	208
11.4 zenpack.yaml.....	209
11.5 DirFileMap modeler plugin.....	213
11.5.1 * CommandPlugin code in core Zenoss.....	214
11.5.2 Using zProperties in the modeler plugin.....	217
11.5.3 CommandPlugin command.....	218
11.5.4 The process method of the modeler plugin.....	219
11.5.5 * What's in an object map?.....	224
11.5.6 zenpacklib and the modeler plugin.....	225
11.5.7 Testing the DirFileMap modeler.....	226
11.5.7.1 * Analysing the zenmodeler log.....	228
12 Collecting performance data.....	230
12.1 Testing environment for the ZenPack.....	231
12.2 Collecting device performance data.....	232
12.2.1 * Analysing the zencommand debug log.....	236
12.3 Collecting component performance data.....	237

12.3.1	Specific component command; single value returned.....	237
12.3.2	Specific component command; multiple values returned.....	238
12.3.3	Generic component command with parser.....	242
12.3.4	Customised datasource to pass customised key values.....	246
12.3.4.1	getDescription method.....	250
12.3.4.2	useZenCommand method.....	250
12.3.4.3	getCommand method.....	251
12.3.4.4	addDataPoints method.....	251
12.3.4.5	Infos, Interfaces and configure.zcml.....	251
12.3.4.6	Testing the new datasource.....	254
12.4	Performance templates and zenpacklib.....	256
12.4.1	Where do things go wrong?.....	258
12.4.1.1	Issues with custom datasources and templates in zenpack.yaml.....	258
13	Converting COMMAND ZenPacks to PythonCollector.....	259
13.1	ZenPacks.zenoss.PythonCollector.....	260
13.1.1	Using the PythonCollector ZenPack.....	261
13.1.2	* Anatomy of a PythonDataSourcePlugin.....	263
13.2	Twisted.....	265
13.3	Creating Python datasources.....	267
13.3.1	Collecting device performance data.....	269
13.3.1.1	Imports for the PythonDataSourcePlugin.....	269
13.3.1.2	proxy_attributes and config_key method for the PythonDataSourcePlugin.....	270
13.3.1.3	collect method for the PythonDataSourcePlugin.....	272
13.3.1.4	onResult method for the PythonDataSourcePlugin.....	276
13.3.1.5	onSuccess method for the PythonDataSourcePlugin.....	277
13.3.1.6	onError method for the PythonDataSourcePlugin.....	277
13.3.1.7	Testing the new PythonDataSourcePlugin.....	278
13.3.1.8	Performance template to drive the PythonDataSourcePlugin.....	278
13.3.2	* Blocking and non-blocking in Twisted.....	280
13.3.2.1	* Comparing blocking and non-blocking collect methods.....	283
13.3.3	Collecting component performance data; specific component command; single value returned.....	284
13.3.3.1	Using a dsplugins directory.....	285
13.3.3.2	Imports, proxy_attributes, config_key and params.....	285
13.3.3.3	* A closer look at the usage of config_keys.....	286
13.3.3.4	collect method.....	289
13.3.3.5	onResult, onSuccess and onError methods.....	290
13.3.3.6	Performance template to drive the PythonDataSourcePlugin.....	291
13.3.4	Collecting component performance data; specific component command; multiple values returned. Nagios plugin conversion.....	292
13.3.4.1	Imports, proxy_attributes, config_key and params.....	293
13.3.4.2	collect method.....	294
13.3.4.3	onResult, onSuccess and onError methods.....	294

13.3.4.4 Performance template to drive the PythonDataSourcePlugin.....	296
13.3.5 Collecting component performance data; generic component command with parser.....	297
13.3.5.1 Imports, proxy_attributes, config_key and params.....	298
13.3.5.2 onSuccess method.....	299
13.3.5.3 Performance template to drive the PythonDataSourcePlugin.....	300
13.3.6 Collecting component performance data; customised datasource to pass customised key values.....	301
13.3.6.1 Building the Python datasource.....	302
13.3.6.2 Deploying the new datasource.....	306
13.4 Converting the modeler to use the PythonCollector ZenPack.....	307
13.4.1 Imports.....	307
13.4.2 Creating a dirRegex directory from zProperties.....	308
13.4.3 DirFilePythonMap class attributes.....	309
13.4.4 collect method.....	310
13.4.5 process method.....	312
13.4.6 Testing the new modeler.....	313
13.5 Combining performance data and modeler data.....	314
14 Events in ZenPacks.....	317
14.1 Detecting duplicate events.....	317
14.2 Event auto-clearing mechanism.....	318
14.3 Exploring the use of event class attributes.....	318
14.3.1 Detecting “repeat” events.....	322
14.3.2 Auto-clearing events.....	322
14.4 Adding transforms to events.....	324
14.5 Providing event details in a ZenPack.....	327
14.6 Providing triggers and notifications in a ZenPack.....	329
14.6.1 * Trigger and notification architecture.....	330
14.6.1.1 Finding trigger details.....	330
14.6.1.2 Finding notification details.....	333
14.6.1.3 Dumping trigger and notification details.....	335
14.6.2 ZenPack file for triggers and notifications.....	335
14.7 Resolving issues with triggers and notifications.....	336
14.8 Known issues with event fields, notifications and triggers.....	337
15 Creating menus in ZenPacks.....	338
15.1 The jargon.....	338
15.1.1 Zenoss 2 (some of which is still relevant!).....	338
15.1.2 Zenoss 3 / 4 / 5.....	340
15.2 Extending Command menus with the GUI.....	341
15.3 ZenPacks.community.MenuExamples.....	342
15.3.1 New device class, device object class and component class.....	344
15.3.2 Menu defined in __init__.py.....	344
15.3.3 Old and new options for page templates for menus.....	345
15.3.4 New-style menus limited to specific device types.....	350

15.3.5	Dropdown menus shipped in objects.xml.....	351
15.3.6	Adding items to the Display dropdown for a component.....	358
15.3.7	Menu on INFRASTRUCTURE -> Devices to add new device type.....	360
15.3.7.1	Routers and facades.....	364
15.3.8	New items for left-hand DeviceClass Action menu.....	365
15.3.9	Adding new items to a device's Action menu.....	368
15.3.10	Adding a new menu to the Footer bar.....	371
16	Testing and debugging ZenPacks.....	377
16.1	Log files and logging.....	377
16.1.1	Log messages and their likely meanings.....	378
16.2	General hints and tips.....	379
16.3	Testing and debugging new object class files.....	380
16.3.1	New components do not appear in left-hand menu.....	380
16.4	Testing and debugging modeler plugins.....	381
16.4.1	Compilation errors.....	381
16.4.2	General modeler debugging hints.....	383
16.4.3	Attributes or relationships are not populated.....	384
16.4.4	Modeler issues related to using zenpacklib.....	385
16.5	Testing and debugging problems with performance data.....	385
16.5.1	Configuration issues.....	385
16.5.2	Checking for collected performance data.....	387
16.5.3	Test buttons in datasources.....	388
16.5.4	Issues with datasource plugins.....	389
16.5.5	Issues with datasources.....	390
16.5.6	Performance collection issues related to using zenpacklib.....	390
16.6	Testing skins files and JavaScript files.....	391
16.6.1	General failure errors.....	391
16.6.2	Problems displaying components.....	393
16.6.3	Issues with Info and Interface definitions and configure.zcml.....	393
16.6.4	GUI issues when using zenpacklib.....	395
16.7	Testing and debugging problems with event elements.....	395
16.8	Problems with installing / removing ZenPacks.....	395
17	Developing a ZenPack and making it publicly available.....	396
17.1	Simple procedure for git development.....	396
17.2	Working with GitHub.....	399
17.2.1	Using ssh authentication with GitHub.....	400
17.2.2	Creating the GitHub repository.....	402
17.3	git branches.....	403
17.4	Cloning from GitHub to a local machine.....	403
17.5	Other ways to use GitHub.....	403
17.6	ZenPacks on the Zenoss wiki.....	404
	References.....	410
	ZenPack Reference.....	415
	Acknowledgements.....	417

About the author.....417